کتابچه نمونه سوالات چیست:

سایت ما **افتخار** دارد برای اولین بار در ایران توانسته است کتابچه نمونه سوالات تمام دروس پیام نور که هر یک حاوی تمامی آزمون های برگزار شده پیام نور (تمامی نیمسالهای موجود حتی الامکان با جواب) را در یک فایل به نام کتابچه جمع آوری کند و هر ترم نیز آن را آپدیت نماید.

مراحل ساخت یک کتابچه نمونه سوال

(برای آشنایی با زحمت بسیار زیاد تولید آن در هر ترم) :

دسته بندی فایلها – سرچ بر اساس کد درس – چسباندن سوال و جواب – پیدا کردن یک درس در نیمسالهای مختلف و چسباندن به کتابچه همان درس – چسباندن نیمسالهای مختلف یک درس به یکدیگر – وارد کردن اطلاعات تک تک نیمسالها در سایت – آپلود کتابچه و خیلی موارد دیگر..

همچنین با توجه به تغییرات کدهای درسی دانشگاه استثنائات زیادی در ساخت کتابچه بوجود می آید که کار ساخت کتابچه را بسیار پیچیده می کند .

WWW.PNUEB.COM

فهرست

تمرینات فصل اول

۱- شش تیم فوتبال در یک لیگ حرفه ای فوتبال با نام های:

Pers, Vultures, Eaglers, Tigers, Lions

برای برگزاری مسابقات فوتبال حاضر هستند. تیم *Vultures* با تیم های *Eagles* و *Lions* بازی کرده است همچنین *Lions* با تیم های *Beavers, Pers* بازی کرده است و تیم *Tigers* با تیمهای *Pers, Eagles* بازی کرده اند. با در نظر گرفتن اینکه هر تیم در هفته فقط یک بازی می تواند انجام دهد، زمانبندی را ادامه دهیم که در آن همه تیم ها با هم بازی کرده باشند به طوری که انجام بازی ها در کمترین تعداد هفته ها انجام شده باشد.

پاسخ :

برای راحتی کار، تیم ها را به ترتیب با حروف زیر نمایش می دهیم:

P,V,E,B,T,L

با توجه به اینکه نیم *L* تنها یک بازی انجام داده و باید با چهار نیم دیگر بازی کند، می‌فهمیم که حداقل به چهار هفته دیگر نیاز داریم. بازی ها را در چهار هفته، به صورت زیر برنامه ریزی می‌کنیم.

هفته اول: تیم *L* با *T*، تیم *E* با *B*، تیم *V* با *P*.

هفته دوم: تیم *L* با *E*، تیم *T* با *V*، تیم *P* با *B*.

هفته سوم: تیم *T* با *B*، تیم *P* با *E*.

هفته چهارم: تیم *V* با *E*.

۲- زمان اجرای الگوریتم (یعنی $T(n)$) را برای هر کدام از الگوریتم های زیر محاسبه نمایید؟

الف)

```
x = ۰
for(i = ۰ ; i < n ; i++)
    for (j = i ; j < n ; j++)
        x++ ;
```

پاسخ :

سطر	هزینه	تعداد
۱	$C_۱$	۱
۲	$C_۲$	n
۳	$C_۳$	$n+(n-۱)+\ldots+۱$
۴	$C_۴$	$(n-۱)+(n-۲)+\ldots+۱$

پس هزینه کل برابر است با:

$$T(n) = C_1 + C_2 n + C_3\left(n + (n-1) + \ldots + 1\right) + C_4\left((n-1) + (n-2) + \ldots + 1\right)$$

با انتخاب $C = \max\{C_1, C_2, C_3, C_4\}$ داریم:

$$T(n) \le C\left[1 + n + \frac{n(n+1)}{2} + \frac{(n-1)n}{2}\right] = C(n^2 + n + 1)$$

در نتیجه $T(n) \in O(n^2)$

ب)

```
S = 0 ;
for (i = 0 ; i < n ; i++)
    for(j = 0 ; j < i ; j++)
        S++ ;
```

پاسخ :

سطر	هزینه	تعداد
۱	C_1	۱
۲	C_2	n
۳	C_3	۱+۲+...+$(n-1)$
۴	C_4	۱+۲+....+$(n-2)$

پس هزینه کل برابر است با:

$$T(n) = C_1 + C_2 n + C_3\left(1 + 2 + \ldots + (n-1)\right) + C_4\left(1 + 2 + \ldots + (n-2)\right)$$

با انتخاب $C = \max\{C_1, C_2, C_3, C_4\}$ داریم:

$$T(n) \le C\left[1 + n + \frac{n(n-1)}{2} + \frac{(n-1)(n-2)}{2}\right] = C\left(n^2 - n + 2\right)$$

در نتیجه $T(n) \in O(n^2)$

ج)

```
P = 0 ;
for (i = 1 ; i < n ; i++)
    for(j = i + 1 ; j <= m ; j++)
        p++ ;
```

پاسخ :

سطر	هزینه	تعداد
۱	C_1	۱
۲	C_2	$n-1$
۳	C_3	$(m-1)+(m-2)+\ldots+(m-n+1)$
۴	C_4	$(m-2)+(m-3)+\ldots+(m-n)$

پس هزینه کل برابر است با:

$$T(m,n)=C_1+C_2(n-1)+C_3\big((m-1)+(m-2)+\ldots+(m-n+1)\big)+C_4\big((m-2)+\ldots+(m-n)\big)$$

با انتخاب $C=\max\{C_1,C_2,C_3,C_4\}$ داریم:

$$T(m,n)\le C\left[1+(n-1)+\frac{(m-n+1)(m-n)}{2}-\frac{(m-1)m}{2}+\frac{(m-n)(m-n+1)}{2}-\frac{(m-2)(m-1)}{2}\right]$$
$$=n^2+3m+2mn-1$$

با توجه به سطر ۳ می توان فرض کرد $m>n$ پس $mn>n^2$ و در نتیجه $T(m,n)\in O(mn)$.

د)

```
for (i = 0 ; j < n; i++)
    for (j = i+1 ; j < n ; j++)
        for (k = j+1; k < n ; k++)
            m++;
```

پاسخ :

سطر	هزینه	تعداد
۱	C_1	۱
۲	C_2	$n-1$
۳	C_3	$(m-1)+(m-2)+\ldots+1$
۴	C_4	$\sum_{p=1}^{n-2}P+\sum_{p=1}^{n-3}P+\ldots+\sum_{P=1}^{1}P$
۵	C_5	$\sum_{p=1}^{n-3}p+\sum_{p=1}^{n-4}p+\ldots+\sum_{p=1}^{1}P$

توضیح: تعداد سطر ۱ که واضح است. برای سطر ۲ از فرمول بیان شده در خلاصه درس استفاده کرده ایم. برای سطر ۳ دیده می شود که هنگامی که i برابر صفر است، jبرابر یک است و تعداد $n-1$ به دست می آید. به همین روال برای $i=1$ به دست می آید $j=2$ و تعداد $n-2$ به دست می آید، الی آخر، برای سطر ۴ می گوییم برای $i=0$ داریم $j=1,2,3,...,n-1$. برای $j=1$ داریم $k=2$ و تعداد $n-2$ به دست می آید. برای $j=2$، داریم $k=3$ و تعداد $n-3$ به دست می آید، و برای $j=n-1$ تعداد صفر تا به دست می‌آید. پس می توان گفت برای $i=0$ تعداد زیر به دست می آید:

$$(n-2)+(n-3)+...+1=\sum_{p=1}^{n-2}P$$

به طریق مشابه، برای $i=1$، با در نظر گرفتن $j=2,3,...,n-1$، تعداد $\sum_{p=1}^{n-3}P$ به دست می آید و الی آخر.

برای سطر ۵، از هر پرانتز در $1+...+(n-3)+(n-2)$ یکی کم می کنیم و به دست می آید $\sum_{p=1}^{n-3}P=0+1+...+(n-4)+(n-3)$. همین تغییر را در سیگماهای دوم، سوم، ... و آخر اعمال می‌کنیم و حاصل به دست می آید.

پس از قرار دادن $C=\max\{C_1,C_2,C_3,C_4\}$ به دست می آید:

$$T(n)\le C\left[1+n+(n-1)+...+1+\left(\sum_{p=1}^{n-2}P+...+\sum_{P=1}^{1}P\right)+\left(\sum_{p=1}^{n-3}P+...+\sum_{P=1}^{1}P\right)\right]$$

و در نتیجه $T(n)\in O(n^3)$.

توضیح: در آخرین عبارت بالا، در پرانتز اول، حاصل سیگمای اول برابر است با:

$$\sum_{p=1}^{n-2}p=\frac{(n-2)(n-3)}{2}$$

که یک عبارت درجه دوم است. همچنین سیگماهای بعدی آن. پس داخل پرانتز، $n-۲$ عـدد سیگما داریم که پس از فاکتور گیری از ضرایب، شامل عبارت

$$۱^۲ + ۲^۲ + ۳^۳ + ... + (n-۲)^۲$$

می شود که عبارت فوق از مرتبه $\theta(n^۳)$است. همچنین در پرانتز دوم هم عبارتی به دسـت می‌آید که از $\theta(n^۳)$ است. و چون $T(n) \le \theta(n^۳)$ پس $T(n) \in O(n^۳)$.

نکته: در رابطه بالا از فرمول زیر، که به اتحاد وایرشتراس معروف است، استفاده شده است.

$$۱^k + ۲^k + ۳^k + ... + n^k \in \theta(n^{k+۱})$$

هـ)

```
L = ۰;
for(i = ۰ ; j < n ;i++)
    for(j = ۰ ; j < m ; j++)
        for (k = ۰ ; k < p ; k++)
            L++ ;
```

پاسخ :

تعداد	هزینه	سطر
۱	$C_۱$	۱
n	$C_۲$	۲
$(n-۱)m$	$C_۳$	۳
$(n-۱)(m-۱)p$	$C_۴$	۴
$(n-۱)(m-۱)(p-۱)$	$C_۵$	۵

پس از انتخاب $C = \max\{C_۱, C_۲, C_۳, C_۴, C_۵\}$ به دست می آید:

$$T(n,m,p) \le C[۱ + n + (n-۱)m + (n-۱)(m-۱)p + (n-۱)(m-۱)(p-۱)]$$

و در نتیجه:

$$T(n,m,p) \in O(nmp)$$

۳- زمان اجرای الگوریتم های زیر را محاسبه نمایید:

الف)

```
i = n ;
While(i >=۱) {
   /* Some Statement requiring θ (۱) timr */
      i = i / ۲ ;
  }
```

پاسخ :

تعداد تکرار حلقه برابر $\lfloor \log_{۲}^{n} \rfloor$ می باشد پس:

$$T(n) = \lfloor \log_{۲}^{n} \rfloor + ۱ \in \theta(\log n)$$

ب)

```
i = ۱
While(i <= n) {
/* Some Statement requiring θ(۱)time */
      i = i * ۲;
}
```

پاسخ :

تعداد تکرار حلقه برابر $\lfloor \log_{۲}^{n} \rfloor$ می باشد، پس:

$$T(n) = \lfloor \log_{۲}^{n} \rfloor + ۱ \in \theta(\log n)$$

ج)

```
i = n ;
While ( i >=۱){
    j = i ;
    While (j <= n) {
      /* Some Statment requiring θ(۱) time */
      j = j * ۲ ;
    }
    i = i / ۲ ;
}
/* Suppose that n > m */
```

پاسخ :

تحلیل را به صورت زیر انجام می دهیم، فرض کنیم $n = ۲^k$، پس

	i	j	اجرای حلقه اصلی
$k+۱$ بار	n	n	$\log_۲^۱ + ۱$
	$n/۲$	$n/۲$	$\log_۲^۲ + ۱$
	$n/۴$	$n/۴$	$\log_۲^۴ + ۱$
	⋮		
	n/n	n/n	$\log_۲^n + ۱$

$$T(n) = \log_۲^۱ + ۱ + \log_۲^۲ + ۱ + \log_۲^۴ + ۱ + \ldots + \log_۲^n + ۱$$

$$= \left(\log_۲^۱ + \log_۲^۲ + \log_۲^۴ + \ldots + \log_۲^n\right) + \underbrace{(۱ + ۱ + \ldots + ۱)}_{\text{بار } k+۱}$$

$$= (۰ + ۱ + ۲ + \ldots + k) + (k+۱) = \frac{k(k+۱)}{۲} + (k+۱)$$

$$= \frac{k(k+۱) + ۲(k+۱)}{۲} = \frac{(k+۱)(k+۲)}{۲} = \frac{۱}{۲}\left(\log_۲^n + ۱\right)\left(\log_۲^n + ۲\right)$$

$$\in \theta\left((\log_۲^n)^۲\right)$$

د)

```
While (n > ۰) {
    r = n % m ;
    n = m ;
    m = r ; }
```

پاسخ :

طول جهش هر پله‌ی این حلقه برابر $(n \bmod m)$ است. چون این عدد را نمی توان به طور صریح مشخص کرد (بر حسب n) پس بدترین حالت را پیدا خواهیم کرد.

واضح است که همواره داریم $(n \bmod m) < \frac{n}{۲}$. این رابطه را می توان از قوانین مقدماتی حساب، به راحتی نتیجه گرفت. پس تعداد دفعات تکرار این حلقه از تعداد دفعات حلقه‌ای که طول جهش هر پله آن برابر $\frac{n}{۲}$ است، اکیداً کمتر می باشد. حلقه ای که طول جهش هر پله

آن برابر $\frac{n}{۲}$ باشد، (مانند قسمت های الف و ب از همین سؤال)، به تعداد $\lfloor \log_۲^n \rfloor + ۱$ تا تکرار می‌شود. پس تعداد دفعات تکرار این الگوریتم، برابر خواهد بود $O(\log n)$.

دقت شود که چون $(n \bmod m)$ هیچگاه مساوی $\frac{n}{۲}$ نمی شود، پس هیچگاه این O به θ تبدیل نمی شود.

۴- ثابت کنید که عبارت زیر برقرار هستند:

۱) $۷n^۲ - ۶n + ۲ \in \theta(n^۲)$

۲) $n^۳ + n^۲ \log n \in \theta(n^۳)$

۳) $n! + ۷n^۵ \in O(n^n)$

۴) $۷n^۲ ۲^n + ۵n^۲ \log n \in \theta(n^۲ ۲^n)$

۵) $۲n^{۲n} + ۷ ۲^n \in \theta(n^{۲^n})$

۶) $\sum_{i=۰}^{n} i^۳ \in \theta(n^۴)$

۷) $n^۴ + ۷۰۰n^۲ \in \theta$

۸) $۱۲^{n^۵} \Big/ \log^n + ۷n^۴ \in O(n^۵)$

۹) $۲n^۳ + ۷n^۲ \in \Omega(n^۲)$

۱۰) $۲n^۳ + ۷n^۲ \in \Omega(n^۳)$

پاسخ :

از قضیه ۴-۱ استفاده می کنیم:

۱) $\lim_{n\to\infty} \frac{۷n^۲ - ۶n + ۲}{n^۲} = ۷ \Rightarrow ۷n^۲ - ۶n + ۲ \in \theta(n^۲)$

۲) $\lim_{n\to\infty} \frac{n^۳ + n^۲ \log n}{n^۳} = ۱ \Rightarrow n^۳ + n^۲ \log n \in \theta(n^۳)$

۳) $\lim_{n\to\infty} \frac{n! + ۷n^۵}{n^۲ ۲^n} = ۰ \Rightarrow n! + ۷n^۵ \in O(n^n)$

۴) $\lim_{n\to\infty} \frac{۷n^۲ ۲^n + ۵n^۲ \log n}{n^۲ ۲^n} = ۷ \Rightarrow ۷n^۲ ۲^n + ۵n^۲ \log n \in \theta(n^۲ ۲^n)$

۵) $\lim_{n\to\infty} \frac{۲n^{۲^n} + ۷ ۲^n}{n^{۲^n}} = ۲ \Rightarrow ۲n^{۲^n} + ۷ ۲^n \in \theta(n^{۲^n})$

۶) $\lim_{n\to\infty}\frac{\sum_{i=0}^{n} i^{۳}}{n^{۴}} = \lim_{n\to\infty}\frac{(\frac{n(n+1)}{۲})^{۲}}{n^{۴}} = \frac{۱}{۴} \Rightarrow \sum_{i=0}^{n} i^{۳} \in \theta(n^{۴})$

۷) $\lim_{n\to\infty}\frac{n^{۴}+۷۰۰n^{۲}}{n^{۴}} = ۱ \Rightarrow n^{۴}+۷۰۰n^{۲} \in \theta(n^{۴})$

۸) $\lim_{n\to\infty}\frac{۱۲^{n^{۵}/\log^{n}}+۷n^{۴}}{n^{۵}} = ۰ \Rightarrow ۱۲^{n^{۵}/\log^{n}}+۷n^{۴} \in O(n^{۵})$

۹) $\lim_{n\to\infty}\frac{۲n^{۳}+۷n^{۲}}{n^{۲}} = \infty \Rightarrow ۲n^{۳}+۷n^{۲} \in \Omega(n^{۲})$

۱۰) $\lim_{n\to\infty}\frac{۲n^{۳}+۷n^{۲}}{n^{۳}} = ۱ \Rightarrow ۲n^{۳}+۷n^{۲} \in \theta(n^{۳}) \subseteq \Omega(n^{۳})$

۵- *Omega-oh , Big-oh*، توابع زمانی زیر را محاسبه کنید.

الف) $T(n) = n^{۲}+۱۰۰۰n$

پاسخ :

$T(n) \in O(n^{۳})$, $T(n) \in \Omega(n)$

ب) $T(n) = \begin{cases} n & if\ n\ is\ odd \\ n^{۳} & other\ wise \end{cases}$

پاسخ :

$T(n) \in O(n^{۴})$, $T(n) \in \Omega\left(\sqrt{n}\right)$

ج) $T(n) = \begin{cases} n & if \quad n \le ۱۰۰۰ \\ n^{۳} & if \quad n > ۱۰۰۰ \end{cases}$

پاسخ :

$T(n) \in O(n^{۴})$, $T(n) \in \Omega(n^{۲})$

۶- فرض کنید $T_{۱}(n) \in \Omega(f(n))$, $T_{۲}(n) \in \Omega(g(n))$ باشند. درستی یا نادرستی عبارات زیر را بررسی کنید.

۱) $T_{۱}(n)+T_{۲}(n) \in \theta(\max\{f(n), g(n)\})$

۲) $T_{۱}(n) * T_{۲}(n) \in \Omega(\max\{f(n), g(n)\})$

۳) $T_{۱}(n)+T_{۲}(n) \in \theta(\max\{f(n), g(n)\})$

۴) $T_{۱}(n) * T_{۲}(n) \in \theta(f(n)) * (g(n))$

پاسخ :

فقط رابطه ۲ صحیح است. برای روابط دیگر می توان مثال نقض زیر را ارائه داد:

$$T_1(n) = T_2(n) = 2^n \quad , \quad f(n) = g(n) = n^2$$

تذکر: به نظر می رسد که این سؤال، اشکال چاپی دارد. ممکن است منظور طراح بر این فرض بوده است که: $T_1(n) \in \theta(f(n))$, $T_2(n) \in \theta(f(n))$، و چون روابط ۱ و ۳ مثل هم می‌مانند، احتمالاً در رابطه ۱ باید قرار دهیم Ω، در این صورت چون $\theta(f) \subseteq \Omega(f)$، هر چهار رابطه صحیح می باشند.

۷- درستی یا نادرستی عبارات زیر را بررسی کنید .

۱) $n^5 + 14n^3 \in \theta(n^3)$

پاسخ :

از قضیه ۴-۱ استفاده می کنیم.

$$\lim_{n\to\infty} \frac{n^5 + 14n^3}{n^3} = \infty \Rightarrow$$ رابطه غلط است

۲) $n^5 + 14n^3 \in \theta(n^5)$

پاسخ :

$$\lim_{n\to\infty} \frac{n^5 + 14n^3}{n^5} = 1 \Rightarrow n^5 + 14n^3 \in \theta(n^5) \Rightarrow$$ رابطه صحیح است

۳) $n^5 + 14n^3 \in \Omega(n^3)$

پاسخ :

$$\lim_{n\to\infty} \frac{n^5 + 14n^3}{n^3} = \infty \Rightarrow$$ رابطه صحیح است

۴) $2^{2^n} \in \theta(2^n)$

پاسخ :

$$\lim_{n\to\infty} \frac{2^{2^n}}{2^n} = \infty \Rightarrow$$ رابطه غلط است

۵) $n^6 + 7n^5 + 12 \in \theta(n^7)$

پاسخ :

$$\lim_{n\to\infty} \frac{n^6 + 7n^5 + 12}{n^7} = 0 \Rightarrow$$ رابطه غلط است

۶) $۷n^{۱۰۷}+۱۰۰۰n \in \theta(n^{۲})$

پاسخ :

$\lim_{n\to\infty} \frac{۷n^{۱۰۷}+۱۰۰۰n}{n^{۲}} = \infty \Rightarrow$ رابطه غلط است

۷) $۷n^{۱۰۷}+۱۰۰۰n \in \Omega(n^{۱۰۷})$

پاسخ :

$\lim_{n\to\infty} \frac{۷n^{۱۰۷}+۱۰۰۰n}{n^{۱۰۷}} = ۷ \Rightarrow$ چون $\theta(f) \subseteq \Omega(f)$، رابطه صحیح است

۸) $۲n^{۲.۸۱}+۱۰۰n^{۲} \in \theta(n^{۲})$

پاسخ :

$\lim_{n\to\infty} \frac{۲n^{۲.۸۱}+۱۰۰n^{۲}}{n^{۲}} = \infty \Rightarrow$ رابطه غلط است

۹- قضیه ۱-۱ را ثابت کنید.

پاسخ :

فرض کنیم $T(n) = a_m n^m + a_{m-۱}n^{m-۱} + ... + a_۱ n + a_۰$.

قابل توجه است که ممکن است بعضی از a_iها منفی باشند. با انتخاب $M = ۱$, $C = |a_m| + |a_{m-۱}| + ... + |a_۰|$ اولاً دیده می شود که $C > ۰$، ثانیاً برای هر $n \geq M$ داریم:

$$T(n) \leq |a_m| n^m + |a_{m-۱}| n^{m-۱} + ... + |a_۰| \leq$$
$$|a_m| n^m + |a_{m-۱}| n^m + ... + |a_۰| n^m = C\, n^m.$$

پس $T(n) \in O(n^m)$ و حکم ثابت است.

۱۰- قضیه ۲-۱ را ثابت کنید.

پاسخ :

ابتدا فرض می کنیم $T(n) \in \theta(g(n))$. پس $C_۲, C_۱$ مثبت و $M \in N$ موجود می باشند به طوری که:

$$\forall n \geq M \Rightarrow C_۲ g(n) \leq T(n) \leq C_۱ g(n).$$

پس می توان نوشت:

$$\forall n \geq M \Rightarrow \begin{cases} T(n) \leq C_1 g(n) \Rightarrow T(n) \in O(g(n)) \\ T(n) \geq C_2 g(n) \Rightarrow T(n) \in \Omega(g(n)) \end{cases}$$

به عکس، فرض می کنیم $T(n) \in O(g(n))$, $T(n) \in \Omega(g(n))$. پس C_2, C_1 مثبت و $M_1, M_2 \in N$ موجودند که

$$\forall n \geq M_1 \Rightarrow T(n) \leq C_1 g(n)$$

$$\forall n \geq M_2 \Rightarrow T(n) \geq C_2 g(n)$$

با انتخاب $M = \max\{M_1, M_2\}$، برای nهای بزرگتر از M، هر دو رابطه اخیر برقرارند.

پس:

$$\forall n \geq M \Rightarrow C_2 g(n) \leq T(n) \leq C_1 g(n)$$

و در نتیجه $T(n) \in \theta(g(n))$.

تمرینات تکمیلی فصل اول

۱- تابع $max\ (a,i)$ را در نظر بگیرید که در آن a یک آرایه n عنصری می باشد و I محل بزرگترین عنصر در آرایه است.

الف) تابع بازگشتی به نام max با پارامترها I,a بنویسید که بیشترین مقدار در یک آرایه را به دست آورد.

پاسخ :

الگوریتم غیر بازگشتی، سریعتر است.

```
k ← ۱
for j = ۲ to n
    if a[j] > a[k], then k ← j
end for
return a[k]
```

ب) $T(n)$ را برای تابع بالا در بهترین و بدترین حالت بررسی کنید.

پاسخ :

در بهترین حالت، بزرگترین عنصر برابر $a[۱]$ است، که در این صورت هیچکدام از *if* ها برقرار نمی‌شوند و تعداد کل دستورات برابر می شود با تعداد دو سطر اول؛ یعنی:

$$T(n) = ۱ + (n-۱) = n$$

در بدترین حالت بزرگترین عنصر برابر $a[n]$ است، که در این صورت همه *if* ها برقرار می شوند و تعداد کل دستورات برابر می شود با:

$$T(n) = ۱ + (n-۱) + (n-۲) = ۲n - ۲$$

ج) حالت متوسط $T(n)$ را محاسبه کنید.

پاسخ :

حالت میانگین برابر است با:

$$T(n) = \frac{۱}{۲}\left(n + (۲n-۲)\right) = \frac{۳n}{۲} - ۱$$

۲- یک ماتریس با مقادیر پیچیده به نام X توسط یک جفت از ماتریس های (A,B) ارائه می شود به نحوی که B,A هر دو حاوی مقادیر اعشاری می باشند.

الگوریتمی بنویسید که حاصلضرب دو ماتریس پیچیده (A,B) , (C,D) را محاسبه کند. به نحوی که:

$$(A,B)*(C,D) = (A,iB)*(C,iD) = (AC - BD) + i(AD + BC)$$

باشد سپس $T(n)$ را برای ماتریس $n \times n$ محاسبه نمایید.

پاسخ :

به نظر می رسد منظور طراح سؤال از کلمه پیچیده، همان مختلط است که ترجمه کلمه *complex* می‌باشد، و منظور از کلمه اعشاری، همان حقیقی است که ترجمه کلمه *real* است. رابطه بالای صفحه ۳۶ شامل اشکال چاپی است، که صحیح آن به صورت زیر است:

$$(A,B)*(C,D)=(A+iB)*(C+iD)=(AC-BD)+i(AD+BC).$$

حال، مسئله را در سه مرحله حل می کنیم:

۱- تابع ضرب دو ماتریس $n\times n$ به نام های Q,P را چنین می نویسیم:

```
Void   Mult(P[ ][ ],Q[ ][ ])
  {   for i = ۱ to n
          for j = ۱ to n
            {M[i][j]= ۰
              for k = ۱ to n
              M[i][j ]= M[i][j ]+ P[i][k ]* Q[k ][j ]
              }
    }
```

۲- تابع جمع دو ماتریس $n\times n$ به نام های Q,P را چنین می نویسیم:

```
Void   sum (P[ ][ ] , Q[ ][ ])
  {   for i = ۱ to n
          for j = ۱ to n
          S[i][j ]= P[i][j ]+ Q[i][J ]
    }
```

۳- با استفاده از فرمول عنوان شده در صورت مسئله، و مراحل ۱ و ۲ می نویسیم:

```
void     complexmult  (A[ ][ ], B[ ][ ], C[ ][ ], D[ ][ ])
{return      sum (Mult(A,C) , Mult(−B,D) )
                 + i sum(Mult(A,D) , Mut(B,C) )
    }
```

توجه : در مرحله ۳ علامت + صرفاً یک نماد است و هیچ گونه ارزش محاسباتی ندارد.

۳- یک آرایه n عنصری را در نظر بگیرید سپس توابعی برای موارد زیر بنویسید:

الف) توسط الگوریتم *shell sort* عناصر آرایه را مرتب کنید.

پاسخ :

در الگوریتم *shell sort* ابتدا یک *Gap* به اندازه طول آرایه در نظر می گیریم که این *Gap* در هر مرحله نصف می گردد و عناصر با این فاصله، با هم مقایسه می شوند.

```
Void    Shell – sort(A [ ] )
{       while   ( (gap / = ۲)>=۱)
        { for    i = ۰  to  lenght (A)
          { j = i
            while ( (j > gap) and (A[j – gap]> A[j] )
            {   Swap(A[j – gap], A[j] )
                j = gap
            }
          }
        }
      }
```

ب) توسط الگوریتم *selection* عناصر آرایه را مرتب کنید.

پاسخ :

```
Void    Selection – Sort (A[ ] )
{   for   i =۱   to  lenght (A) –۱
    {     k = i
          for    j = i +۱   to    lenght (A)
          {if   A[j]< A[k] ,   Then  k ← j
          }
      if  k ≠ i   Then   interchange  A[i] and A[k]
    }
}
```

ج) $T(n)$ را برای توابع بالا محاسبه کنید.

پاسخ :

برای *selection-sort* تعداد مقایسه ها بین صفر تا $\frac{n(n-۱)}{۲}$ است، و تعداد اعمال مقدماتی بین تا $۳(n-۱)$.

۴- یک آرایه n عنصری مرتب را در نظر گرفته توسط تابعی به نام *Search* عنصر x را در آن جستجو نمایید. سپس زمان آن را تحلیل نمایید.

پاسخ :

الگوریتم زیر آرایه $A[۱,...,n]$ و عنصر x را می گیرد، سپس اگر $x = A[j]$، آنگاه j را می‌دهد، و اگر x در A نبود صفر را می دهد.

```
Void   Search (A[ ], x)
{      L = ۱
       H = Lenght(A)
       j = ۰
       while ((L ≤ H) and (j = ۰))
              M = ⌊(L + H)/۲⌋
              if   x = A[M]  then  j ← M
              else if  x < A[M]  then   H ← M − ۱
                   else   L ← M + ۱
       end   while
       return  j
}
```

برای محاسبه بدترین زمان، می توان فرض کرد x از همه اعضای آرایه بزرگتر (یا مساوی با بزرگترین عنصر آرایه) است. دو حالت در نظر می گیریم:

۱- اگر n زوج باشد آنگاه تعداد اعضای $A[M+۱,...n]$ برابر $\frac{n}{۲}$ است.

۲- اگر n فرد باشد آنگاه تعداد اعضای $A[M+۱,...n]$ برابر $\frac{n-۱}{۲}$ است.

پس در هر دو حالت تعداد اعضای $A[M+۱,...n]$ برابر $\left\lfloor \frac{n}{۲} \right\rfloor$ است پس $\left\lfloor \frac{n}{۲} \right\rfloor$ به طریق مشابه تعداد اعضایی که در مرحله سوم مورد جستجو واقع می شوند برابر است با:

$\frac{\lfloor \lfloor \frac{n}{۲} \rfloor \rfloor}{۲}$ که مساوی است با $\left\lfloor \frac{n}{۴} \right\rfloor$

پس در مرحله jام، آرایه ای به طول $\left\lfloor \frac{n}{۲^{j-۱}} \right\rfloor$ مورد جستجو واقع می شود. چون بدترین حالت را در نظر گرفته ایم، پس j در رابطه $\left\lfloor \frac{n}{۲^{j-۱}} \right\rfloor = ۱$ صدق می کند. یعنی

$$۱ \le \frac{n}{۲^{j-۱}} < ۲ \Rightarrow \quad ۲^{j-۱} \le n < ۲^{j}$$

$$\Rightarrow \quad j-۱ \le \log n < j \Rightarrow j-۱ \lfloor \log n \rfloor$$

$$\Rightarrow \quad j = \lfloor \log n \rfloor + ۱$$

پس در بدترین حالت، تعداد مراحل انجام شده برابر است با $\lfloor \log n \rfloor + ۱$

۵- گروهی شامل n نمایشگر فوتبال که همه کلاهی نشانه طرفداری از تیم مورد علاقه خود بر سر دارند را در نظر بگیرید. در انتهای مسابقه علاقمندان تیم برنده، کلاه های خود را به هوا پرتاب می کنند. با فرض اینکه هر کلاه روی سر یکی پرتاب کننده ها فرود بیاید تعداد حالت هایی را که K نفر از پرتاب کننده ها کلاه اصلی خود را دریافت کنند چند است؟

پاسخ :

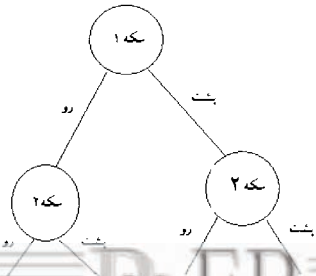
تعداد تماشاگران تیم برنده را با n نمایش می دهیم. با توجه به اینکه آن k نفر مورد نظر، کلاه خود را می گیرند، پس باید تعداد حالاتی را در نظر گرفت که $n-k$ کلاه باقیمانده به $n-k$ نفر می‌رسد، که این تعداد به وضوح برابر است با $(n-k)!$.

۶- یک سکه دو رو را در نظر بگیرید. با استفاده از این سکه ها، چگونه می توانید یک سکه n طرفه را شبیه سازی کنید.

پاسخ :

برای پیاده سازی سکه n طرفه، سکه اول را می اندازیم.

اگر هر رویی آمد، سکه دوم را می اندازیم و الی آخر:



پس برای پیاده سازی یک سکه nرو، احتیاج به $\log_2^n$ عدد سکه داریم.

الف) چه زمانی n توانی از ۲ می باشد؟

پاسخ :

زمانی که درخت ما، یک درخت پر باشد (درخت تام) آنگاه n توانی از ۲ می باشد.

ب) چه زمانی n توانی از ۲ نیست؟

پاسخ :

زمانی که n توانی از ۲ نباشد، درخت ما پر نبوده و برخی از حالات حذف می شوند.

تمرینات فصل دوم

۱- در تابع زیر *List* را یک آرایه n عنصری در نظر بگیرید:

```
int  S(int List[ ] , int n)
{    if (n==1)
        return (list [1]);
     else
        return (list [n]+ S(list , n-1) );
}
```

خروجی تابع بالا را تعیین کنید. در حالت کلی تابع بالا چه عملی انجام می دهد؟

پاسخ :

این تابع، مجموع اعضای آرایهٔ *List* را حساب می کند.

۲- تابع *func* را به صورت زیر در نظر بگیرید:

```
int Func (int n)
{
   if (n==1)
      return (1);
    else
      return (n+ func(n-1));
}
```

خروجی تابع بالا را به ازای n =۸ محاسبه کرده، سپس در حالت کلی عملی که تابع *Func* روی n انجام می دهد را بیان کنید؟

پاسخ :

به ازای هر n، این تابع حاصل n...+۳+۲+۱ را محاسبه می کند، که برابر است با $\frac{n(n+1)}{2}$

مثلاً برای n=۸، خروجی برابر است با ۳۶.

۳- در تابع زیر *List* یک آرایه n عنصری از اعداد صحیح می باشد:

```
int M (int List [ ], int n)
{
    if (n = = ۱)
        return (List[ ] );
    else
        return (List [n] , M (List , n −۱) );
}
```

خروجی تابع M را برای ۱۰ عدد صحیح محاسبه نموده، سپس عملی که تابع M در حالت کلی روی عناصر آرایه *List* انجام می دهد را بیان کنید؟

پاسخ :

در حالت کلی، این تابع اعضای آرایهٔ *List* را از آخر به اول می نویسد.

۴- تابع *test* یک درخت دو دوئی را دریافت می کند:

```
int test (Node* tree)
{
    if(tree = = Null)
        return ۰ ;
    else
        return (۱+ max (test (tree → left) + test(tree → right) ));
}
```

خروجی تابع *test* را در حالت کلی پیدا کرده سپس مراحل محاسبه تابع بازگشتی را برای آن بنویسید.

پاسخ :

سطر آخر برنامه شامل غلط چاپی است. شکل صحیح آن به این صورت است.

Return (۱+max(test (tree → left), test (tree → right)))

این تابع، عمق یک درخت را به دست می آورد.

تمرینات تکمیلی فصل دوم

۵- الگوریتم بازگشتی بنویسید تا کلیه ترکیبات ارقام ۱ تا n را تولید نماید.

توجه کنید کلیه ترکیبات ارقام از ۱ تا n-۱ را داشته باشیم.

پاسخ :

این الگوریتم چنین عمل می کند که هر عددی که در اول قرار می دهد، تمام ترکیبات n-۱ رقم دیگر را جلوی آن می نویسد. ابتدا هر عدد را به صورت یک آرایه تصور می کنیم.

```
for   j = 1   to   n
      P[j] ← j
perm (1)
procedure  perm (m)
if   m = n   then  output      P[1,....,n]
else   for   j = m   to   n
            {int erchang   P[j]  and  P[m]
              perm  (m − 1)
               int erchang     P[j] and  P[m]
              }
```

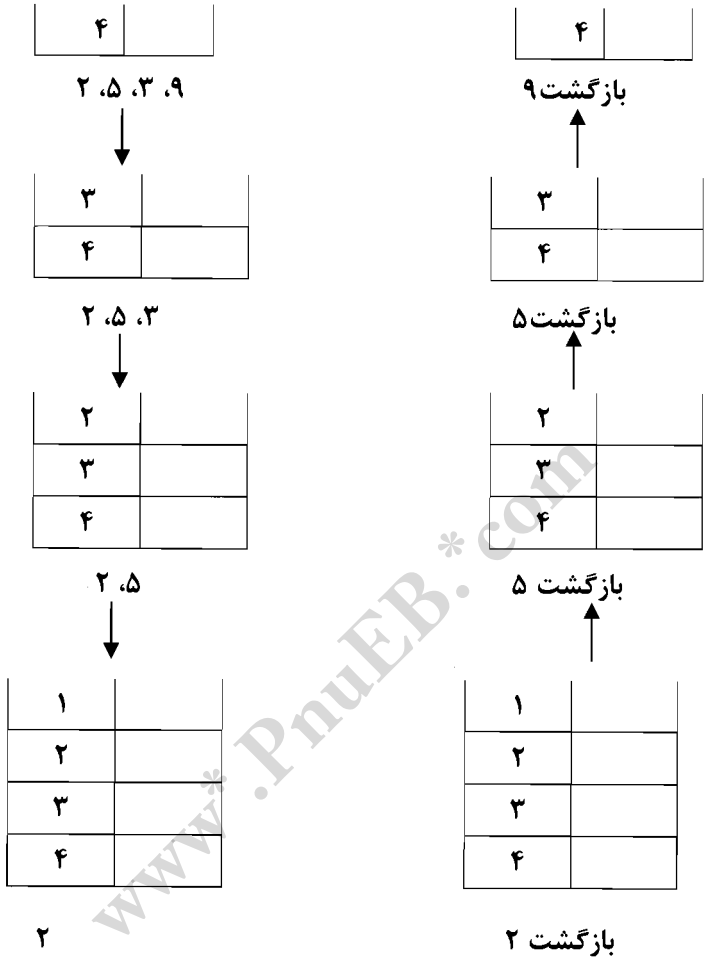
۶- الگوریتم بازگشتی برای یافتن بیشترین مقدار در یک آرایه از اعداد صحیح بنویسید.

سپس مراحل محاسبه مقدار تابع بازگشتی را با ارائه مثالی بحث نمایید.

پاسخ :

```
max (a[ ] , n)
{   if   n = 1  then  return   a[1]
    else  b = max  (a , n − 1)
    if    a[n] >= b  , then return  a[n]
    else  return   b
 }
```

روال الگوریتم را برای آرایهٔ ۲،۵،۳،۹ نشان می دهیم.



۷- الگوریتم بازگشتی طراحی کنید تا یک ماتریس $n \times n$ را دریافت کرده سپس دترمینان ماتریس را به عنوان خروجی برمی گرداند.

پاسخ :

چون هدف این مسئله ارایهٔ الگوریتم است، نه ارایهٔ برنامه، ما هم الگوریتم را در چند مرحله ارائه می دهیم، و جزئیات هر مرحله را بیان نمی کنیم. مراحل زیر را انجام دهید.

الف) ابتدا باید ماتریس A_{ij} را به این گونه تعریف کنیم که از حذف سطر i ام و و ستون jام از ماتریس A به دست آمده اند.

ب)

```
Void  det  (A[][], n)
{     if   n=۱   then   return   A[۱][۱]
      else   d=۰
             for   i=۱      to   n
                   d = d + (-۱)^(i+۱) * A[۱][i] * det (A_۱i , n-۱)
      return (d)
}
```

۸- یک درخت دودوئی را در نظر گرفته، تابع بازگشتی بنویسید که تعداد گره های غیر برگ درخت را بشمارد.

پاسخ :

مسئله را در سه مرحله حل می کنیم.

الف) طبق الگوریتم ۳-۲ در صفحهٔ ۴۷ کتاب، تعداد برگهای درخت را می شماریم و آن را با A نشان می دهیم.

ب) طبق الگوریتم ۴-۲ در صفحهٔ ۴۸ کتاب، تعداد گره های درخت را می شماریم و آن را با B نشان می دهیم.

ج) حاصل $B-A$ را حساب می کنیم.

۹- الگوریتم بازگشتی برای محاسبه تعداد گره های دو فرزندی یک درخت دودوئی بنویسید.

پاسخ :

```
int     k (Node * tree)
  {     if  (tree = Null) then  return ۰.
        else
        return  (k(tree → left) + k(tree → vight) + ۱)
  }
```

۱۰- صفحه ای به ابعاد $n * ۳$ موجود است. می خواهیم این صفحه ر با موزائیک های ۱*۲فرش کنیم. رابطه بازگشتی برای حل این مسئله ارائه دهید.

پاسخ :

اولاً n باید زوج باشد تا این کار مقدور باشد. ثانیاً نحوهٔ کاشی کاری به این طریق است که در سطر اول کاشیها را افقی قرار می دهیم و در دو سطر پایین کاشیها را به صورت عمودی قرار می‌دهیم. دیده می شود که با اضافه کردن دو ستون به n ستون (که باز هم تعداد ستونها زوج

می‌شود) به سه کاشی جدید نیازمندیم. پس اگر $T(n)$ نشان دهندهٔ تعداد کاشیهای مورد نیاز برای پر کردن صفحهٔ $n*۳$ باشد،داریم:

$$T(n+۲)=T(n)+۳$$

۱۱- صفحه ای به ابعاد $n*۲$ موجود است. می خواهیم این صفحه را با موزائیک های ۱*۲ فرش کنیم. به چند طریق می توان این کار را انجام داد (رابطه ای بازگشتی بیابید).

پاسخ :

فرض کنیم $T(n)$ نشان دهندهٔ تعداد حالات ممکن برای فرش کردن صفحهٔ $n*۲$ باشد. می‌خواهیم $T(n+۱)$ را بر حسب $T(n)$ بیابیم. برای فرش کردن صفحه ای به ابعاد $(n+۱)*۲$ دو امکان وجود دارد.

الف) ستون $(n+۱)$ ام را با یک کاشی عمودی فرش کنیم و مابقی خانه ها را به $T(n)$ روش فرش کنیم.

ب) دو ستون $(n+۱)$ ام و n ام را با دو کاشی افقی فرش کنیم و مابقی خانه ها را به $T(n-۱)$ روش فرش کنیم. پس داریم:

$$T(n)=T(n)+T(n-۱),\quad T(۱)=۱\quad T(۱)=۲$$

۱۲- روی محیط دایره ای چند کارت کوچک سیاه و سفید قرار داده ایم. دو نفر به نوبت این عمل را انجام می دهند. اولی همه کارتهای سیاهی که در مجاورت یک سفید باشند، برمی دارد و دومی روی کارتهای سفید مجاور سیاه همین کار را انجام می دهد. اگر ۴۰ کارت وجود داشته باشد، آیا ممکن است پس از انجام ۲ حرکت فقط یک کارت باقی بماند؟ اگر کارتها ۱۰۰۰ باشد، حداقل چند حرکت لازم است؟ اگر n کارت دور دایره باشد، حداقل چند حرکت لازم است؟

پاسخ :

صورت مسئله گنگ است. مشخص نیست که نفر اول آیا کارتهایی را بر می دارد که در مجاورت دقیقاً یک سفید قرار دارند یا کارتهایی را بر می دارد که در کنار حداقل یک سفید قرار دارند.

۱۳- مسئله برج های هانوی را در نظر گرفته، رابطه بازگشتی برای حل آن بنویسید.

پاسخ :

تعداد حرکات لازم برای جا به جایی n دیسک را با $T(n)$ نمایش می دهیم. برای جا به جایی $n+۱$ دیسک، ابتدا با $T(n)$ حرکت، n دیسک بالایی را به میلهٔ دوم برده، سپس با یک حرکت بزرگترین دیسک را به میلهٔ سوم برده، و نهایتاً با $T(n)$ حرکت تمام n دیسک میلهٔ دوم را به میلهٔ سوم منتقل می کنیم پس داریم:

$$T(n+۱)=T(n)+۱+T(n)=۲T(n)+۱\qquad ,T(۱)=۱$$

تمرینات فصل سوم

۱- زمان اجرای الگوریتم های زیر را محاسبه نمایید:

الف)

```
i = n  ;
while (i >= ۱) {
  /* Some Statement requiring θ(۱) time* /
     i = i / ۲ ;
}
```

پاسخ :

$$T(n) = T\left(\frac{n}{۲}\right) + ۱$$

با قرار دادن $n = ۲^k$, $T(۲^k) = a_k$ داریم :

$$a_k - a_{k-۱} = ۱$$

پس از تبدیل k به $k+۱$ و کم کردن از رابطه اصلی داریم:

$$a_{k+۱} - ۲a_k + a_{k-۱} = ۰ \Rightarrow x^۲ - ۲x + ۱ = ۰ \Rightarrow (x-۱)^۲ = ۰$$

$$\Rightarrow a_k = c_۱ + c_۲ k \qquad \Rightarrow T(n) = c_۱ + c_۲ \log_۲^n \in \theta(\log n)$$

ب)

```
i = ۱
while (i <= n ) {
/* Some Statement requiring θ (۱) time* /
     i = i * ۲ ;
}
```

پاسخ :

رابطهٔ بازگشتی $T(n) = T\left(\frac{n}{۲}\right) + ۱$ به دست می آید ، که با اعمال مراحل حل قسمت الف، به دست می آید ، $T(n) \in \theta(\log n)$

ج)

```
i = n ;
while (i >= ۱) {
      j = i ;
      while (j <= n) {
         /* Some Statement requiring θ (۱) time */
         j = j * ۲ ;
         }
         i = i / ۲ ;
}
```

پاسخ :

دیده می شود که

$$T(n) = ۱ + ۲ + ... + (\lfloor \log n \rfloor + ۱) = \frac{(\lfloor \log n \rfloor + ۱)(\lfloor \log n \rfloor + ۲)}{۲}$$

و در نتیجه:

$$T(n) \in \theta\left((\log n)^۲\right)$$

د)

```
/* Suppose that n > m */
  while (n > ۰)
  {
     r = n%m ;
     n = m ;
     m = r ;
}
```

پاسخ :

طبق توضیحات بیان شده در حل قسمت د از تمرین ۴ صفحهٔ ۳۳، داریم:

$$T(n) < ۲ + T\left\lfloor \frac{n}{۲} \right\rfloor$$

اگر $T(n)$ مساوی بود با $۲ + T\left\lfloor \frac{n}{۲} \right\rfloor$ ، آنگاه پس از انتخاب $n = ۲^k$ ، و تبدیل n به $n+۱$ ، و کم کردن از رابطهٔ اصلی، به دست می آمد $T(n) \in \theta(\log n)$. پس جواب این نامساوی به صورت $T(n) \in O(\log n)$ می باشد.

۲- در تابع زیر $List$ را یک آرایه n عنصری در نظر بگیرید:

```
int S (int List [ ] , int n)
{
    if (n == ۱)
        return (List[۱]);
     else
        return (List [n ]+ S(List , n−۱) ) ;
}
```

تابع زمانی و پیچیدگی زمانی تابع بالا را به روش حدس و استقراء محاسبه نمایید.

پاسخ :

از الگوریتم بیان شده، رابطهٔ بازگشتی زیر به دست می آید:

$$T(n) = T(n-۱)+۱$$

حدس می زنیم جواب به صورت $T(n) \in O(n)$ باشد. برای اثبات حدس خود از استقراء کمک میگیریم. پس باید یک $C>۰$ پیدا کنیم که $T(n) \le Cn$.

ادعا می کنیم برای $n \ge ۲$, $C \ge ۱$ رابطهٔ فوق برقرار است.

برای $n=۲$ داریم:

$$T(۲) = ۱ + T(۱) = ۲ \le C \times ۲$$

که رابطهٔ مطلوب است. فرض کنیم برای $n=k$ برقرار باشد، یعنی $T(k) \le Ck$. حال اگر $n=k+۱$ ، داریم:

$$T(k+۱) = ۱ + Tk \le Ck \le C + Ck = C(k+۱)$$

و حکم ثابت است.

۳- تابع $Func$ را به صورت زیر در نظر بگیرید:

```
int  Func (int n)
{
    if (n = = ۱)
        return (۱);
     else
        return (n + func (n−۱) ) ;
 }
```

تابع زمانی و پیچیدگی زمانی تابع بالا را به روش حدس و استقراء محاسبه نمایید.

پاسخ :

از الگوریتم بیان شده رابطهٔ بازگشتی زیر به دست می آید:

$$T(n) = n + T(n-1)$$

حدس می زنیم جواب آن به صورت $T(n) \in O(n^2)$ باشد. برای اثبات حدس خود از استقراء استفاده می کنیم. باید یک $C > 0$ و یک $M > 0$ پیدا کنیم که:

$$\forall n \geq M \Rightarrow T(n) \leq Cn^2$$

ادعا می کنیم برای $M = 2$ ، $C \geq 2$ رابطهٔ فوق برقرار است.

برای $n=2$ داریم:

$$T(2) = 2 + T(1) = 3 < 2 \times 2 \leq 2 \times C$$

فرض می کنیم برای $n=k$ حکم برقرار باشد، یعنی $T(k) \leq Ck^2$ اگر $n=k+1$ ، داریم:

$$T(k+1) = k+1+T(k) \leq k+1+Ck^2 \leq 2Ck + C + Ck^2 + C + Ck^2 = C(k+1)^2$$

۴- در تابع زیر $List$ یک آرایه n عنصری از اعداد صحیح می باشد:

```
int M (int List [ ], int n)
{
    if (n == 1)
        return (List [1]);
    else
        return (List [n] , M (List , n-1) ) ;
}
```

تابع زمانی و پیچیدگی زمانی تابع بالا را به روش تکرار با جایگذاری محاسبه نمایید.

پاسخ :

از الگوریتم بیان شده، رابطهٔ بازگشتی زیر به دست می آید:

$$T(n) = 1 + T(n-1) = 2 + T(n-2) = \ldots = n-1+T(1) = n-1+1 = n$$
$$\Rightarrow T(n) \in \theta(n)$$

۵- تابع *test* یک درخت دودوئی را دریافت می کند:

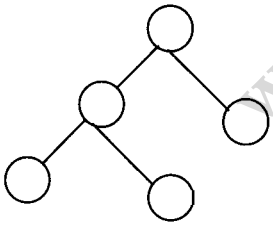
```
int test (Node * tree)
{
    if (tree == Null)
        return ۰;
    else
        return (۱ + max(test(tree → left) + test(tree → right)));
```

تابع زمانی و پیچیدگی زمانی تابع بالا را محاسبه نمایید.

پاسخ :

سطر آخر برنامه غلط چاپی دارد. باید مانند سطر آخر برنامهٔ تمرین ۴ از صفحهٔ ۵۱ تصحیح شود. رابطهٔ $T(n)=۱+۲T(\frac{n}{۲})$ را در نظر می گیریم و با توجه به قضیهٔ ۱-۳ (قسمت الف) داریم $T(n)\in\theta(n)$.

تذکر: در حالت کلی مشخص نیست در هر گره، تعداد گره های زیر درخت چپ با تعداد گره های زیر درخت راست چه نسبتی داشته باشند. رابطهٔ $T(n)=۱+۲T(\frac{n}{۲})$ برای هنگامی است که این تعداد دقیقاً مساوی باشند. پس می توان آن را بهترین حالت دانست. در بدترین حالت، هر گرهی شامل یک برگ است مثل درخت



در این حالت رابطه به صورت $T(n)=۲+T(n-۱)$ در می آید که باز هم در $\theta(n)$ است. پس حالت میانگین آن هم $\theta(n)$ می شود.

۶- پیچیدگی زمانی توابع زیر را محاسبه نمایید:

$$T(n)=۲T(n/۴)+d \quad if \quad n>۲$$
$$C \quad if \quad n=۲$$

پاسخ :

در قضیهٔ ۱-۳ قرار می دهیم $a=۲$ و $b=۴$ و $d=f(n)$ چون $\log_۴^۲=۱/۲$ پس قسمت الف قضیه به ازای $\varepsilon=۱/۲$ برقرار است. در نتیجه $T(n)\in\theta(n^{۱/۲})=\theta(\sqrt{n})$.

۷- تابع زیر:

$$T(n) = ۲T\left(\left\lfloor \frac{n}{۲} \right\rfloor + ۱۷\right) + n \ , \ n > ۱۰۰$$

$$T(۱) = T(۲) = ... = T(۱۰۰) = ۱$$

را در نظر گرفته آن را با روش حدس و استقراء حل نمایید (حدس می زنیم $T(n) \le CnLogn$).

پاسخ :

ادعا می کنیم برای $C \ge ۱$ حکم برقرار است. برای $n=۱۰۱$ که حکم به وضوح برقرار است. فرض می کنیم حکم برای n های کوچکتر و مساوی $k \ge ۱۰۱$ برقرار باشد.

$$۲T\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۷\right) + k \le Ck \log_{۲}^{k}$$

حال اگر $n=k+۱$ دو حالت در نظر می گیریم.

حالت اول: k زوج باشد در این حالت $\left\lfloor \frac{k+۱}{۲} \right\rfloor = \frac{k}{۲}$ و در نتیجه:

$$T(k+۱) = ۲T\left(\left\lfloor \frac{k+۱}{۲} \right\rfloor + ۱۷\right) + k + ۱ = ۲T\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۷\right) + k + ۱$$

چون $k > \frac{k}{۲} + ۱۷$ پس عبارت فوق از عبارت زیر کوچکتر یا مساوی است.

$$\le Ck \log^{k} + ۱ \le Ck \log^{k} + C \log^{k} =$$

$$C(k+۱) \log^{k} \le c(k+۱) \log^{(k+۱)}$$

حالت دوم: k فرد است در این حالت $\left\lfloor \frac{k+۱}{۲} \right\rfloor = \left\lfloor \frac{k}{۲} \right\rfloor + ۱$ پس

$$T(k+۱) = ۲T\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۸\right) + k + ۱$$

چون $k > \left\lfloor \frac{k}{۲} \right\rfloor + ۱۸$ پس عبارت فوق از عبارت زیر کوچکتر یا مساوی است:

$$\le ۲C\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۸\right) \log\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۸\right) + k + ۱$$

چون k فرد است، پس $۲\left\lfloor \frac{k}{۲} \right\rfloor = k - ۱$ پس عبارت فوق برابر است با

$$C(k + ۳۵) \log\left(\left\lfloor \frac{k}{۲} \right\rfloor + ۱۸\right) + k + ۱$$

با انتخاب k های به اندازهٔ کافی بزرگ، حاصل فوق از $C(k+1)\log(k+1)$ کوچکتر یا مساوی است. (چون به ازای کاهای بزرگ k+۱ خیلی از $\lfloor \frac{k}{۲} \rfloor$+۱۸ بزرگتر می شود و کمبود مقدار پرانتز اول را جبران می کند.)

پس در حالت کلی داریم:

$$T(n) \in O(n \log n)$$

توجه: این مسئله راههای دیگری هم داریم. ولی ترجیح داریم حداقل یک مسئله را این گونه حل کنیم.

۸- رابطه بازگشتی زیر را به روش حدس و استقراء حل کنید:

$$T(n) = \begin{cases} ۰ & if \quad n = ۱ \\ ۱ & if \quad n = ۲ \\ T\left(\left\lfloor \frac{n}{۲} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{۲} \right\rfloor\right) + ۲ & if \quad n > ۲ \end{cases}$$

پاسخ :

حدس می زنیم $T(n) \in O(n \log n)$ به منظور اثبات این رابطه، با استقراء ثابت می کنیم $T(n) \le ۲n - ۱$

برای n=۱ رابطهٔ $T(۱) \le ۱$ به وضوح برقرار است.فرض کنیم برای هر $k < n$ حکم برقرار باشد،یعنی

$$\forall k < n \Rightarrow T(k) \le ۲k - ۱$$

حال فرض می کنیم n=k چون $\lfloor \frac{k}{۲} \rfloor < n$, $\lceil \frac{k}{۲} \rceil < n$ داریم:

$$T(k) = T\left(\left\lfloor \frac{k}{۲} \right\rfloor\right) + T\left(\left\lceil \frac{k}{۲} \right\rceil\right) + ۲ \le ۲\left\lceil \frac{k}{۲} \right\rceil - ۱ + ۲\left\lfloor \frac{k}{۲} \right\rfloor - ۱ + ۲$$
$$< ۲(k+۱) - ۱$$

۹- روابط بازگشتی زیر را حل کنید:

الف)

$$T(n) = \begin{cases} ۱ & if \quad n = ۴ \\ T\left(\sqrt{n}\right) + c & if \quad n > ۴ \end{cases}$$

پاسخ :

با توجه به ضابطهٔ رابطه، n باید به صورت ۴^k باشد که $k = ۲^m$ پس $n = ۴^{۲^m}$.

در نتیجه $\sqrt{n} = ۴^{۲^{m-۱}}$ پس $T(۴^{۲^m}) - T(۴^{۲^{m-۱}})$ با قرار دادن $T\left(۴^{۲^m}\right) = a_m$ داریم:

$$a_m - a_{m-۱} = c \Rightarrow a_m \in \theta(m)$$

چون $\log_۴^n = ۲^n$ پس $m = \log_۲ \log_۴^n$ و نهایتاً $T(n) \in \theta(\log_۲ \log_۴^n)$.

تذکر: می توان از راه حل مثال ۳-۳ نیز استفاده کرد.

ب)

$$T(n) = \begin{cases} ۱ & if \quad n = ۴ \\ ۲T(n/۴) + cn & if \quad n > ۴ \end{cases}$$

در قضیه ۱-۳ قرار می دهیم $a = ۲$, $b = ۴$, $f(n) = cn$ چون $\log_۴^۲ = \frac{۱}{۲}$ پس قسمت ج

از قضیه به ازای $\varepsilon < \frac{۱}{۲}$ برقرار است. پس $T(n) \in \theta(n)$.

البته به صورت مستقیم هم می توان چنین عمل کرد.

از ضابطهٔ رابطهٔ بازگشتی مشخص است که $n = ۴^k$ پس $T(۴^k) - ۲T(۴^{k-۱}) = c۴^k$ با قرار دادن $T(۴^k) = a_k$ داریم:

$$a_k - ۲a_{k-۱} = c۴^k$$

یکبار دو طرف را در ۴ ضرب کرده و یکبار هم kها را به $k+۱$ تبدیل می کنیم. پس داریم:

$$\begin{cases} a_{k+۱} - ۲a_k = c۴^{k+۱} \\ ۴a_k - ۸a_{k-۱} = c۴^{k+۱} \end{cases} \Rightarrow a_{k+۱} - ۶a_k + ۸a_{k-۱} = ۰$$

$$\Rightarrow x^۲ - ۶x + ۸ = ۰ \Rightarrow x = ۲ \ , \ x = ۴$$

$$\Rightarrow a_k = c_۱۲^k + c_۲۴^k \Rightarrow$$

$$T(n) = c_۱۲^{\log_۴^n} + c_۲n = c_۱n^{۱/۲} + c_۲n \in \theta(n)$$

ج)

$$T(n) = \begin{cases} d & if \quad n = ۴ \\ ۷T\ (n-۴) + cn & if \quad n > ۴ \end{cases}$$

پاسخ :

$$T(n) - ۷T(n-۴) = cn$$

با توجه به صورت مسئله n باید به شکل $۴k$ باشد. یعنی $n = ۴k$ پس

$$T(۴k) - ۷T\left(۴(k-۱)\right) = ۴ck$$

$T(۴k)$ را با a_k نمایش می دهیم، پس

$$a_k - a_{k-۱} = ۴ck$$

با تبدیل k به $k+۱$ و کم کردن از رابطهٔ فوق، به دست می آید:

$$a_{k+۱} - ۸a_k + ۷a_{k-۱} = ۴c$$

یکبار دیگر در رابطهٔ اخیر k را به $k+۱$ تبدیل کرده و از رابطهٔ اخیر کم می کنیم. پس

$$a_{k+۱} - ۹a_{k+۱} + ۱۵a_k - ۷a_{k-۱} = ۰ \Rightarrow$$

$$x^۳ - ۹x^۲ + ۱۵x - ۷ = ۰ \Rightarrow (x-۱)^۲(x-۷) = ۰ \Rightarrow x = ۱ \,,\, x = ۱ \; x = ۷$$

$$\Rightarrow a_k = c_۱ + c_۲k + c_۳۷^k \Rightarrow$$

$$T(n) = c_۱ + c_۲\frac{n}{۴} + c_۳۷^{n/۴} \in \theta\left(\sqrt[۴]{۷^n}\right)$$

د)

$$T(n) = \begin{cases} d & if \quad n = ۲ \\ ۵T(n/۲) + n^۲ & if \quad n > ۲ \end{cases}$$

در قضیهٔ ۱-۳ قرار می دهیم $a = ۵$, $b = ۲$, $F(n) = n^۲$

چون $\log_۲^۵ > ۲$ پس حالت الف از قضیه اتفاق می افتد. پس

$$T(n) \in \theta\left(n^{\log_۲^۵}\right) \approx \theta\left(n^{۲/۳}\right)$$

۱۰- روابط بازگشتی زیر را حل کنید:

الف)

$$T(n) = \begin{cases} T(n-۱) + T(n-۲) & if \quad n > ۲ \\ T(۰) = ۰ \;\;,\;\; T(۱) = ۱ \end{cases}$$

پاسخ :

$$T(n) = T(n-۱) + T(n-۲)$$

با انتخاب $T(n) = a_n$ داریم:

$$a_n - a_{n-۱} - a_{n-۴} = ۰ \Rightarrow x^۲ - x - ۱ = ۰ \Rightarrow x = \frac{۱+\sqrt{۵}}{۲} \quad , \quad x = \frac{۱-\sqrt{۵}}{۲}$$

$$\Rightarrow a_n = c_۱\left(۱ + \frac{\sqrt{۵}}{۲}\right)^۲ + c_۲\left(\frac{۱-\sqrt{۵}}{۲}\right)^n \in \theta\left(\frac{۱+\sqrt{۵}}{۲}\right)^n$$

ب)

$$T(n)\begin{cases}۲T(n-۱)-۳T(n-۲) & if \quad n>۲\\ T(۰)=۱\ ,\ T(۱)=۲\end{cases}$$

پاسخ :

$T(n)$ را با a_n نشان می دهیم:

$$a_n - ۲a_{n-۱} + ۳a_{n-۲} = ۰$$

چون این معادله ریشهٔ حقیقی ندارد، با اطلاعات این کتاب نمی توان این رابطه را حل کرد.

ج)

$$T(n)=\begin{cases}۴T(n-۱)+۳T(n-۲) & if \quad n>۲\\ T(۰)=۲\ ,\ T(۱)=۱\end{cases}$$

پاسخ :

$T(n)$ را با a_n نشان می دهیم.

$$a_n - ۴a_{n-۱} - ۳a_{n-۲} = ۰ \Rightarrow$$

$$x^۲ - ۴x - ۳ = ۰ \qquad x = ۲+\sqrt{۷} \quad , \quad x = ۲-\sqrt{۷}$$

$$\Rightarrow a_n = c_۱(۲+\sqrt{۷})^n + c_۲(۲-\sqrt{۷})^n \in \theta\left((۲+\sqrt{۷})^n\right)$$

د)

$$T(n)=\begin{cases}۷T(n-۱)-۴T(n-۲) & if \quad n>۲\\ T(۰)=۰\ ,\ T(۱)=۲\end{cases}$$

پاسخ :

$T(n)$ را با a_n نشان می دهیم.

$$a_n - ۷a_{n-۱} + ۴a_{n-۲} = ۰ \Rightarrow$$

$$x^۲ - ۷x + ۴ = ۰ \Rightarrow x = \frac{۷+\sqrt{۳۳}}{۲} \ , \ x = \frac{۷-\sqrt{۳۳}}{۲}$$

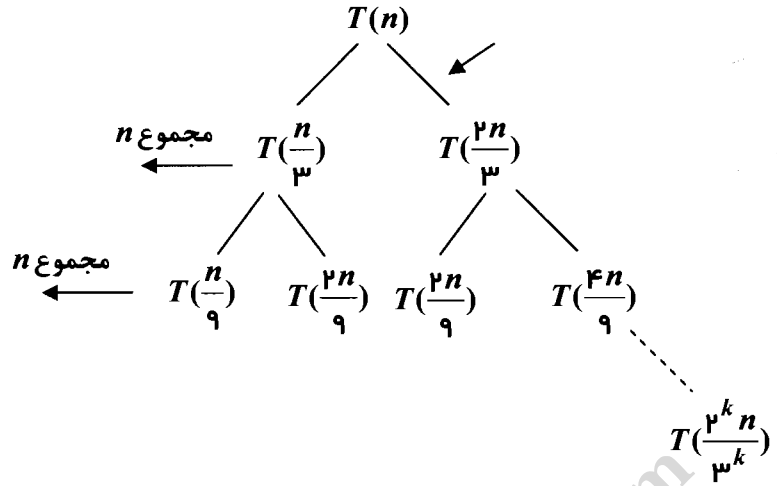
$$\Rightarrow a_n = c_۱\left(\frac{۷+\sqrt{۳۳}}{۲}\right)^n + c_۲\left(\frac{۷-\sqrt{۳۳}}{۲}\right)^n \in \theta\left((\frac{۷+\sqrt{۳۳}}{۲})^n\right)$$

۱۱- روابط بازگشتی زیر را حل کنید:

الف)

$$T(n)=\begin{cases}T(n/۳)+T(۲n/۳)+n & if \quad n>۳\\ T(۰)=۰\ ,\ T(۱)=۲\end{cases}$$

پاسخ :



شاخهٔ سمت راست درخت (که با فلش نشان داده شده است) از بقیهٔ شاخه ها دیرتر اجرا میشود، پس آن را حساب می کنیم. چون باید به $T(۱)$ برسد پس

$$\left(\frac{۲}{۳}\right)^k \times n = ۱ \Rightarrow \left(\frac{۲}{۳}\right)^k = \frac{۱}{n} \Rightarrow n = \left(\frac{۳}{۲}\right)^k \Rightarrow k = \log_{\frac{۳}{۲}}^{n}$$

چون k تا سطر موجود است، پس داریم:

$$T(n) \le nk = n\log_{\frac{۳}{۲}}^{n} \Rightarrow T(n) \in O(n\log n)$$

ب)

$$T(n) = \begin{cases} ۷T(n-۱) - ۴T(n-۲) + n & if \quad n > ۲ \\ T(۰) = ۰ \ , \ \ T(۱) = ۲ \end{cases}$$

پاسخ :

n را به ازای $n+۱$ تبدیل کرده و صورت مسئله را از آن کم می کنیم. به دست می آید

$$a_{n+۱} - ۸a_n + ۱۱a_{n-۱} - ۴a_{n-۲} = ۱$$

در عبارت فوق n را به $n+۱$ تبدیل کرده و حاصل را از خود عبارت کم می کنیم، پس:

$$a_{n+۲} - ۹a_{n+۱} + ۱۹a_n - ۱۵a_{n-۱} + ۴a_{n-۲} = ۰ \Rightarrow$$

$$x^۴ - ۹x^۳ + ۱۹x^۳ - ۱۵x + ۴ = ۰ \Rightarrow (x-۱)^۲\left(x^۲ - ۷x + ۴\right) = ۰ \Rightarrow$$

$$a_n = c_۱ + c_۲n + c_۳\left(\frac{۷ + \sqrt{۳۳}}{۲}\right)^n + c_۴\left(\frac{۷ - \sqrt{۳۳}}{۲}\right)^n \in \theta\left((\frac{۷ + \sqrt{۳۳}}{۲})^n\right)$$

ج)

$$T(n)=\begin{cases}T(n-۱)+T(n-۲)+۴n & if \quad n>۲\\ T(۰)=۱ \quad , \quad T(۱)=۱\end{cases}$$

پاسخ :

n ها را به $n+۱$ تبدیل کرده و از صورت مسئله کم می کنیم. پس

$$a_{n+۱}-۲a_n+a_{n-۲}=۴$$

در رابطهٔ اخیر هم n را به $n+۱$ تبدیل کرده و از رابطه می کنیم.

به دست می آید:

$$a_{n+۲}-۳a_{n+۱}+۲a_n+a_{n-۱}-a_{n-۲}=۰ \Rightarrow$$

$$x^۴-۳x^۳+۲x^۲+x-۱=۰ \Rightarrow (x-۱)^۲\left(x^۲-x-۱\right)=۰ \Rightarrow$$

$$T(n)=c_۱+c_۱n+c_۳\left(\frac{۱-\sqrt{۵}}{۲}\right)^n+c_۴\left(\frac{۱+\sqrt{۵}}{۲}\right)^n \Rightarrow$$

$$T(n)\in\theta\left((\frac{۱+\sqrt{۵}}{۲})^n\right)$$

د)

$$T(n)=\begin{cases}۲T(n/۲)-۴T(n/۴)+n^۲ & if \quad n>۴\\ T(۰)=۰ \quad , \quad T(۱)=۱\end{cases}$$

پاسخ :

n را برابر ۲^k می گیریم. پس

$$T(۲^k)-۲T(۲^{k-۱})+۴T(۲^{k-۲})=۲^{۲k}$$

$T(۲^k)$ را a_k می گیریم. پس

$$a_k-۲a_{k-۱}+۴a_{k-۲}=۲^{۲k}$$

یکبار k را به $k+۱$ تبدیل کرده و یکبار در ۴ ضرب می کنیم. پس

$$\begin{cases}a_{k+۱}-۲a_k+۴a_{k-۱}=۲^{۲k+۲}\\ ۴a_k-۸a_{k-۱}+۱۶a_{k-۲}=۲^{۲k+۲}\end{cases} \Rightarrow a_{k+۱}-۶a_k+۱۲a_{k-۱}-۱۶a_{k-۲}=۰$$

$$\Rightarrow x^۳-۶x^۲+۱۲x-۱۶=۰ \Rightarrow (x-۲)(x^۲-۲x+۴)=۰$$

$$\Rightarrow x=۴ \Rightarrow a_k\in\theta(۴^k) \Rightarrow$$

$$T(n)\in\theta(۴^{\log_۲^n})=\theta(n^۲)$$

تمرینات فصل چهارم

۱- نسخه بازگشتی و غیر بازگشتی جستجوی دودوئی را اجرا نموده و زمانهای به دست آمده را با هم مقایسه کنید.

پاسخ :

این یک مسئلهٔ عملی است و باید روی کامپیوتر اجرا شود.

۲- پیچیدگی زمانی در بدترین حالت را برای جستجوی دودوئی محاسبه نمایید.

پاسخ :

در جستجوی دودویی، بدترین حالت زمانی اتفاق می افتد که عنصر مورد جستجو یا در آخرین دور از حلقهٔ تکرار پیدا شود، و یا اصلاً در آرایه وجود نداشته باشد. اگر m برابر با تعداد دفعاتی باشد که باید آرایه نصف شود، یا به عبارت دیگر m برابر با تعداد دفعات مقایسه برای جستجوی عنصر مورد نظر باشد، و اگر n برابر با تعداد اعضای آرایه باشد، خواهیم داشت:

$$۲^m = n \Rightarrow m = \log_۲^n \in O(\log n).$$

اگر از مدل روابط بازگشتی بخواهیم استفاده کنیم، جستجوی دودویی به رابطهٔ زیر می رسد:

$$T(n) = \begin{cases} ۱ & \text{اگر } n = ۱ \\ ۱ + T\left(\frac{n}{۲}\right) & \text{اگر } n > ۱ \end{cases}$$

برای حل این روابط بایدn را برابر ۲^m گرفت، و پس از حل رابطهٔ بازگشتی با استفاده از روشهای عنوان شده در فصل قبل، بدست می آید: $T(n) \in O(\log n)$

تذکر: اگر n فرد باشد، رابطهٔ بازگشتی به صورت $T(n) = ۱ + T\left(\left\lfloor \frac{n}{۲} \right\rfloor\right)$ در می آید، که پس از حل، به جواب $T(n) \in O(\log n)$ می رسیم.

۳- در یک آرایه نا محدود، n محل اول حاوی اعداد صحیح مرتب شده می باشد بقیه محل ها با ∞ پر شده است. الگوریتمی ارائه دهید که X را به عنوان ورودی دریافت کرده و موقعیت X را در آرایه با زمان $\theta(Log\ n)$ به دست آورد. (n مشخص نیست)

پاسخ :

برای فهم بهتر، الگوریتم را مرحله به مرحله، به صورت فارسی بیان می کنیم.

۱) از ابتدای آرایهٔ داده شده، زیر آرایه ای به طول یک را در نظر می گیریم و آنرا $A[\ ]$ می‌نامیم.

۲) طول زیر آرایهٔ $A[\ ]$ را دو برابر می کنیم.

۳) مرحلهٔ ۲ را تا هنگامی ادامه می دهیم که آخرین عضو $A[\ ]$ برابر ∞ شود.

۴) روی آرایهٔ $A[\ ]$ که نهایتاً از ۳ به دست آمده، الگوریتم جستجوی دودویی را انجام می‌دهیم.

برای تحلیل الگوریتم فوق دیده می شود، مراحل ۱ تا۳ حداکثر آرایه ای با طول $2n$ تولید می‌کنند و زمان اجرای آنها $\theta\left(\log_{2}^{2n}\right)$ است. چون مرحلهٔ ۴ در طول سه مرحلهٔ قبل قرار دارد و زمان آن نیز از مرتبهٔ $\theta\left(\log_{2}^{2n}\right)$است، پس زمان کل الگوریتم برابر است با:

$$\theta\left(\log_{2}^{2n}\right)=\theta\left(\log_{2}^{2}+\log_{2}^{n}\right)=\theta(\log n)$$

۴- الگوریتمی بنویسید که لیست مرتب شده ای از n عنصر را با تقسیم آن به سه لیست کوچکتر، هر یک با حدود $n/3$ عنصر جستجو کند. این الگوریتم را تحلیل نموده و نتایج آن را ارائه دهیم.

پاسخ :

الگوریتم بازگشتی آن را می نویسیم. می خواهیم داخل آرایهٔ n عنصری $a[\ ]$،xرا به روش گفته شده در صورت مسئله، جستجو کنیم.

L وH را حدود بالا و پایین آرایه $a[\ ]$ می گیریم. الگوریتم را $search\ (L, H, a, x)$ می نامیم.

فرض می کنیم آرایه از $L=0$ تا $H=n-1$ اندیس گذاری شده است.

```
Void      search (L, H, a[ ], x)
{
 if     L > H
  then    p ← 0
  else  { r ← (L + H) / 3
          S ← 2 * (L + H) / 3
          if   x = a[r] ,      then   return (r).
          if   x = a[s] ,      then   return (s).
          if   x < a[r] ,      then   return   search(L, r − 1, a, x).
          if   a[r] < x < a[s] , then   return   search(r + 1, s − 1, a, x).
          if   x > a[s] ,      then   return   search(s + 1, H, a, x).
        }
}
```

برای تحلیل الگوریتم فوق، رابطهٔ بازگشتی $T(n)=T\left(\frac{n}{۳}\right)+۲$ را باید حل کرد، زیرا با دو مقایسه (مقایسه با $a[r]$ و $a[s]$)، به آرایه ای به طول $\frac{n}{۳}$ می رسیم.

برای حل این رابطه بازگشتی، باید فرض کرد $n=۳^k$. پس اولاً $k=\log_۳^n$، ثانیاً
$T(۳^k)-T(۳^{k-۱})=۲$

با انتخاب $T(۳^k)=a_k$ به رابطهٔ $a_k - a_{k-۱}=۲$ می رسیم. k را به $k+۱$ تبدیل کرده و از رابطه اصلی کم می‌کنیم. پس:

$$a_{k+۱}-۲a_k+a_{k-۱}=۰ \Rightarrow x^۲-۲x+۱=۰ \Rightarrow x=۱,\ x=۱ \Rightarrow$$
$$a_k=c_۱+c_۲k \Rightarrow T(n)=c_۱+c_۲\log_۳^n \in \theta\left(\log_۳^n\right).$$

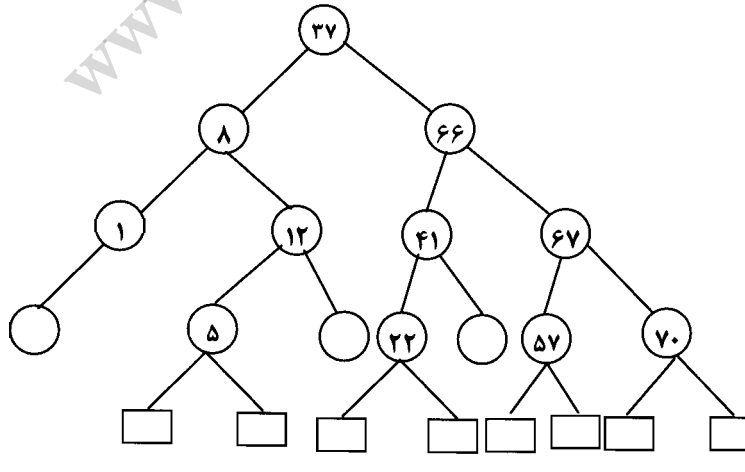
پس تعداد مقایسه های این الگوریتم از جستجوی دودویی کمتر است، ولی چون $\theta\left(\log_۲^n\right)=\theta\left(\log_۳^n\right)$ پس از نظر پیچیدگی زمانی با جستجوی دودویی یکسان است.

۵- اعداد زیر را در نظر بگیرید:

۱ ۵ ۸ ۱۲ ۲۲ ۳۷ ۴۱ ۵۷ ۶۶ ۶۷ ۷۰

درخت فراخوانی بازگشتی را برای لیست بالا در الگوریتم جستجوی دودوئی ترسیم نمایید.

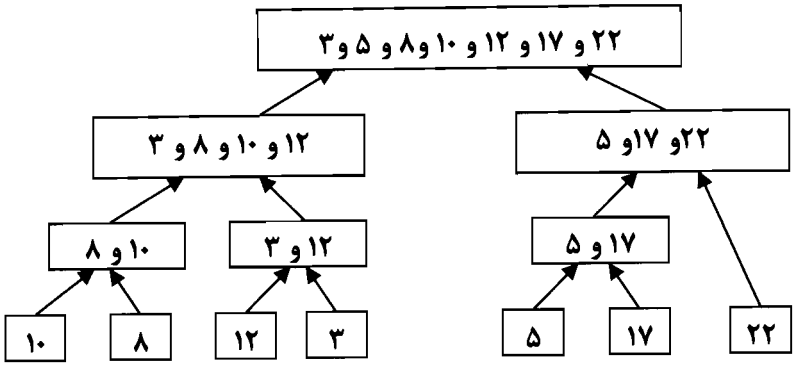
پاسخ :



۶- اعداد زیر را با روش مرتب سازی ادغامی مرتب نموده و درخت فراخوانی بازگشتی آن را ترسیم نمایید.

۱۰ ۸ ۱۲ ۳ ۵ ۱۷ ۲۲

پاسخ :



۷- زمان بدترین حالت و بهترین حالت مرتب سازی ادغامی را تحلیل نمایید.

پاسخ :

در بهترین حالت یکی از دنباله ها، به انتهای دنبالهٔ دیگر می چسبد که در این صورت تعداد مقایسات از رابطهٔ بازگشتی

$$T(n)=\begin{cases}0 & n=1\\ 2T\left(\frac{n}{2}\right)+\frac{n}{2} & n>1\end{cases}$$

به دست می آید.

با انتخاب $n=2^k$ و قرار دادن $T\left(2^k\right)=a_k$ به دست می آید.

$$a_k-2a_{k-1}=2^{k-1}$$

با ضرب طرفین ۲ و تبدیل k به $k-1$ داریم:

$$\begin{cases}2a_k-4a_{k-1}=2^k\\ a_{k+1}-2a_k=2^k\end{cases}$$

با کم کردن روابط فوق از هم داریم:

$$a_{k+1}-4a_k+4a_{k-1}=0\Rightarrow x^2-4x+4=0\Rightarrow x=2\ ,\ x=2$$

$$\Rightarrow a_k=c_1 2^k+c_2 k 2^k\Rightarrow T(n)=c_1 n+c_2 n\log_2^n$$

چون آرایهٔ یک عضوی نیازی به مقایسه ندارد و آرایهٔ دو عنصری با یک مقایسه مرتب می‌شود پس

$$T(2)=1\ ,\ T(1)=0$$

پس داریم:

$$\begin{cases} c_1 = 0 \\ c_1 + 2c_2 = 1 \end{cases} \Rightarrow c_1 = 0 \ , \ c_2 = \frac{1}{2}$$

$$\Rightarrow T(n) = \frac{1}{2} n \log_2^n$$

در بدترین حالت، اعضای دو آرایه به صورت یک در میان قرار می گیرند که تعداد مقایسه ها از رابطهٔ زیر محاسبه می شود.

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n - 1 & n > 1 \end{cases}$$

با انتخاب $n = 2^k$, $T(2^k) = a_k$ به دست می آید.

$$a_k - 2a_{k-1} = 2^k - 1$$

یک بار k را به $k+1$ تبدیل کرده و یکبار دو طرف رابطه را در ۴ ضرب می کنیم.

$$\begin{cases} a_{k+1} - 2a_k = 2^{k+1} - 1 \\ 2a_k - 4a_{k-1} = 2^{k+1} - 2 \end{cases} \Rightarrow a_{k+1} - 4a_k + 4a_{k-1} = 1$$

در رابطهٔ فوق k را به $k+1$ تبدیل کرده و حاصل را از رابطهٔ فوق کم می کنیم به دست می آید:

$$a_{k+2} - 5a_{k+1} + 8a_k - 4a_{k-1} = 0 \Rightarrow$$

$$x^3 - 5x^2 + 8x - 4 = 0 \Rightarrow x^3 - 5x^2 + 8x - 4 = 0 \Rightarrow$$

$$(x-1)(x-2)^2 = 0 \Rightarrow x = 1 \ , \ x = 2 \ , \ x = 2$$

$$\Rightarrow a_k = c_1 + c_2 2^k + c_3 k_2^k \Rightarrow$$

$$T(n) = c_1 + c_2 n + c_3 n \log_2^n$$

با قرار دادن $T(1)=0$ و $T(2)=1$ و $T(4)=5$ در رابطهٔ فوق داریم:

$$\begin{cases} c_1 + c_2 = 0 \\ c_1 + 2c_2 + 2c_3 = 1 \\ c_1 + 4c_2 + 8c_3 = 5 \end{cases}$$

$$\Rightarrow c_1 = 1 \quad , \ c_2 = -1 \ , \ c_3 = 1$$

$$\Rightarrow T(n) = n \log_2^n - n + 1$$

۸- یک روش مرتب سازی پایدار گفته می شود اگر در پایان روش، عناصر یکسان به همان ترتیبی که در لیست نامرتب اصلی بودند، قرار بگیرند. در این صورت آیا مرتب سازی ادغامی یک روش مرتب سازی پایدار است؟

پاسخ :

خیر. مثال ۲ ،۲. ۳ ،۲. مطلب را روشن می کند.

۹- فرض کنید ۴ لیست مرتب L_1 , L_2 , L_3 , L_4 موجود هستند، الگوریتمی ارائه دهید که، این چهار لیست را در هم ادغام نماید. زمان آن را تحلیل نمایید.

پاسخ :

مراحل زیر را به ترتیب انجام می دهیم.

۱- تابع زیر را برای ادغام دو آرایهٔ a و b که به ترتیب p و q عضو دارند می‌نویسیم:

```
Void marge(a[],b[],c[],p,q)
{
    i ← 0
    j ← 0
    k ← 0
    while(i < q and j < p)
        if(a[i]<= b[j])
        {    c[k]= a[i]
             ++i
             ++k
        }
        else {c[k]= b[j]}
             ++ j
             ++k
             }
        /* end of while* /
  if(i >= q)
    for(;j < p;++j)
        {c[k]= b[j]
             ++k
```

```
        }
    else
        for(;j<q,++i)
            {c[k]=a[i]
             ++k
            }
}
```

۲- آرایه های L_1 و L_2 را طبق الگوریتم بیان شده در ۱ با هم ادغام می کنیم، و حاصل را L_5 می‌نامیم.

۳- آرایه های L_3 و L_4 را طبق الگوریتم بیان شده در ۱ با هم ادغام می کنیم ، و حاصل را L_6 می‌نامیم.

۴- آرایه های L_5 و L_6 را طبق الگوریتم بیان شده در ۱ با هم ادغام می کنیم.

زمان اجرای این الگوریتم برابر $\theta(m)$است، که m برابر است با مجموع تعداد اعضای هر چهار آریه.

۱۰- یک لیست مرتب به نام A موجود است و لیست دیگری به نام B را در نظر بگیرید که نامرتب است. الگوریتمی برای ادغام لیست B در A (بافرض اینکه A ظرفیت لازم را داشته باشد) ارائه دهید.

پاسخ :

مراحل زیر را به ترتیب انجام می دهیم.

۱- B را طبق یکی از الگوریتم های مرتب سازی ادغامی، یا مرتب سازی سریع، مرتب می‌کنیم و حاصل را C می نامیم.

۲- Cو A را طبق الگوریتم بیان شده در قسمت اول تمرین۹ با هم ادغام کنیم.

۱۱- دنباله ای از لیست مرتب $L_1, L_2, ..., L_k$ موجود هستند به طوری که $\sum_{i=1}^{K} |Li| = n$، نشان دهید چگونه می توان این k دنباله را در زمان $O(n \, Log \, k)$ مرتب کنیم.

پاسخ :

ابتدا آرایه های L_1 و L_2 سپس آرایه های L_3 و L_4، و به همین ترتیب تا آرایهٔ آخر را طبق الگوریتم بیان شده در قسمت ۱ تمرین ۹ با هم ادغام می کنیم. برای این کار به تعداد $\theta(n)$ تا مقایسه لازم است و به $\left\lceil \frac{k}{2} \right\rceil$ تا آرایهٔ مرتب شده جدید می رسیم. برای این آریه ها نیز، به طور مشابه، دو تا دو تا، الگوریتم ادغام را انجام می دهیم و پس از تا مقایسه به تعداد

$\left\lceil \frac{\left\lfloor \frac{k}{۲} \right\rfloor}{۲} \right\rceil$ تا آرایهٔ مرتب شده می رسیم که این تعداد برابر است با $\left\lceil \frac{k}{۲^۲} \right\rceil$. بـه همـین طریـق ادامه می دهیم تا به یک آرایه برسیم که اگر آن مرحله را i بنامیم داریم:

$$\left\lceil \frac{k}{۲^i} \right\rceil = ۱ \Rightarrow i = \lfloor \log_۲^k \rfloor$$

چون در هر مرحله، $\theta(n)$ تا مقایسه انجام می شود و $\lfloor \log_۲^k \rfloor$ تا مرحله موجـود اسـت پـس زمان کل برابر است با $\theta(n \log k)$

۱۲- الگوریتمی بنویسید که لیستی از n عنصر را به سه زیر لیست هر یک با حدود $n/۳$ عنـصر، تقـسیم نماید هر یک از زیر لیست ها را به روش مرتب سازی ادغامی، مرتب کنیـد. الگـوریتم را تحلیـل نمـوده، نتایج آن را ارائه دهید.

پاسخ :

مراحل زیر را به ترتیب انجام میدهیم:

۱- قرار می دهیم :

$$B \leftarrow \left\lceil \frac{۲n}{۳} \right\rceil \quad , \quad A \leftarrow \left\lceil \frac{n}{۳} \right\rceil$$

۲- آرایهٔ $a[۱],\ldots a[A]$ را طبق مرتب سازی ادغامی، مرتب می کنیم و $L_۱$ مینامیم.

۳- آرایهٔ $a[A+۱],\ldots a[B]$ را طبق مرتب سازی ادغامی، مرتب می کنیم و $L_۲$ مینامیم.

۴- آرایهٔ $a[B+۱],\ldots a[n]$ را طبق مرتب سازی ادغامی، مرتب می کنیم و $L_۳$ مینامیم.

۵- آرایهٔ $L_۱, L_۲, L_۳$ را طبق تمرین ۱۱ با هم ادغام می کنیم.

در تحلیل این الگوریتم، دیده می شود که هر یک از مراحـل ۴،۳،۲ از مرتبـهٔ $O\left(\frac{n}{۳}\log\frac{n}{۳}\right)$ می‌باشند. چون

$$O\left(\frac{n}{۳}\log\frac{n}{۳}\right) = O\left(\frac{۱}{۳}(n\log n - n\log ۳)\right) = O(n\log n)$$

پس در مجموع مراحل ۱تا ۴ از مرتبهٔ $O(n\log ۳)$ می باشند. مرحلهٔ ۵ هم که طبق تمرین ۱۱ از مرتبهٔ $O(n\log ۳)$است که برابر اسـت بـا $O(n)$. پـس نهایتـاً، کـل الگـوریتم از مرتبـهٔ $O(n\log n)$ است.

۱۳- الگوریتم اصلی مرتب سازی سریع را که *Hoare* ارائه داده را بنویسید و تحلیل نمایید.

پاسخ :

حدود پایین و بالای آرایهٔ a را با L و H نشان می دهیم.

$$Void \quad quick-sort\,(a[\,],\,L,H)$$

{

$$i \leftarrow L\,,\; j \leftarrow H \quad ,\; x \leftarrow \frac{L+H}{۲}$$

$$while\,(i <= j)$$

$$\{while\,(a[i] < x \;and\; i < H) ++i$$

$$while\,(a[j] > x \;\; and \;\; j > L) --j$$

$$if\,(i <= j)$$

$$\{ \quad t \leftarrow a[i]\;,\; a[i] \leftarrow a[j]\;,\; a[j] \leftarrow t$$

$$i++\,,\; j--$$

}

}

$$if\,(j > L)\; then\; quick-sort\,(a\;,\;L\;,\;j)$$

$$if\,(i < H)\; then\; quick-sort\,(a,i,H)$$

}

۱۴- اعداد زیر را در نظر گرفته، با روش مرتب سازی سریع آنها را مرتب نموده، و درخت فراخوانی بازگشتی برای آن ترسیم نمایید.

۱۲ ۱ ۲۵ ۳ ۲۸ ۴۷ ۱۰ ۸ ۵۲

پاسخ :

هر مرحله از اجرای الگوریتم را در یک خط می نویسیم. در هر مرحله، عناصری که در جای قطعی خود قرار گرفته اند را علامتگذاری می کنیم.

۱۲ , ۱ , ۲۵,۳ , ۲۸ , ۴۷, ۱۰ , ۸ , ۵۲

۸ , ۱ , ۳ , ۱۰, ۱۲ , ۲۵ , ۲۸ , ۴۷ , ۵۲

۳ , ۱ , ۸ , ۱۰ , ۱۲ , ۲۵ , ۲۸ , ۴۷ , ۵۲

۱ , ۳ , ۸ , ۱۰ , ۱۲ , ۲۵ , ۲۸ , ۴۷ , ۵۲

۱ , ۳ , ۸ , ۱۰ , ۱۲ , ۲۵ , ۲۸ , ۴۷ , ۵۲

۱۵- بهترین حالت، بدترین حالت و حالت متوسط مرتب سازی سریع را با ارائه یک مثال و در هر حالت توضیح دهید.

پاسخ :

بهترین حالت، زمانی اتفاق می افتد که در هر مرحله، آریه ها نصف شوند، و یا با اختلاف تعداد حداکثر یک واحد تقسیم شوند. مانند آرایهٔ زیر (بررسی کنید).

۱۰،۸،۲،۱،۴،۱۶،۱۵،۱۷،۱۲،۱۹

بدترین حالت، زمانی اتفاق می افتد که در هر مرحله، عنصر محوری در یک کنار قرار گیرد. مانند آرایهٔ زیر (بررسی کنید).

۱۰،۹،۵،۷،۱

حالت متوسط زمانی اتفاق می افتد که، نتوان هیچ نظمی برای مکان قرار گیری عنصر محوری در مراحل مختلف پیدا کرد. مانند مثال زیر (بررسی کنید).

۱۷،۲۰،۱۰،۲۵،۱۱،۸،۱۸،۲۳

۱۶- یک الگوریتم غیر بازگشتی برای مرتب سازی سریع بنویسید الگوریتم را تحلیل کرده و نتایج را ارائه دهید.

پاسخ :

برنامه را به صورت کامل می نویسیم.

```
struct  stack {
        int a ;
        int b ;
} S[n];
int  top = -1 ;
Void quick sort (int n)
{
```

```
int pivot po int, low , high;
 push (۰,n);
 while (top! = −۱)
 {high = pop (۰);
    low= pop (۰);
    partition (low, high , pivotpo int)
    if (high > pivotpo int−۱)
       push (pivotpo int+۱ , high );
    else if (low < pivotpo int−۱);
          push (low , pivotpo int−۱);
 }
}
```

دقت: *Partition* همان تابعی است که در صفحهٔ ۹۹ کتاب نوشته شده است.

۱۷- الگوریت ضرب استراسن را در حالتی که از سه ضرب استفاده می کند را نوشته و تحلیل نمایید.

پاسخ :

صورت سؤال گنگ است.

۱۸- الگوریتم ضرب استراسن را با ارائه یک مثال توضیح دهید.

پاسخ :

مانند مثال ۴-۴ عمل می کنیم. فرض کنیم ماتریسهای A و B به صورت زیر باشند.

$$A=\begin{bmatrix} ۰ & ۲ & -۱ & ۲ \\ ۲ & ۰ & ۱ & ۱ \\ -۲ & -۱ & ۱ & ۰ \\ ۰ & ۰ & ۱ & -۱ \end{bmatrix} \qquad B=\begin{bmatrix} ۰ & ۱ & ۰ & ۲ \\ ۳ & ۴ & ۱ & -۱ \\ ۷ & ۰ & ۲ & ۰ \\ ۶ & -۱ & ۲ & -۱ \end{bmatrix}$$

$$\Rightarrow A_{۱۱}=\begin{bmatrix} ۰ & ۲ \\ ۲ & ۰ \end{bmatrix} \qquad A_{۱۲}=\begin{bmatrix} -۱ & ۲ \\ ۱ & ۰ \end{bmatrix} \qquad A_{۲۱}=\begin{bmatrix} -۲ & -۱ \\ ۰ & ۰ \end{bmatrix}$$

$$A_{۲۲}=\begin{bmatrix} ۱ & ۰ \\ ۱ & -۱ \end{bmatrix} \qquad B_{۱۱}=\begin{bmatrix} ۰ & ۱ \\ ۳ & ۴ \end{bmatrix} \qquad B_{۱۲}=\begin{bmatrix} ۰ & ۲ \\ ۱ & -۱ \end{bmatrix}$$

$$B_{۲۱}=\begin{bmatrix}۷ & ۰\\ ۶ & -۱\end{bmatrix} \qquad B_{۲۲}=\begin{bmatrix}۲ & ۰\\ ۲ & -۱\end{bmatrix} \Rightarrow$$

$$P=\begin{bmatrix}۱ & ۲\\ ۳ & -۱\end{bmatrix}\begin{bmatrix}۲ & ۱\\ ۵ & ۳\end{bmatrix}=\begin{bmatrix}۱۲ & ۷\\ ۱ & ۰\end{bmatrix}$$

$$Q=\begin{bmatrix}-۱ & -۱\\ ۱ & -۱\end{bmatrix}\begin{bmatrix}۰ & ۱\\ ۳ & ۴\end{bmatrix}=\begin{bmatrix}-۳ & -۵\\ -۳ & -۳\end{bmatrix}$$

$$R=\begin{bmatrix}۰ & ۲\\ ۲ & ۰\end{bmatrix}\begin{bmatrix}-۲ & ۲\\ -۱ & ۰\end{bmatrix}=\begin{bmatrix}-۲ & ۰\\ -۴ & ۴\end{bmatrix}$$

$$S=\begin{bmatrix}۱ & ۰\\ ۱ & -۱\end{bmatrix}\begin{bmatrix}۷ & -۱\\ ۳ & -۵\end{bmatrix}=\begin{bmatrix}۷ & -۱\\ ۴ & ۴\end{bmatrix}$$

$$T=\begin{bmatrix}-۱ & ۴\\ ۳ & ۰\end{bmatrix}\begin{bmatrix}۲ & ۰\\ ۲ & -۱\end{bmatrix}=\begin{bmatrix}۶ & -۴\\ ۶ & ۰\end{bmatrix}$$

$$U=\begin{bmatrix}-۲ & -۳\\ -۲ & ۰\end{bmatrix}\begin{bmatrix}۰ & ۳\\ ۴ & ۳\end{bmatrix}=\begin{bmatrix}-۱۲ & -۱۵\\ ۰ & -۶\end{bmatrix}$$

$$V=\begin{bmatrix}-۲ & ۲\\ ۰ & ۱\end{bmatrix}\begin{bmatrix}۹ & ۰\\ ۸ & -۲\end{bmatrix}=\begin{bmatrix}-۲ & -۴\\ ۸ & -۲\end{bmatrix} \Rightarrow$$

$$C_{۱۱}=\begin{bmatrix}۱۲ & ۷\\ ۱ & ۰\end{bmatrix}+\begin{bmatrix}۷ & -۱\\ ۴ & ۴\end{bmatrix}-\begin{bmatrix}۶ & -۴\\ ۶ & ۰\end{bmatrix}+\begin{bmatrix}-۲ & -۴\\ ۸ & -۲\end{bmatrix}=\begin{bmatrix}۱۱ & ۶\\ ۷ & ۲\end{bmatrix}$$

$$C_{۱۲}=\begin{bmatrix}۱۲ & ۷\\ ۱ & ۰\end{bmatrix}+\begin{bmatrix}۶ & -۴\\ ۶ & ۰\end{bmatrix}=\begin{bmatrix}۱۸ & ۳\\ ۷ & ۰\end{bmatrix}$$

$$C_{۲۱}=\begin{bmatrix}-۳ & -۵\\ -۳ & -۳\end{bmatrix}+\begin{bmatrix}۷ & -۱\\ ۴ & ۴\end{bmatrix}=\begin{bmatrix}۴ & -۶\\ ۱ & ۱\end{bmatrix}$$

$$C_{۲۲}=\begin{bmatrix}۱۲ & ۷\\ ۱ & ۰\end{bmatrix}+\begin{bmatrix}-۲ & ۰\\ -۴ & ۴\end{bmatrix}-\begin{bmatrix}-۳ & -۵\\ -۳ & -۳\end{bmatrix}+\begin{bmatrix}-۱۲ & -۱۵\\ ۰ & -۶\end{bmatrix}=\begin{bmatrix}۱ & -۳\\ ۰ & ۱\end{bmatrix}$$

$$\Rightarrow AB=\begin{bmatrix}۱۱ & ۶ & ۱۸ & ۳\\ ۷ & ۴ & ۷ & ۰\\ ۴ & -۶ & ۱ & -۳\\ ۱ & ۱ & ۰ & ۱\end{bmatrix}$$

۱۹- یک معادله بازگشتی برای الگوریتم اصلاح شده استراسن (توسط ساموئل وینوگراد) بنویسید که به جای ۱۸ عمل جمع و تفریق از ۱۵ عمل جمع و تفریق استفاده می کند. معادله حاصل را حل نمایید.

پاسخ :

$$T(n)=\begin{cases}۱ & n\le ۱\\ ۷T\left(n/۲\right)+۱۵\left(\frac{n}{۲}\right)^{۲} & \end{cases}$$

با انتخاب $n=۲^k$, $a_k=T(n)$ داریم:

$$a_k-۷a_{k-۱}=۱۵\times ۲^{۲k-۲}$$

یکبار k ها را به $k+۱$ تبدیل می کنیم و یکبار دو طرف را در ۴ ضرب می کنیم. پس:

$$\begin{cases}a_{k+۱}-۷a_k=۱۵\times ۲^{۲k}\\ ۴a_{۱۲}-۲۸a_{k-۱}=۱۵\times ۲^{k}\end{cases}\Rightarrow a_{k+۱}-۱۱a_k+۲۸a_{k-۱}=۰$$

$$\Rightarrow x^{۲}-۱۱x+۲۸=۰\Rightarrow x=۴\ ,\ x=۷\Rightarrow$$

$$a_k=c_۱۴^k+c_۲۷^k\Rightarrow T(n)=c_۱۴^{\log_۲^n}+c_۲۷^{\log_۲^n}=$$

$$c_۱n^{۲}+c_۲n^{\log_۲^۷}\in\theta\left(n^{\log_۲^۷}\right)$$

چون $\log_۲^۷<۲/۸۱$ پس این الگوریتم بهتر از الگوریتم استراسن کار می کند.

۲۰) فرض کنید که V,U دو عدد n بیتی بوده به نحوی که برای سادگی، n توانی از ۲ در نظر گرفته شده است. الگوریتم ضرب معمولی نیازمند $O(n^۲)$ عملگر است. یک الگوریتمی متکی بر روش تقسیم و حل با تقسیم اعداد به دو زیربخش و زیربخش ها ارائه دهید به طوری که:

$$UV = (a \times ۲^{n/۲} + b)(C \times ۲^{n/۲} + d)$$

$$UV = ac \times ۲^{n/۲} + (ad + bc)۲^{n/۲} + bd$$

با استفاده از این الگوریتم بازگشتی، ضرب های bd,bc,ad,ac انجام می شوند:

الف) پیچیدگی زمانی الگوریتم را تحلیل نمایید.

پاسخ :

```
Void    mult (U,V)
{
  if (U = ۰  or  V = ۰) then   return ۰
  else if (n = ۱), then return U * V
  else { m ← n/۲
       a ← U/۲^m   ;  b ← U%۲^m
       c ← V/۲^m   ;  d ← V%۲^m
       return (mult (a,c) * ۲^(۲m) + (mult(a,d) + mult(b,c)) * ۲^m + mult (b,d).
       }
}
```

برای تحلیل الگوریتم بالا، چون ضرب در توانهای مختلف مبنا (یعنی ضرب در ۲^m ها) هزینهٔ زمانی ندارد، پس تعداد ضربهای دیگر را می شماریم که چهار عدد میباشند. پس

$$T(n) = ۴T(\frac{n}{۲}) + \theta(n)$$

برای حل این رابطهٔ بازگشتی، با توجه به قضیهٔ ۱-۳ در صفحهٔ ۶۵ کتاب، به دست می آید.

$$T(n) \in \theta(n^۲)$$

ب) برای اصلاح الگوریتم بالا، به جای $ad+bc$ از $(a+b)(c+d)-(ac-bd)$ استفاده کنید. الگوریتم حاصل را تحلیل کنید؟

پاسخ :

سطر آخر الگوریتم قسمت الف را به صورت زیر اصلاح می کنیم:

$$return \left(mult\ (a,c)*۲^{۲m}+(mult(a+b\ ,\ c+d)-mult(a,c)-mult(b,d))*۲^{m}+mult(b,d)\right)$$

برای تحلیل این الگوریتم به رابطهٔ $T(n)=۳T\left(\frac{n}{۲}\right)+\theta(n)$ می رسیم که پس از حل به رابطهٔ $T(n)\in\theta\left(n^{\log_{۲}^{۳}}\right)$ می‌رسیم چون $\log_{۲}^{۳}<۲$ ، پس این الگوریتم کارایی بهتری نسبت به الف دارد.

۲۱- ضرب ماتریس های با اندازه $n\times n$ که در آن $n=۳^{k}$ باشد را در نظر بگیرید با استفاده از روش تقسیم و حل، مسئله را به ضرب ماتریس های ۳×۳ تبدیل نمایید روش معمولی برای محاسبه این فرآیند نیازمند ۲۷ ضرب می باشد تعداد دفعات لازم برای آنکه بتوان ضرب ۳×۳ انجام داد را مشخص کنید به طوری که زمان محاسباتی مورد نظر کمتر از $O\left(n^{۲/۸۱}\right)$ باشد.

پاسخ :

صورت مسئله گنگ است. اگر منظور طراح، پیدا کردن فرمولی مشابه فرمول استراسن است، آن را نمی‌توان به عنوان یک تمرین مقدماتی در نظر گرفت و احتیاج به ورزیدگی در مسائل تئوری جبر خطی دارد. اگر منظور طراح، اطلاع رسانی به دانشجو است که چنین الگوریتمی وجود دارد، قابل ذکر است که در کتابهای طراحی الگوریتم پیشرفته ، الگوریتم‌هایی معرفی می‌شوند که به جای ۲۷ ضرب ، از ۲۱ ضرب استفاده می‌کنند. در الگوریتم بیان شده رابطهٔ بازگشتی $T(n)=۲۱T\left(\frac{n}{۳}\right)+\theta(n^{۲})$ صادق است که با استفاده از قضیهٔ ۱-۳ صفحهٔ ۶۵ به دست می آید:

$$T(n)\in\theta\left(n^{\log_{۳}^{۲۱}}\right)$$

چون $\log_{۳}^{۲۱}<\log_{۲}^{۷}$ الگوریتم فوق نه تنها از الگوریتم استراسن، بلکه از الگوریتم بیان شده در تمرین ۱۹ نیز کاراتر است.

۲۲- (وینوگراد) فرض کنید $n = ۲P$ ، $V = (V_۱,...,V_n)$ ، $W = (w_۱,...,w_n)$ باشد آنگاه می توانیم با استفاده از فرمول زیر ضرب برداری VW را محاسبه کنیم:

$$\sum_{i=۱}^{P}(V_{۲i-۱}+W_{۲i})(V_{۲i}+W_{۲i-۱})-\sum_{i=۱}^{P}V_{۲i-۱}V_{۲i}$$

به طوری که این فرمول نیازمند $۳n/۲$ ضرب می باشد، نشان دهید که چگونه می توان از این فرمول برای ضرب دو ماتریس $n \times n$ با استفاده از فقط $n^۳/۲ + n^۲$ ضرب به جای $n^۳$ ضرب استفاده نمود.

پاسخ :

سؤال گنگ است. اگر منظور طراح، فرمول وینوگراد برای ضرب عددی دو بردار است به صورت زیر است.

$$\sum_{i=۱}^{p}(V_{۲i-۱}+W_{۲i})(V_{۲i}+w_{۲i-۱})-\sum_{i=۱}^{p}V_{۲i-۱}V_{۲i}-\sum_{i=۱}^{p}w_{۲i-۱}w_{۲i}$$

که این فرمول، تعداد ضربها را به $۳p$ تبدیل می کند، که چون $p=\frac{n}{۲}$ پس تعداد ضربها برابر $\frac{۳n}{۲}$ می شود. باید دقت کرد که $n < \frac{۳n}{۲}$ و چون ضرب عددی استاندارد دو بردار، تنها شامل n ضرب است، پس فرمول وینوگراد در این مورد کارایی خوبی ندارد و موارد استعمال دیگری دارد. پس بدیهی است الگوریتم ضرب ماتریسهایی که بر مبنای این فرمول طراحی شود، از $n^۳$ بدتر است. اگر منظور طراح سؤال، چیز دیگری است، از عبارت به کار برده شده در صورت سؤال، قابل برداشت نیست.

۲۳- الگوریتم $MaxMin$ را به یک روال غیر بازگشتی معادل تبدیل نمایید.

پاسخ :

فرض کنیم آریهٔ $a[\ ]$ داده شده و می خواهیم max و min آن را بیابیم. می‌توانیم الگوریتم غیر بازگشتی زیر را به کار ببریم. فرض کنیم طول آرایهٔ $a[\ ]$ برابر x باشد.

```
Void maxmin (a[ ], n)
{
   max ← a[۰], min ← a[۰]
    for (k = ۱ ; k < n ; ++k)
      if (a[k] > max), then  max ← a[k]
    else if (a[k] < min) then min ← a[k]
   return (max,min)
}
```

۲۴- بهترین و بدترین حالت متوسط الگوریتم *MaxMin* را با ارائه یک مثال توضیح دهید.

پاسخ :

بهترین حالت زمانی اتفاق می افتد که آرایهٔ مورد نظر به صورت صعودی مرتب شده باشد. در این صورت هیچ وقت دستور *else* اجرا نخواهد شد و در نتیجه تعداد مقایسه ها برابر $n-۱$ خواهد شد.

بدترین حالت زمانی اتفاق می افتد که آرایهٔ مورد نظر به صورت نزولی مرتب شده باشد. در این صورت در هر تکرار حلقه، هر دو مقایسهٔ مربوط به دستورات *if* و *else* اجرا خواهند شد و تعداد کل مقایسه برابر $۲(n-۱)$ خواهد بود.

در حالت میانگین تعداد مقایسه ها برابر $\frac{۳n}{۲}-۱$ خواهد بود.

۲۵- اگر در این مسأله پیدا کردن *Min,Max* بخواهیم مسأله اصلی را به سه زیر مسأله تقسیم کنیم در این صورت الگوریتم لازم برای این کار را طراحی نماید سپس زمان آن را تحلیل کنید.

پاسخ :

آرایه را $a[\ ]$ نامیده و کرانهای بالا و پایین آن را به ترتیب با H و L نمایش میدهیم ابتدا الگوریتم مربوط به *max* را می نویسیم.

```
Void  maximum (L, H, a [ ])
{
  if (L = H), then        max ← a[L]
  else {      k1 ← ⌊(L + H)/۳⌋
              k۲ ← ۲k1
              M1 ← maximum (L, k1, a)
              M۲ ← maximum (k1 + ۱, k۲, a)
              M۳ ← maximum (k۲ + ۱, H, a)
              if (M1 <= M۲ and M۳ <= M۲) ; then return(M۲)
              if (M1 <= M۳ and M۲ <= M۳) ; then return(M۳)
              if (M۲ <= M1 and M۳ <= M1) ; then return(M1)
          }
}
```

الگوریتم مربوط به *min* را هم به طریق مشابه می نویسیم. و از دنبال هم قرار دادن این دو الگوریتم، الگوریتم *max* *min* به دست می آید.

برای تحلیل این الگوریتم به رابطهٔ بازگشتی $T(n)=۳T\left(\frac{n}{۳}\right)+۱$ می رسیم. با انتخاب $n=۳^k$, $T(۳^k)=a_k$ به دست می آید.

$a_k - ۳a_{k-۱} = ۱$

k ها را به $k+۱$ تبدیل کرده و از رابطهٔ فوق کم می کنیم. پس

$$a_{k+۱} - ۴a_k + ۳a_{k-۱} = ۰ \Rightarrow x^۲ - ۴x + ۳ = ۰ \Rightarrow x = ۱, x = ۳$$

$$\Rightarrow a_k = c_۱ + c_۲ ۳^k \Rightarrow T(n) = c_۱ + c_۲ n \in \theta(n)$$

۲۶- الگوریتم فرض کردن یک صفحه شطرنجی را بنویسید سپس زمان آن را تحلیل نمایید.

پاسخ :

این الگوریتم نقطه ابتدایی و تعداد خانه های یک صفحهٔ شطرنجی را گرفته و آن را فرش می‌کند.

j و i نقطهٔ ابتدایی و n تعداد سطرها را نشان می دهد.

```
Void   carpet (a[ ][ ], i , j , n)
       {
          if (n ≤ ۴) {
             for (k = ۰ to n k++)
                  for (L = ۰ to n ; L++)
                     if (k = L) then a [i][j] = ۱
                     }
               else {
                        carpet (a , ۰ ,۰ , n/۲)
                        carpet (a , n/۲ , n/۲ , n/۲)
                        carpet (a , ۰ , n/۲ , n/۲)
                        carpet (a , n/۲ , ۰ , n/۲)
                        }
          }
```

برای تحلیل الگوریتم فوق دیده می شود که رابطهٔ بازگشتی

$$T(n)=۴T\left(\frac{n}{۲}\right)+\theta(n^۲)$$

باید حل شود. با استفاده از قضیهٔ ۱-۳ در صفحهٔ ۶۵ کتاب، به دست می آید.

$$T(n) \in O(n^{۲})$$

۲۷- الگوریتم ضرب دو چند جمله ای را نوشته و زمان انجام محاسبات آن را تحلیل کنید.

پاسخ :

روش کار چنین است که یک چند جمله ای به شکل

$$a_{۱}x^{n} + a_{n-۱}x^{n-۱} + ... + a_{۱}x + a_{۰}$$

را به صورت زیر نمایش می دهیم:

$$(a_{n}x^{n/۲} + ... + a_{n/۲})x^{n/۲} + (a_{n/۲-۱}x^{n/۲-۱} + ... + a_{۰})$$

پس هر چند جمله‌ای $p(x)$ از درجهٔ n را می توان به صورت $P_{۱}(x)x^{x/۲} + P_{۲}(x)$ نمـایش داد که $p_{۱}(x)$ و $p_{۲}(x)$ حداکثر از درجهٔ $\frac{n}{۲}$ می باشد.

بدون لطمه به کلیت می‌توان فرض کرد هر دو چند جمله ای از درجهٔ n هستند.

دقت شود که هر چند جمله ای از درجهٔ n، یعنی یک آرایه با طول $n+۱$.

۲۸- الگوریتم زمانبندی مسابقات را نوشته و تحلیل زمانی آن را انجام دهید.

پاسخ :

```
Void  mult (P[ ], Q [ ], n)
{
  if (n = ۱) then
      return (P[۱]*Q[۱]x^۲ + (P[۱]*Q[۲]+ P[۲]*Q[۱])x + P[۲]*Q[۲])
                                  else {
                                        for (n/۲  to n  , i ++)
                                          {P_۱[i] ← P[i]
                                           Q_۱[i] ← Q[i]
                                          }
                                        for (۰ to n/۲ − ۱ i++)
                                        {P_۲[i] ← p[i]
                                        Q_۲[i] ← Q[i]
                                        }
```

$$\text{eturn}\Big(mult\left(P_1, Q_1, n/2\right)* x^n + \Big(mult\left(P_1+P_2, Q_1+Q_2, n/2\right) - mult\left(P_1, Q_1, n/2\right) - mult\left(P_2, Q_2, n/2\right)\Big)* x^{n/2} + mult\left(P_2, Q_2, n/2\right)\Big)$$

}

}

برای تحلیل این الگوریتم، به رابطهٔ بازگشتی $T(n) = 3T\left(\frac{n}{2}\right) + \theta(n)$ می رسیم، که پس از استفاده از قضیهٔ ۳-۱ صفحهٔ ۶۵ کتاب، به دست می آید :

$$T(n) \in \theta\left(n^{\log_2^3}\right)$$

۲۹- با روش تقسیم و حل، مسأله برج هانوی را حل نمایید. تحلیل زمانی آن را با حالت معمولی مقایسه نمایید.

پاسخ :

روشهای کلاسیک تقسیم و حل، برای مسئله برجهای هانوی، قابل به کارگیری نیستند. چون مثلاً فرض کنیم $n/2$ تا مهرهٔ بالایی را از میلهٔ A به میلهٔ B بردیم. در این جابه جایی از میلهٔ C نیز استفاده می شود.

حال برای جابه جایی $n/2$ تا مهرهٔ بقیه از میلهٔ A به میلهٔ C لازم است از میلهٔ B استفاده کنیم که مجاز نیستیم، زیرا مهره های کوچکتری از قبل در آن قرار گرفته اند. یک روش تقسیم و حل غیر مفید به صورت زیر می توان گرفت.

ابتدا با $T(n/2)$ تا حرکت، $n/2$ مهرهٔ بالایی را از میلهٔ A به میلهٔ C ببریم. حال پس از هر حرکت برای جابه جایی $n/2$ مهرهٔ بقیه، کل $n/2$ مهرهٔ کوچکی که قبلاً روی میلهٔ C قرار گرفته اند را جابه‌جا کنیم.

چون برای این کار به $T(n/2)$ تا حرکت نیاز داریم، پس در این قسمت به $\left(T\left(\frac{n}{2}\right)\right)^2$ تا حرکت نیاز داریم. پس نهایتاً داریم:

$$T(n) = \left(T\left(\frac{n}{2}\right)\right)^2 + T\left(\frac{n}{2}\right)$$

۳۰- مجموعه A,B هر کدام دارای n عنصر به صورت آرایه های مرتب شده می باشند الگوریتمی ارائه دهید که $A \cup B$, $A \cap B$ را در زمانی برابر $O(n)$ ارائه دهد.

پاسخ :

C شامل اشتراک و D شامل اجتماع دو آرایه می باشد. قابل توجه است که هر مجموعه را به صورت یک آرایه در نظر می گیریم.

لازم به تذکر است که در الگوریتم زیر، مرتب بودن آرایه ها، نقشی اساسی بازی می کند.

```
{
    for (i = 0 to n , i++)
    {   while (A[i] > B[j])
        {  if (j = n), then Exit ( );
           D[i++] ← B[j++]
        }
        if (B [j] = A[i] ),
        {   j++
              C[k++] ← A[i]
              D[i++] ← A[i]
            }
          if (B[j] > A[i], then D[i++] ← A[i])
        }
    }
```

۳۱- مجموعه های B,A به ترتیب دارای n,m عنصر به صورت لیست خطی هستند. این مجموعه الزاماً مرتب شده نیستند $(m \le n)$ نشان دهید که $A \cup B$, $A \cap B$ در زمانی برابر $O(n \text{ Log } n)$ محاسبه می شوند.

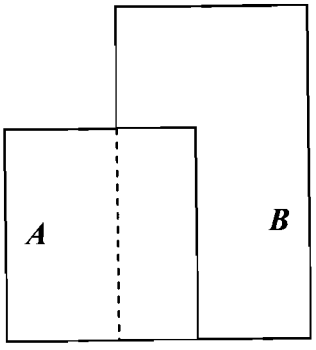
پاسخ :

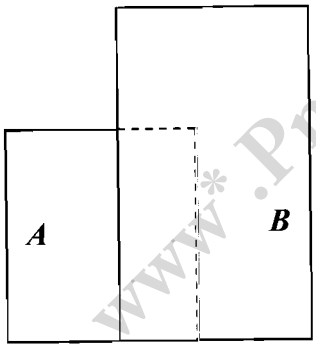
ابتدا، هر دو آرایه را با زمان $O(n\log n)$ مرتب می کنیم و سپس با استفاده از الگوریتم بیان شده در تمرین۳۰، اجتماع و اشتراک آنها را در زمان $O(n)$ به دست می آوریم.

۳۲- n ساختمان داریم که نمای هر یک از آنها از جلو به صورت مستطیل هستند که با سه عدد k,j ،i نمایش داده می شود. I طول از مبدأ شروع ساختمان، j ارتفاع و k طول از مبدأ پایان ساختمان می‌باشد. می خواهیم نمای کلی مجموعه ساختمان ها را رسم کنیم و خطوط نامرئی را حذف کنیم. الگوریتم مربوطه را طراحی کنید. مرتبه زمانی الگوریتم را تحلیل نمایید.

پاسخ :

سؤال ناقص است. باید عمق ساختمانها را نیز داشته باشیم. مثلاً برای دو ساختمان AوB در زیر، هر یک از دو حالتی که در زیر رسم شده اند، می تواند جواب باشد، مگر اینکه مؤلفه عمق ساختمان را هم داشته باشیم، که در آن صورت تنها یک جواب قابل قبول است.





۳۳- مسئله مرتب سازی یک رشته S از بیت ها که هر کدام به عنوان یک کلید هستند و دارای ارزش ۱و۰ می باشند را در نظر بگیرید، یک روش مرتب سازی S در حقیقت شروع با دو بیت تهی L_0 و L_1 است. فرض کنید $S = k_1, k_2, ..., k_n$ باشد موارد زیر را انجام دهید:

- اگر $k_i = 0$ باشد آنگاه k_i برابر ۰ خواهد بود.
- اگر $k_i = 1$ باشد انگاه k_i برابر ۱ خواهد بود.

بعد از پردازش تمام بیت ها رشته S، به این صورت لیست L_5 و به دنبال آن لیست L_1 را نمایش دهید. الگوریتم حاصل را تحلیل زمانی کنید.

پاسخ :

۳۴- (ستاره مشهود) می خواهیم با حداقل تعداد پرسش از نفر n، فردی که ویژگی های یک ستاره مشهور (*Celebrity*) را دارد، در صورت وجود پیدا کنیم یک نفر ستاره است اگر بقیه او را از قبل بشناسد و او با آنها آشنا نباشد ما مجاز هستیم از a بپرسیم که آیا b را می شناسد؟ این یک پرسش است توجه کنید که در این مسئله رابطه شناختن یک رابطه یک طرفه است. چون در این مسئله $n(n-1)/2$ گروه دو نفری داریم چون اگر پرسش به طور دلخواه باشد، در بدترین حالت $n(n-1)$ پرسش داریم. الگوریتمی برای کمترین تعداد پرسش، برای پیدا کردن ستاره مشهور ارائه دهید.

پاسخ :

نحوهٔ کار چنین است که از همهٔ آدمها می پرسیم آیا نفر اول را می شناسند؟ اگر همهٔ جوابها مثبت بود، نفر اول جواب است. اولین نفری که گفت نفر اول را نمی‌شناسد، نفر دوم کاندید می شود. همین طور ادامه می دهیم تا به جواب برسیم. برای فرموله کردن الگوریتم، مجموعهٔ A را شامل همهٔ زوج مرتبهایی مانند (K,L) می گیریم که K، L را می شناسد.

```
Void  celebrity (A,n)
    {
      for  i=1  to n ;  i++
           for  j=1  to n ;  j++
           if  ( (j,i) ∉ A); then Exit
           if  (j=n) ; then  return (i)
    }
```

تمرینات فصل پنجم

۱- نشان دهید که الگوریتم خرد کردن پول اگر سکه های موجود ۱، ۲، ۵ و ۷ تومانی باشد جواب بهینه را به دست می دهد.

پاسخ :

فرض کنیم F یک مجموعه امید بخش از جواب ها باشد و e، طبق الگوریتم خرد کردن پول به آن اضافه شده باشد. کافیست ثابت کنیم $F \cup \{e\}$ نیز یک مجموعه موجه است. اگر این حکم ثابت شود آنگاه با استقراء، حکم اصلی ثابت می شود. چند حالت در نظر می گیریم:

حالت اول: اگر e برابر با هفت تومانی باشد پس F شامل فقط ۷ تومانی است و چون $۱+۲<۷$, $۲+۵\leq ۷$, $۱+۵<۷$ پس هر جایگزینی دیگری به جای ۷ تومانی، تعداد سکه ها را بیشتر می کند.

حالت دوم: اگر e برابر با پنج تومانی باشد، پس F شامل ۷ توامانی و احتمالاً پنج تومانی است و چون $۵ > ۲+۱$ پس هر جایگزینی دیگری به جای ۵ تومانی، تعداد سکه ها را بیشتری می کند.

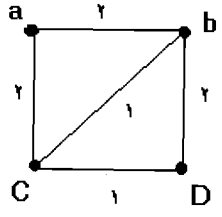
حالت سوم: اگر e برابر با دو تومانی باشد، برای حذف آن باید از دو عدد یک تومانی استفاده کرد که تعداد سکه ها را بیشتر می کند.

حالت چهارم: اگر e برابر با یک تومانی باشد، پس در مرحله آخر الگوریتم هستیم و جایگزین دیگری نداریم.

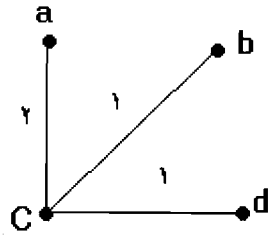
۲- یک گراف وزن دار را در نظر گرفته و مثالی بزنید که بیش از یک درخت پوشای بهینه داشته باشد. سپس همه درخت های پوشای بهینه این گراف را به دست آورید.

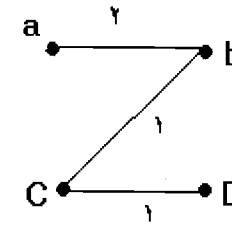
پاسخ :

فرض کنیم G به صورت زیر باشد.

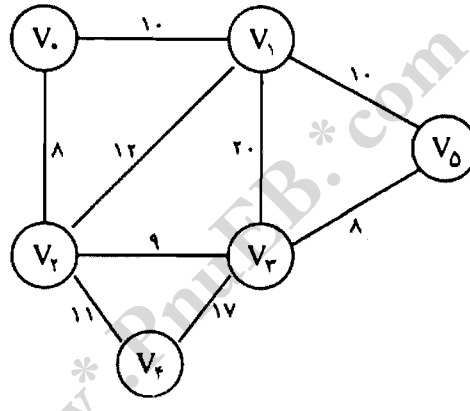


آنگاه هر یک از درختان پوشای زیر، درختان پوشای بهینه آن هستند.





۳- الگوریتم پریم را برای پیدا کردن درخت پوشای کمینه گراف زیر به کار برده، مرحله به مرحله عملیات را نمایش دهید:



پاسخ :

از یک رأس دلخواه مثلاً از V_5 شروع می کنیم. اول از همه $\overline{V_5V_3}$ انتخاب می شود. سپس به ترتیب $\overline{V_5V_1}$ ، $\overline{V_2V_0}$ انتخاب می شوند. در اینجا می توان هر یک از بالهای $\overline{V_5V_1}$ و یا $\overline{V_1V_0}$ را در نظر گرفت. و در مرحله نهایی یال $\overline{V_2V_4}$ انتخاب خواهد شد. پس دو درخت پوشای زیر بهینه هستند:

$$T_1 = \{V_5V_3, V_3V_2, V_2V_0, V_5V_1, V_2V_4\}$$

$$T_2 = \{V_5V_3, V_3V_2, V_2V_0, V_1V_0, V_2V_4\}$$

۴- کارایی الگوریتم پریم را با ارائه مثال های مختلف بررسی نمایید.

پاسخ :

باید چند گراف بکشیم و روی هر کدام الگوریتم را اجرا کنیم. مثلاً می توان به مثال ۳-۵ کتاب یا تمرین ۳ همین فصل مراجعه کرد.

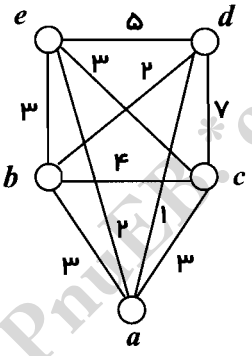
۵- مثالی در مورد برق کشی یک ساختمان ۵ طبقه بزنید و گراف خود را بر اساس رئوسی که جریان در آنها مورد استفاده قرار می گیرد تشکیل دهید سپس آن را با الگوریتم پریم حل نمایید.

پاسخ :

یک گراف با پنج رأس در نظر می گیریم که هر رأس نشان دهنده یک طبقه است. بین هر دو طبقه، یالی رسم می کنیم که وزن آن برابر هزینه سیم کشی بین آن دو طبقه است. به دنبال یک درخت پوشای بهینه می گردیم، که با الگوریتم پرایم قابل محاسبه است. در گراف زیر از رأس *d* شروع کرده و به درخت *T* می رسیم.

(البته می توان درختان بهینه دیگری هم دست یافت)

ابتدا یال *da*، سپس یال *db*، بعد یال *ae*، و در انتها یال *ec* انتخاب می شود :



$$T = \{da\ ,\ ae\ ,\ db\ ,ec\}$$

۶- الگوریم کروسکال را برای تمرین ۳ به کار برده و درخت پوشای کمینه گراف را به دست آورید.

پاسخ :

ابتدا بال ها را به طور صعودی مرتب می کنیم:

$$V_3V_5\ ,\ V_0V_2\ ,\ V_2V_3\ ,\ V_1V_5\ ,\ V_0V_1\ ,\ V_2V_4\ ,\ V_1V_2\ ,\ V_3V_4\ ,\ V_1V_3$$

حال مرحله به مرحله به صورت زیر جلو می رویم:

$T = \varnothing$ $\quad \{V_0\}\ \{V_1\}\ \{V_2\}\ \{V_3\}\ \{V_4\}\ \{V_5\}$

$T = \{V_3V_5\}$ $\quad \{V_0\}\ \{V_1\}\ \{V_2\}\ \{V_3,V_5\}\ \{V_4\}$

$T = \{V_3V_5\ ,\ V_0V_2\}$ $\quad \{V_0\ ,\ V_2\}\ \{V_1\}\ \{V_3\ ,\ V_5\}\ \{V_4\}$

$T = \{V_3V_5\ ,\ V_0V_2\ ,\ V_2V_3\}$ $\quad \{V_0\ ,\ V_2\ ,\ V_3\ ,\ V_5\}\ \{V_1\}\ \{V_4\}$

$T = \{V_3V_5\ ,\ V_0V_2\ ,\ V_2V_3,V_1V_5\}$ $\quad \{V_0\ ,\ V_1\ ,\ V_2\ ,\ V_3,V_5\}\ \{V_4\}$

$T = \{V_3V_5\ ,\ V_0V_2\ ,V_2V_3\ ,\ V_1V_5\ ,\ V_2V_4\}$ $\quad \{V_0\ ,\ V_1\ ,V_2\ ,\ V_3\ ,\ V_4\ ,\ V_5\}$

۷- لوله کشی یک ساختمان ۵ طبقه را در نظر گرفته، سپس الگوریتم کروسکال را برای پیدا کردن درخت پوشای کمینه که هدف آن پیدا کردن کوتاهترین مسیر لوله کشی از یک منبع به یک مقصد می باشد را بیابید.

پاسخ :

مجدداً گراف مربوط به مسئله ۵ را می کشیم و با استفاده از الگوریتم کروسکال به T میرسیم.

۸- با ارائه مثال هایی از گراف ها (گراف کامل و غیر کامل) الگوریتم های پریم و کروسکال را برای پیدا کردن درخت پوشای بهینه به کار برده، نتایج حاصل را از نظر زمان اجرا با هم مقایسه کنید.

پاسخ :

این مسئله یک پروژه عملی است ولی در حالت کلی هر چه گراف ما چگالتر باشد (یعنی به گراف کامل نزدیکتر باشد) کارایی الگوریتم پریم بهتر از کروسکال می شود و هرچه گراف به درخت نزدیکتر شود، کارایی الگوریتم کروسکال بهتر از پریم می شود.

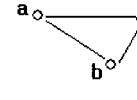
۹- فرض کنید یک گراف بدون جهت باشد. می خواهیم رئوس گراف را طوری رنگ آمیزی کنیم به طوری که هیچ دو رأس گراف هم رنگ نباشند و همچنین تعداد رنگ های استفاده شده برای رنگ‌آمیزی گراف کمترین تعداد ممکن باشد. (به این مسئله رنگ آمیزی گراف می گویند).

پاسخ :

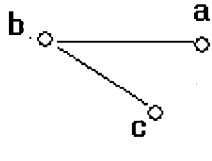
به رأس اول، رنگ اول را نسبت می دهیم. رأس دوم اگر به رأس اول متصل بود با رنگ دوم و اگر متصل نبود، با رنگ اول رنگ می شود. همین روال را تا رأس آخر ادامه می دهیم. باید دقت کرد که طرح گفته شده، مقدماتی ترین طرح می باشد و از مرتبه نمایی است. برای ملاحظه یک طرح کاراتر (که البته آن هم از مرتبه نمایی است) به صفحه ۲۶۴ کتاب مراجعه شود.

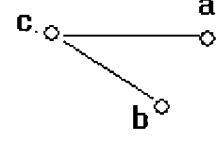
۱۰- نشان دهید که تعداد درخت های پوشای یک گراف کامل می تواند از $۲^{n-1} - ۲$ بیشتر باشد.

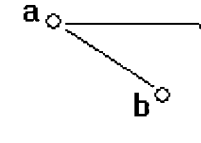
پاسخ :

برای گراف کامل سه رأسی مثل 

سه درخت پوشا داریم:



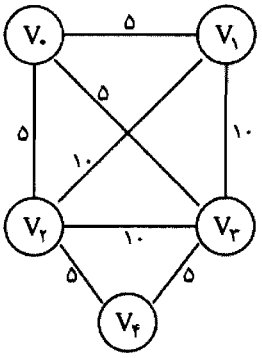




ولی $۲^{۳-۱} - ۲ = ۲$

تذکر: طبق قضیه کیلی، تعداد درختان پوشای گراف کامل k_n برابر n^{n-2} است (البته خیلی از آنها با هم یک ریخت می باشند، مانند مثال ارائه شده).

۱۱- در گراف زیر چند درخت پوشای کمینه دارد:



پاسخ :

دو درخت پوشای کمینه داریم (که طبق الگوریتم پریم یا کروسکال قابل بررسی می باشند)

$$T_1 = \{V_4V_3 , V_4V_2 , V_2V_0 , V_0V_1\}$$

$$T_2 = \{V_4V_3 , V_4V_2 , V_3V_0 , V_0V_1\}$$

۱۲- الف) نشان دهید که اگر یک درخت پوشا برای گراف بدون جهت G باشد آنگاه افزودن یک یال P به T، به نحوی که $P \notin E(T)$, $P \in E(G)$ باشد، یک دور منحصر به فرد ایجاد می کند.

پاسخ :

فرض کنیم یال P به صورت $\overline{ab}$ باشد که $a,b \in V(G)$. چون در T (طبق خاصیت درختها) یک مسیر منحصر به فرد از a به b مانند $ac_1c_2c_3....c_kb$ وجود دارد با اضافه کردن P به T، دور زیر حاصل می شود:

$$ac_1c_2c_3....c_kba$$

با توجه به منحصر به فرد بودن مسیر فوق، دور به دست آمده نیز منحصر به فرد خواهد بود.

ب) نشان دهید که اگر هر یک از یال های این دور منحصر به فرد از $E(T)\cup\{P\}$ حذف شود، آنگاه یال‌های باقیمانده تشکیل یک درخت پوشا برای G می دهند.

پاسخ :

فرض کنیم یال حذف شده به صورت $\overline{c_i c_{i+1}}$ باشد و u_2, u_1 دو رأس دلخواه از $T_1 = T\cup\{P\} - \{\overline{c_i c_{i+1}}\}$ باشند. می دانیم در T یک مسیر منحصر به فرد از u_1 به u_2 موجود است. دو حالت ممکن است اتفاق بیفتد:

حالت اول: مسیر فوق از $\overline{c_i\, c_{i+1}}$ نمی گذرد. در این حالت همان مسیر در T_1 نیز u_1 را به u_2 وصل می کند و منحصر به فرد نیز هست.

حالت دوم: مسیر فوق از $\overline{c_i\, c_{i+1}}$ می گذرد. در این حالت به جای یال $\overline{c_i\, c_{i+1}}$، از مسیر $c_i c_{i-1}...c_1 abc_k c_{k-1}...c_{i+1}$ استفاده می کنیم و مسیری منحصر به فرد از u_1 به u_2 در T_1 می‌سازیم.

پس در هر دو حالت، در T_1 می توان بین هر دو رأسی یک مسیر منحصر به فرد پیدا کرد و این یعنی T_1 یک درخت است.

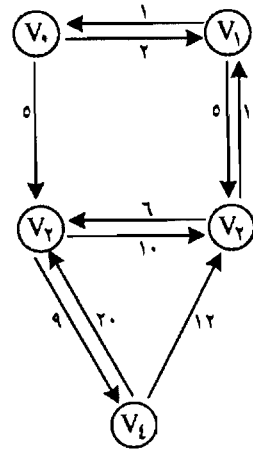
۱۳- فرض G یک گراف همبند و وزن دار باشد آنگاه:

اگر C یک دور از G باشد. نشان دهید که یال با بیشترین وزن از C نمی تواند به یک درخت پوشای کمینه از G تعلق داشته باشد.

پاسخ :

فرض کنیم T درختی پوشا باشد که شامل یال P باشد (P را همان یالی از C می گیریم که دارای بیشترین وزن است). پس حداقل یک یال مانند q از c در T نمی باشد (چون اگر همه یالهای C در T باشد، T شامل دور می شود). طبق مسئله قبل $T \cup \{q\} - \{p\}$ یک درخت پوشا است و چون وزن q از وزن p کمتر است، وزن این درخت جدید از وزن T کمتر است. پس T نمی تواند درخت کمینه باشد.

۱۴- الگوریتم دایکسترا را برای گراف زیر به کار بگیرید:



پاسخ :

از رأس V_1 شروع می کنیم و به مسیرهای زیر دست می یابیم:

تذکر: چون در شکل دو رأس V_3 داریم، رأس سمت راست را V_3 می نامیم:

مرحله اول	V_1V_0
مرحله دوم	V_1V_0 , V_0V_2
مرحله سوم	V_1V_3
مرحله چهارم	V_1V_0 , V_0V_2 , V_2V_4

۱۵- آیا الگوریتم دایکسترا را باری گرافی که دارای یال با وزن منفی می توان به کار برد یا نه؟

پاسخ :

خیر. تمام الگوریتم هایی که در این کتاب، برای کوتاهترین فاصله ارائه می شوند (مانند الگوریتم‌های دایکسترا ، فلوید، و غیره) روی گرافهایی با وزن های مثبت کار می کنند.

۱۶- یک جواب بهینه برای مسئله کوله پشتی در حالتی که

$W = (۵, ۴, ۳, ۶, ۱, ۴, ۱)$, $P = (۱۰, ۱۵, ۵, ۸, ۱۲, ۲۰, ۴)$, $m = ۱۵$, $n = ۷$

است، بیابید.

پاسخ :

به ترتیب اجسام پنجم، ششم، هفتم، دوم و اول انتخاب می شوند. پس مجموع وزن های آنها $۱۵ = ۵ + ۴ + ۱ + ۴ + ۱$ و مجموع ارزش آنها $۴۵ = ۵ + ۴ + ۴ + ۲۰ + ۱۲$ می باشد.

۱۷- مسئله کوله پشتی ۰/۱ را به صورت زیر تعریف می کنیم:

$$Max \sum_{i=1}^{n} p_i x_i$$

$$Subject\ to \sum_{i=1}^{n} W_i x_i \le m$$

$$and\ x_i = ۰ \text{ یا } ۱ \ , \ ۱ \le i \le n$$

یک روش حریصانه برای حل مسئله کوله پشتی ۰/۱ ارائه دهید.

پاسخ :

به منظور فهم بهتر، الگوریتم را به زبان ساده، و مرحله به مرحله بیان می کنیم.

۱- قرار دهیم $t_i = \frac{p_i}{w_i}$

۲- t_iها را از بزرگ به کوچک مرتب کنیم. اندیس x_iها را بر حسب این ترتیب، شماره گذاری می کنیم.

۳- قرار دهیم $x_i = ۱ \ , \ i = ۱$

۴- اگر $\sum_{k=1}^{i} x_k w_k \le m$ **آنگاه قرار می دهیم** $x_i = ۱$**، مگر نه قرار می دهیم** $x_i = ۰$

۵- i را به $i+1$ تبدیل می کنیم و بر می گردیم به مرحله ۳ .

تذکر: این الگوریتم لزوماً جواب بهینه را پیدا نمی کند، مانند مثال زیر

$$P=(۸,۸,۳۱,۸,۸)$$

$$W=(۴,۴,۱۵,۴,۴) \qquad m=۱۶$$

با الگوریتم بالا به دست می آید $x_۱=x_۲=x_۴=x_۵=۰$, $x_۳=۱$ و ارزش کلی برابر ۳۱ می شود.

ولی اگر قرار دهیم $x_۱=x_۲=x_۴=x_۵=۱$, $x_۳=۰$ ارزش کلی برابر ۳۲ خواهد شد.

۱۸- با ارائه مثال هایی حالتهای مختلف کوله پشتی را بررسی نمایید.

پاسخ :

باید برنامه آنرا نوشت و چندین مثال به کامپیوتر داد و نتیجه ها را بررسی کرد. این مسئله یک پروژه عملی است.

۱۹- یک مجموعه شامل n برنامه، موجود است، برنامه iام، زمان پردازش t_i و مهلت نهایی d_i دارد که با توجه به آنها تکمیل شود. یک برنامه زمانبندی ممکن، یک جایگشت از برنامه هاست به طوری اگر برنامه ها به آن ترتیب پردازش شوند، آنگاه هر برنامه تا زمان مهلت نهایی اش به اتمام می رسد. یک برنامه زمانبندی حریصانه تعریف کنید که طبق آن، برنامه ها با یک ترتیب نزولی از مهلت های نهایی پردازش می شوند.

پاسخ :

صورت مسئله غلط به نظر می رسد چون در هدف مسئله زمان های پردازش هیچ دخالتی ندارند. آنچه از صورت مسئله فهمیده می شود برای حل آن باید برنامه ها را بر حسب مهلتهای نهایی به طور نزولی مرتب کنیم، سپس از ابتدا تا سطری از جدول پیش رفت که مهلت نهایی از شماره سطر کمتر نباشد.

مثلاً اگر جدول زیر را در نظر بگیریم:

$x_۱$	$x_۲$	$x_۳$	$x_۴$	$x_۵$	$x_۶$
۳	۳	۴	۲	۱	۵

ابتدا به صورت

$x_۶$	۵
$x_۳$	۴
$x_۱$	۳
$x_۲$	۳
$x_۴$	۲
$x_۵$	۱

در می آید و چون برای $x_۲$، مهلت نهایی (۳) از شماره سطر (۴) کمتر است پس فقط ۳ سطر اول را انتخاب کرده و برنامه ها را از بالا به پایین پردازش می کنیم.

۲۰- کارها و زمان های خدمات دهی زیر را در نظر بگیرید:

زمان خدمات دهی	کار
۷	۱
۵	۲
۱۵	۳
۳	۴
۱۷	۵

مسئله زمانبندی بالا را حل نمایید. (با استفاده از الگوریتم ۷-۵)

پاسخ :

زمان	کار
۳	۴
۵	۲
۷	۱
۱۵	۳
۱۷	۵

زمان بهینه:

$$۳+(۳+۵)+(۳+۵+۷)+(۳+۵+۷+۱۵)+(۳+۵+۷+۱۵+۱۷)=۱۰۳$$

۲۱- کارها، سودها و مهلت هر کدام از ۷ کار در جدول زیر آمده است:

کار	مهلت	بهره
۱	۱	۵
۲	۳	۱۰
۳	۴	۱
۴	۳	۱۲
۵	۲	۶
۶	۱	۸
۷	۲	۲۰
		۱۰

مسئله زمانبندی بالا را با استفاده از الگوریتم ۸-۵ حل نمایید.

پاسخ :

ابتدا جدول بر حسب بهره به طور نزولی مرتب می شود. یعنی:

کار	مهلت	بهره
۷	۲	۲۰
۴	۳	۱۲
۲	۳	۱۰
۶	۱	۸
۵	۲	۶
۱	۱	۵
۳	۴	۱

با $j = 0$ شروع می کنیم و مراحل الگوریتم را ادامه می دهیم.

مجموعه امکان پذیر	سود	j	مرحله
هست	۰	۰	۰
هست	۲۰	{۷}	۱
هست	۳۲	{۷,۴}	۲
هست	۴۲	{۷,۴,۲}	۳
نیست	۵۰	{۶,۷,۴,۲}	۴
نیست	۴۸	{۷,۵,۴,۲}	۵
نیست	۴۷	{۱,۷,۴,۲}	۶
هست	۴۳	{۷,۴,۲,۳}	۷

پس جواب بهینه {۷,۴,۲,۳} با سود ۴۳ می باشد.

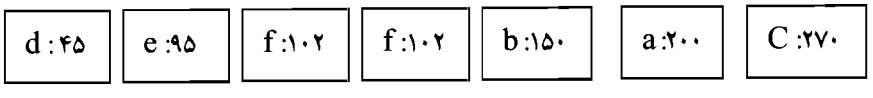
۲۲- کد گذاری هافمن را برای متن با مشخصات ذیل اعمال نموده، کد حاصل را ارائه دهید. متن شامل حروف G,f,e,d,c,b,a می باشد.

کاراکترها	a	b	c	d	e	F
تعداد تکرار	۲۰۰	۱۵۰	۲۷۰	۴۵	۹۵	۱۰۲

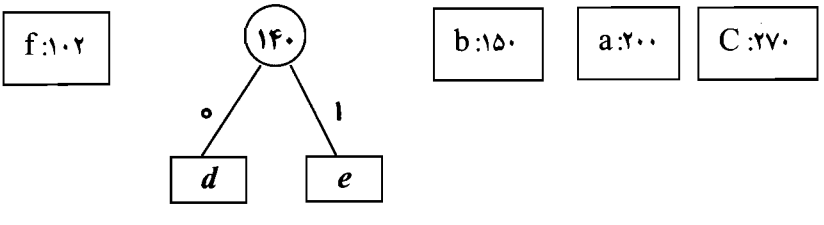
پاسخ :

مرحله به مرحله را می نویسیم:

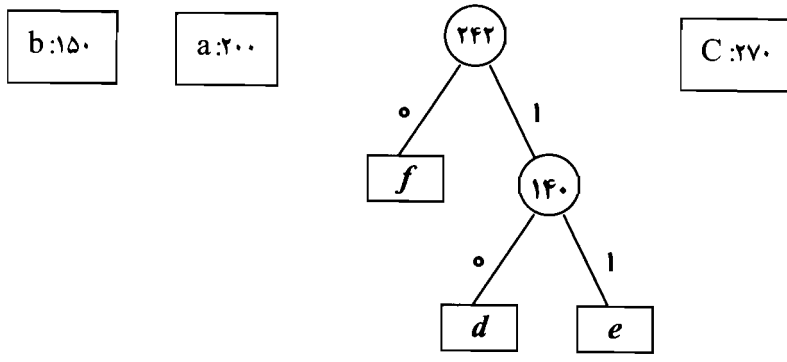
مرحله اول



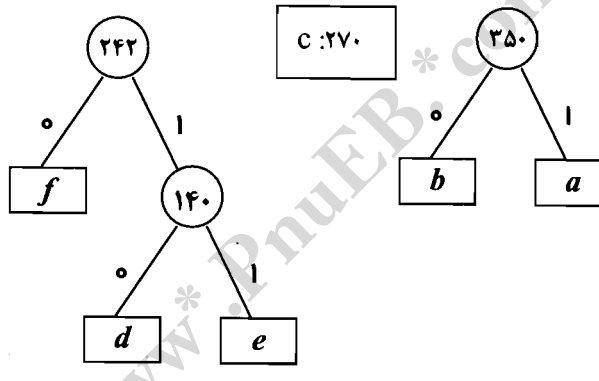
مرحله دوم



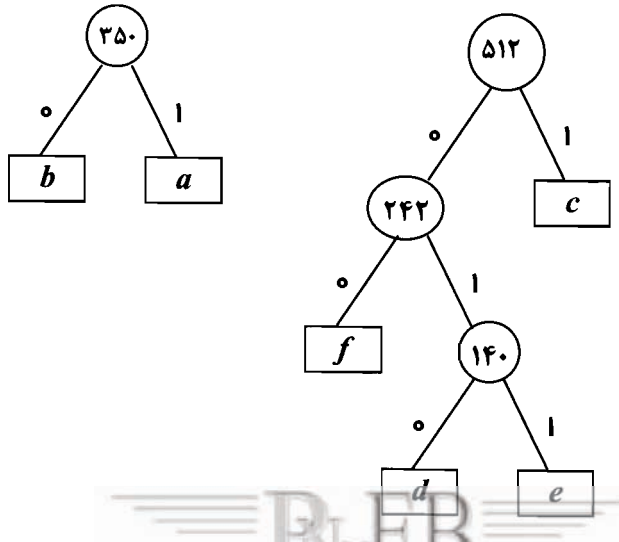
مرحله سوم



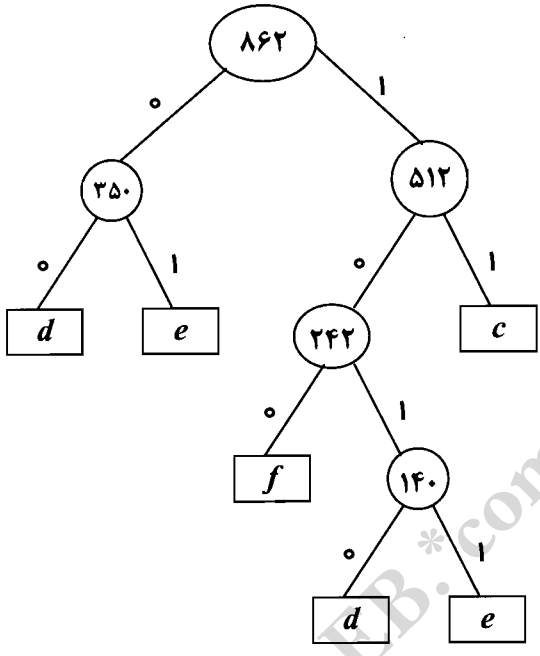
مرحله چهارم



مرحله پنجم :



مرحله ششم :



نتیجه:

a	b	c	d	e	f
۰۱	۰۰	۱۱	۱۰۱۰	۱۰۱۱	۱۰۰

۲۳- ثابت کنید خروجی حاصل از الگوریتم هافمن یک درخت دودوئی بهینه می باشد.

پاسخ :

منظور از درخت دودویی بهینه، درختی است که پس از کد گذاری شدن یالها توسط ۰ و۱ و کدگذاری حروف، ذخیره سازی رشته کمترین فضا را اشغال کند. از آنجایی که فضای اشغالی هر حرف برابر عمق آن در درخت دودویی است، پس واضح است که هر چه تعداد دفعات تکرار یک حرف بیشتر باشد، باید در عمق کمتری قرار بگیرد.

ساختار درخت دودویی هافمن به گونه ای است که اولاً کمترین عمقهای ممکن مورد استفاده قرار می گیرند و ثانیاً هر حرفی که تعداد تکرار آن بیشتر باشد در عمق کمتری قرار می‌گیرد، و این دو واقعیت باعث بهینه شدن آن می شود.

۲۴- یک شیوه ذخیره سازی بهینه برای ۱۱ برنامه بر روی سه نوار T_2, T_1, T_0 پیدا کنید. که در آن طول برنامه ها به ترتیب ۵، ۱۲، ۸، ۲۰، ۴، ۶، ۱۷، ۲، ۹، ۱۰و۱۱ می باشد.

پاسخ :

صورت مسئله گنگ است چون منظور از بهینه مشخص نیست و همچنین در رابطه با طول نوارها چیزی نمی دانیم.

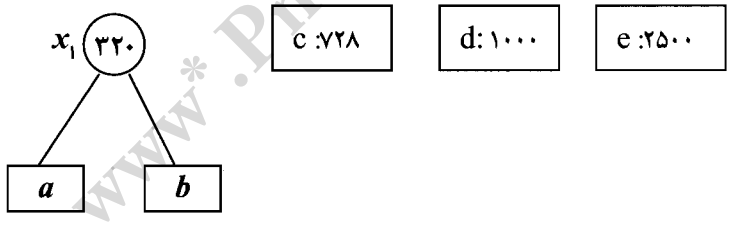
۲۵- فرض کنید ۵ فایل به ترتیب با ۱۰۰۰، ۱۵۰، ۲۵۰۰، ۷۲۸و ۱۷۰ رکورد به طور مرتب موجودند، می‌خواهیم این فایل ها را با هم ادغام کنیم، درخت دودوئی ادغام بهینه برای ادغام این فایل ها ارائه دهید.

پاسخ :

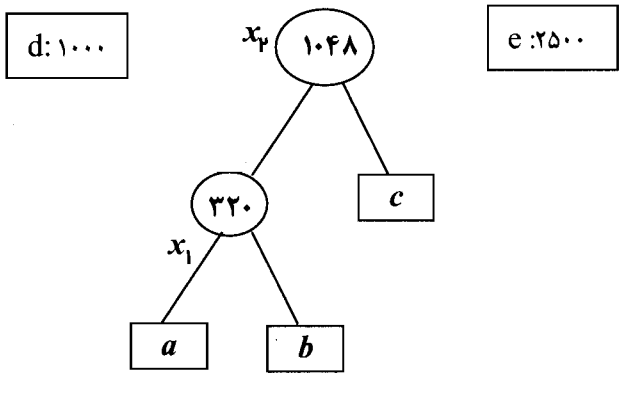
مانند الگوریتم هافمن عمل می کنیم:

مرحله اول

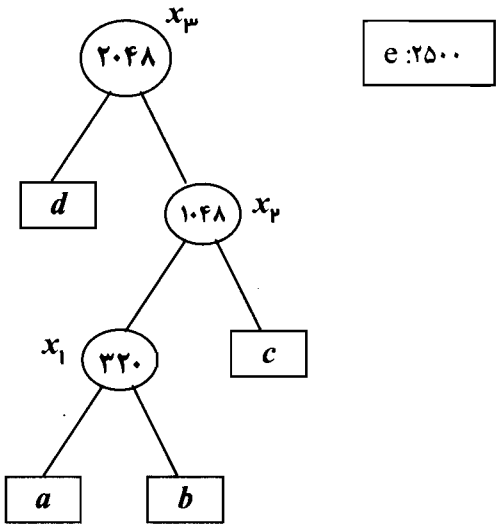
مرحله دوم



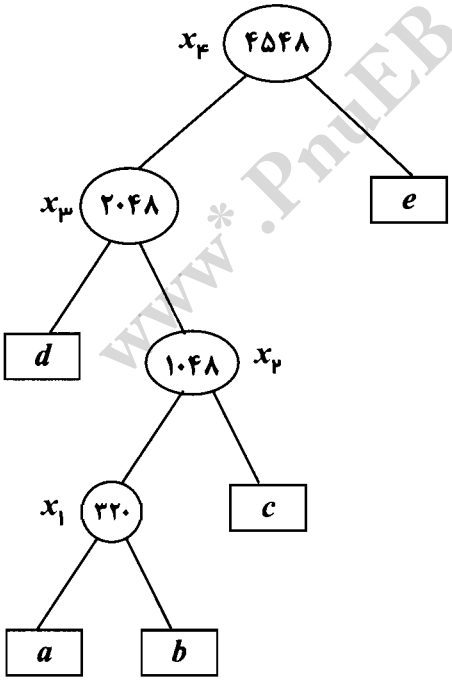
مرحله سوم



مرحله چهارم



مرحله پنجم :



۲۶- فرض کنید n فایل به طول $L_1,...,L_n$ بر روی یک نوار توانند ذخیره شوند. همچنین می دانیم که فایل iام به تعداد F_i مرتبه ارزیابی می شود. اگر فایل ها به ترتیب $L_1,...,L_n$ ذخیره شوند زمان انتظار برای بازیابی برابر است با:

$$\frac{\left[\sum_i \left(F_{iJ} \sum_{k=1}^{i} L_{ik} \right)\right]}{\sum F_i}$$

الف) نشان دهید ذخیره سازی فایل ها به ترتیب غیر نزولی L_i لزوماً زمان بازیابی را حداقل نمی کند.

پاسخ :

غلط چاپی دارد. به نظر می رسد فرمول زمان انتظار به صورت زیر باشد:

$$\left[\sum_{i=1}^{n} F_i \left(\sum_{k=1}^{i} L_k \right)\right] / \sum_{i=1}^{n} F_i$$

سه فایل به طول های ۹ و ۷و ۵ و به تعداد ارزیابی ۱۸ و ۲۱ و ۵ را در نظر بگیرید. اگر قرار دهیم:

$$L_1 = ۵ \qquad L_2 = ۷ \qquad L_3 = ۹$$
$$F_1 = ۵ \qquad F_2 = ۲۱ \qquad F_3 = ۱۸$$

L_i ها، غیر نزولی هستند و داریم:

$$\sum_{i=1}^{۳} F_i = ۴۲ \quad , \quad F_1 L_1 + F_2 \left(L_1 + L_2\right) + F_3 \left(L_1 + L_2 + L_3\right) = ۶۵۵$$

$$\Rightarrow \text{زمان انتظار} = \frac{۶۵۵}{۴۲} \approx ۱۵/۵۹$$

اگر قرار دهیم

$$L_1 = ۹ \qquad L_2 = ۷ \qquad L_3 = ۵$$
$$F_1 = ۱۸ \qquad F_2 = ۲۱ \qquad F_3 = ۵$$

داریم:

$$\sum_{i=1}^{۳} F_i = ۴۲ \quad , \quad F_1 L_1 + F_2 \left(L_1 + L_2\right) + F_3 \left(L_1 + L_2 + L_3\right) = ۶۰۳$$

$$\Rightarrow \text{زمان انتظار} = \frac{۶۰۳}{۴۲} \approx ۱۴/۳۵$$

پس ترتیب غیر نزولی L_i ها، بهینه نیست.

ب) نشان دهید ذخیره سازی فایل ها به ترتیب غیر صعودی F_i، لزوماً زمان بازیابی را حداقل نمی‌کند.

پاسخ :

سه فایل به طول های ۱۰۰ و ۱۰۰۰ و ۱ و به تعداد ارزیابی ۱و ۲و ۱۰۰ را در نظر بگیریم. اگر قرار دهیم:

$$L_1=1 \qquad L_2=1000 \qquad L_3=100$$
$$F_1=100 \qquad F_2=2 \qquad F_3=1$$

F_i ها غیر صعودی هستند و داریم:

$$\sum_{i=1}^{3} F_i = 103 \quad , \quad F_1L_1 + F_2(L_1+L_2) + F_3(L_1+L_2+L_3) = 3203$$

$$\Rightarrow \text{زمان انتظار} = \frac{3203}{103} \approx 31$$

اگر قرار دهیم

$$L_1=1 \qquad L_2=100 \qquad L_3=1000$$
$$F_1=100 \qquad F_2=1 \qquad F_3=2$$

داریم:

$$\sum_{i=1}^{3} F_i = 103 \quad , \quad F_1L_1 + F_2(L_1+L_2) + F_3(L_1+L_2+L_3) = 2403$$

$$\Rightarrow \text{زمان انتظار} = \frac{2403}{103} \approx 23$$

پس ترتیب غیر صعودی F_i ها، بهینه نیست.

ج) نشان دهید زمان بازیابی مینیمم است هر گاه فایل ها به ترتیب غیر صعودی F_i/L_i قرار بگیرند.

پاسخ :

برای اثبات دقیق این حکم باید از یک سری روابط و نامساوی های ریاضی استفاده کنیم که در سطح این کتاب نیست. ولی برای توجیح آن می توان حکم را برای $n=2$, $n=3$ بررسی کرد. (سعی کنید حکم را با استقراء ثابت کنید تا به مشکلات اثبات پی ببرید). ما برای $n=2$ حکم را بررسی می کنیم:

فرض می کنیم $\frac{F_1}{L_1} \le \frac{F_2}{L_2}$ حال باید نشان دهیم که این حالت از حالت دیگر بهتر است، یعنی

$$\frac{F_1 L_1 (L_1 + L_2)}{F_1 + F_2} \le \frac{F_1 L_1 + F_1 (L_1 + L_2)}{F_1 + F_2}$$

در واقع باید ثابت کرد $F_2 L_1 \le F_1 L_2$ که با استفاده از فرض، بدیهی است.

(دقت شود که برای $n = 3$ شش حالت ممکن است و حالت مطلوب را باید با پنج حالت دیگر مقایسه کرد.)

تمرینات فصل ششم

۱- پرانتز گذاری بهینه یک ضرب زنجیره ای ماتریس ها را پیدا کنید به طوری که توالی ابعاد آن < ۵, ۱۰, ۳, ۱۲, ۵, ۵۰, ۶ > باشد.

پاسخ :

داریم

$r_0 = 5 \quad r_1 = 10 \quad r_2 = 3 \quad r_3 = 12 \quad r_4 = 5 \quad r_5 = 50 \quad r_6 = 6$

مرحله اول:

$$m_{12} = \min\{m_{11} + m_{22} + r_0\ r_1\ r_2\} = 150$$
$$m_{23} = \min\{m_{22} + m_{33} + r_1\ r_2\ r_3\} = 360$$
$$m_{34} = \min\{m_{33} + m_{44} + r_2\ r_3\ r_4\} = 180$$
$$m_{45} = \min\{m_{44} + m_{55} + r_3\ r_4\ r_5\} = 3000$$
$$m_{56} = \min\{m_{55} + m_{66} + r_4\ r_5\ r_6\} = 1500$$

مرحله دوم

$$m_{13} = \min\{m_{11} + m_{22} + r_0 r_1 r_3\ ,\ m_{12} + m_{33} + r_0 r_2 r_3\}$$
$$= \min\{600\ ,\ 150 + 180\} = 330$$
$$m_{24} = \min\{m_{22} + m_{33} + r_1 r_2 r_4\ ,\ m_{23} + m_{44} + r_1 r_3 r_4\}$$
$$= \min\{150\ ,\ 360 + 600\} = 150$$
$$m_{35} = \min\{m_{33} + m_{44} + r_2 r_3 r_5\ ,\ m_{34} + m_{55} + r_2 r_4 r_5\}$$
$$= \min\{1800\ ,\ 180 + 750\} = 930$$
$$m_{46} = \min\{m_{44} + m_{55} + r_2 r_4 r_6\ ,\ m_{45} + m_{55} + r_3 r_5 r_6\}$$
$$= \min\{360\ ,\ 3000 + 3600\} = 360$$

مرحله سوم

$$m_{14} = \min\{m_{11} + m_{24} + r_0 r_1 r_4\ ,\ m_{12} + m_{34} + r_0 r_2 r_4\ ,\ m_{13} + m_{44} + r_0 r_3 r_4\}$$
$$= \min\{150 + 600, 150 + 180 + 75\ ,\ 330 + 300\} = 405$$
$$m_{25} = \min\{m_{22} + m_{35} + r_1 r_2 r_5\ ,\ m_{23} + r_{45} + r_1 r_3 r_5\ ,\ m_{24} + m_{55} + r_1 r_4 r_5\}$$
$$= \min\{930 + 1500, 360 + 3000 + 6000, 150 + 2500\} = 2430$$
$$m_{36} = \min\{m_{33} + m_{46} + r_2 r_3 r_6\ ,\ m_{34} + m_{56} + r_2 r_4 r_6\ ,\ m_{35} + m_{66} + r_2 r_5 r_6\}$$
$$= \min\{360 + 216, 180 + 1500 + 90\ ,\ 930 + 900\} = 576$$

مرحله چهارم

$$m_{15} = \min\{m_{11} + m_{25} + r_0 r_1 r_5 \ , \ m_{12} + m_{35} + r_0 r_2 r_5 \ , \ m_{13} + m_{45} + r_0 r_3 r_5 \ , \ m_{14} + m_{55} + r_0 r_4 r_5\}$$
$$= \min\{2430 + 2500, 150 + 930 + 750, 330 + 3000 + 3000, 405 + 1250\} = 1655$$

$$m_{26} = \min\{m_{22} + m_{36} + r_1 r_2 r_6 \ , \ m_{23} + m_{46} + r_1 r_3 r_6 \ , \ m_{24} + m_{56} + r_1 r_4 r_6 \ , \ m_{25} + m_{66} + r_1 r_5 r_6\}$$
$$= \min\{576 + 180, 360 + 360 + 720, 150 + 1500 + 300, 2430 + 3000\} = 756$$

مرحله پنجم

$$m_{16} = \min\{m_{11} + m_{26} + r_0 r_1 r_6 \ , \ m_{12} + m_{36} + r_0 r_2 r_6 \ , \ m_{13} + m_{46} + r_0 r_3 r_6 \ , \ m_{14} + m_{56} + r_0 r_4 r_6 \ , \ m_{15} + m_{66} + r_0 r_5 r_6\} = 1056$$

پس جواب نهایی ۱۰۵۶ است.

۲- نشان دهید که پرانتز گذاری کامل یک عبارت n عنصری، دقیقاً $n-1$ جفت پرانتز دارد.

پاسخ :

با استقراء حکم را ثابت می کنیم.

برای $n = 3$ حکم بر قرار است چون دو جفت پرانتز لازم داریم مثل $(A(BC))$. فرض کنیم حکم برای $n=k$ برقرار باشد.

حال اگر $n = k+1$ یعنی $k+1$ ماتریس داریم. k تای اول را جدا می کنی که طبق فرض برای آن به $k-1$ جفت پرانتز نیاز داریم. برای ضرب کردن حاصل ضرب آن k ماتریس با ماتریس $k+1$ ام هم یک جفت پرانتز نیاز داریم. پس در کل $k-1+1$ جفت ماتریس نیاز داریم، که این موضوع حکم را ثابت می کند.

۳- یک مسأله متفاوت ضرب زنجیره ای ماتریس ها را در نظر بگیرید که در آن هدف، پرانتز گذاری توالی ماتریس ها به منظور ماکزیمم کردن تعداد ضرب ها به جای مینیمم آنها باشد. آیا این مسأله زیرساختار بهینه ارائه می کند؟

پاسخ :

خیر. زیرا این مسئله در اصل بهینگی صدق نمی کند.

۴- طولانی ترین زیر رشته مشترک مربوط به $<1,0,0,1,0,1,0,1>$, $<0,1,0,1,1,0,1,1,0>$ را تعیین کنید.

پاسخ :

با اجرای الگوریتم بیان شده در کتاب، طولانی ترین زیر رشتهٔ مشترک دو رشتهٔ داده شده عبارت است از ۰۱۰۱.

۵- نشان دهید چطور می‌توان یک طولانی‌ترین رشته مشترک از روی جدول کامل شده C و رشته‌های اولیه $X =< x_1, x_2, ..., x_m >$ و $Y =< y_1, y_2, ..., y_m >$ را در زمان $O(m+n)$ و بدون استفاده از جدول b بازسازی نمود .

پاسخ :

۶- الگوریتمی با زمان $O(n^2)$ ارائه دهید که طولانی ترین زیر رشته یکنواخت صعودی یک رشته n عددی را پیدا کند.

پاسخ :

برای فهم راحت تر، الگوریتم را به زبان فارسی و مرحله به مرحله می نویسیم. فرض کنیم آرایهٔ مفروض $a[\,]$ باشد و طول آن n باشد.

۱) قرار دهیم $i \leftarrow 1$

۲) قرار دهیم $b^i[1] \leftarrow a[i]$ (در اینجا $b^i[\,]$ یک آرایه است)

۳) قرار دهیم $j \leftarrow i$, $k_i \leftarrow 1$

۴) اگر $a[j+1] \geq a[j]$ آنگاه

{

۵) قرار دهیم $b^i[1+k_i] \leftarrow a[j+1]$

۶) k_i را به k_i+۱ تبدیل می کنیم.

۷) j را به j+۱ تبدیل می کنیم.

۸) اگر $j<n$، آنگاه به خط ۴ بر می گردیم.

}

۹) i را به i+۱ تبدیل می کنیم.

۱۰) اگر $i<n$، آنگاه به خط ۲ بر می گردیم.

۱۱) بزرگترین عدد بین $k_1, k_2, ..., k_n$ را پیدا می کنیم، و فرض کنیم k_p باشد.

۱۲) آرایهٔ $b^p[\,]$ را چاپ می کنیم.

در واقع منطق الگوریتم چنین است که برای هر $1 \leq i \leq n$ طولانی ترین دنبالهٔ صعودیی که از $a[i]$ شروع می شود را در آرایهٔ $b^i[\,]$ ذخیره کرده و طول آن را k_i می‌نامیم . سپس برای بزرگترین k_i ، آرایه‌ی $b^i[\,]$ را چاپ می کنیم.

برای محاسبهٔ پیچیدگی الگوریتم فوق، دیده می شود که از خط ۱ الی خط ۱۰، دو حلقهٔ تودوتو که حد بالای هر دو n است. خطوط ۱۱ و ۱۲ هم از مرتبهٔ $O(n)$ هستند پس کل الگوریتم از مرتبهٔ $O(n^2)$ است.

۷- الگوریتمی با زمان $O(nLog\ n)$ برای پیدا کردن طولانی ترین زیر رشته یکنواخت صعودی یک رشته n عددی، ارائه دهید.

پاسخ :

آرایه را x می نامیم. با زمان $O(nlogn)$ آنرا به صورت صعودی مرتب می کنیم و مرتب شدهٔ آنرا y می نامیم. نهایتاً با زمان $O(n+n)$ (برابر است با $O(n)$) طبق الگوریتم صفحهٔ ۲۴۲ کتاب، طولانی ترین زیر رشتهٔ مشترک بین آرایه های x و y را پیدا می کنیم که همان حل مسئله است.

راه دوم چنین است: فرض کنیم.

$T[i]$ = زیر رشته صعودی به طول k که به a_k (کوچکترین عدد ممکن) ختم میشود و از تمام زیر رشته های صعودی به طول i با عناصر $a_1, a_2, \ldots, a_i$ ($k \le i$) استخراج شده است.

$P[k]$= کپی برداری از زیر رشته قبلی LIS که به a_k ختم می یابد.

L= متغیری که طول پیدا شده را تاکنون نگه می دارد.

```
Void LIS (A[], n)
 {
   L=0 , P[0]=0 , T[0]=0
   for (i=0 to n , i++)
 {
```

با استفاده از جستجوی دو رویی طولانی ترین زیررشته $j < L$ را پیدا کن به طوری که $A[T[j]] < A[j]$ باشد و یا اگر مقداری وجود ندارد آنگاه $j = 0$ قرار بده

```
 }
P[i]=T[j]
if (j==L || A[i]<A[T[j+1]])
  { T[j+1]=i
      L=max(L , j+1)
      }
}
```

پیچیدگی جستجوی دودویی $O(Logn)$ است و چون حلقه n بار تکرار می شود، کلاً $O(n\log_n)$ است.

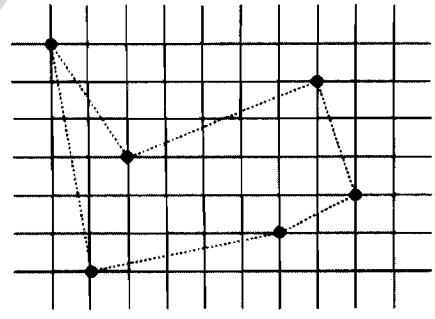
مسائل تکمیلی فصل ششم

۱- مسأله اقلیدسی مسیر *Bitonic* فروشنده دوره گرد

مسأله اقلیدسی فروشنده دوره گرد، مسأله مشخص کردن کوتاهترین مسیر بسته می باشد که در یک صفحه، مجموعه ای از n نقطه داده شده را به هم متصل می کند. شکل (الف) حل مسأله ای با ۷ نقطه را نشان می دهد. مسأله فروشنده دوره گرد، جزء مسائل *NP* کامل است و بنابراین حل آن نیاز به زمان نمایی دارد.

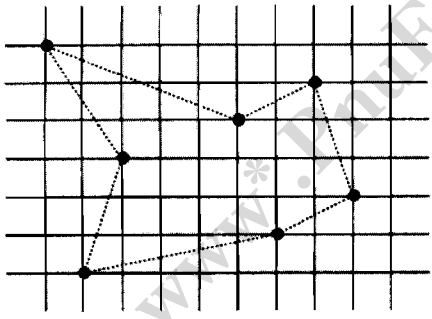
J.L.Bentley پیشنهاد کرده است که برای ساده سازی مسأله، تمرکزمان را به مسیرهای *Bitonic* معطوف می کنیم. به عبارت دیگر مسیرهایی که از سمت چپ ترین نقطه شروع می شوند، و به سمت راست‌ترین نقطه رفته و سپس از راست به طرف چپ بازمی گردند. شکل (ب) کوتاهترین مسیر *Bitonic* را برای همان ۷ نقطه نشان می دهد. در این حالت می توان برای آن الگوریتمی با زمان چند جمله ای نوشت.

الگوریتمی با زمان $O(n^2)$ برای تعیین مسیر *Bitonic* بهینه شرح دهید. فرض کنید هیچ دو نقطه ای دارای مختصات x مشابه نیستند.



(ب)

الف) کوتاهترین مسیر بسته با طول تقریباً ۸۹,۲۴ این مسیر bitonic نمی‌باشد.



(الف)

ب) کوتاهترین مسیر bitonic. طول تقریباً ۵۸,۲۵ است.

پاسخ :

برای فهم بهتر، الگوریتم را به فارسی و مرحله به مرحله می نویسیم.

الف) نقاط داده شده را a_1 , a_2,...,a_n می نامیم.

ب) نقاط را طوری تغییر آرایش می دهیم که طولهایشان به طور صعودی باشند و این ترتیب آرایش را b_1,...b_n می نامیم و قرار می دهیم $b_i=(x_i,y_i)$.

ج) قرار می دهیم $i=1$ (تا وقتی که $i<n$)

د) فرض کنیم j اندیس کمترین y_{i+1},...y_n باشد، یعنی

$$y_j = \min\{ y_{i+1}, \ldots, y_n \}$$

و) b_i را به b_j وصل می کنیم.

هـ) i را مساوی j قرار می دهیم و بر می گردیم به خط د

ی) b_j را به b_n وصل می کنیم. (تا اینجا از سمت چپ ترین نقطه شروع کرده و هر نقطه را به راست ترین نقطهٔ نسبت به خودش وصل کرده ایم.)

م) b_k های باقیمانده را از آخر به اول به هم وصل می کنیم. (این دستور، خود شامل چندین خط دستور است.)

دقت: چون اجرای دستور د خود شامل حلقه است، پس پیچیدگی الگوریتم از مرتبهٔ $\theta(n^2)$ است.

۲- نشان دهید که ضرب دو ماتریس B,A به ابعاد $p\times q$, $q\times r$ نیاز به $p\times q\times r$ ضرب عددی دارد.

پاسخ :

فرض کنیم $A=\lfloor a_{ij}\rfloor_{p\times q}$, $B=\lfloor b_{ij}\rfloor_{q\times r}$, $AB=C=\lfloor C_{ij}\rfloor_{p\times r}$ می دانیم:

$$C_{ij}=\sum_{k=1}^{q} a_{ik}b_{kj}$$

پس محاسبه هر C_{ij}، نیاز به q عدد ضرب دارد. چون ماتریس C به تعداد $p\times r$ تا درایه دارد، پس در کل به $p\times q\times r$ ضرب نیاز است.

۳- فرض کنید $X=aabaababaa$, $Y=babaabab$ و ارزش اعمال درج و حذف یک واحد و ارزش عمل تغییر ۲ واحد می باشد. ارزش بهینه تبدیل X به Y را پیدا کنید.

پاسخ :

ابتدا طولانی ترین زیر رشته مشترک y,x را پیدا می کنیم که عبارت است از $baabab$ در نتیجه در x دو aی آخر را حذف می کنیم و اولین a را تبدیل به b می کنیم. پس ارزش بهینه برابر ۴ است. (می توان از الگوریتم ۸-۶ نیز استفاده کرد).

۴- فرض کنید $T(n)$ تعداد روش های مختلف پرانتز گذاری حاصلضرب n ماتریس باشد آنگاه $T(n)=\sum_{i=1}^{n-1}T(i)T(n-i)$ ثابت کنید که:

$$T(n)=\frac{1}{n}\binom{2n-2}{n-1}$$

پاسخ :

فرض کنیم می خواهیم $A_1A_2A_3....A_n$ را پرانتز گذاری کنیم.

الف) می توان $A_۲A_۳....A_n$ را پرانتز گذاری کرد و حاصل را در $A_۱$ ضرب کرد. تعداد حالات این مدل برابر $T(۱)T(n-۱)$ می باشد.

ب) می توان $A_۱A_۲$ را پرانتز گذاری کرد و حاصل را در $A_۳A_۴....A_n$ ضرب کرد. تعداد حالات این مدل برابر $T(۲)T(n-۲)$ می باشد.

ج) می توان $A_۱A_۲A_۳$ را پرانتز گذاری کرد و حاصل را در $A_۴A_۵....A_n$ ضرب کرد. تعداد حالات این مدل برابر $T(۳)T(n-۳)$ می باشد.

با ادامه روش بالا، دیده می شود که:

$$T(n)=T(۱)T(n-۱)+T(۲)T(n-۲)+...+T(n-۱)T(۱)=\sum_{i=۱}^{n-۱}T(i)\,T(n-i)$$

این یک رابطه بازگشتی غیر خطی است که محاسبه آن، کمی پیچیده است. برای سادگی در نوشتن، $T(n+۱)$ را با b_n نمایش می دهیم. فرض کنیم تابع مولد آن به صورت زیر باشد:

$$f(x)=\sum_{n=۰}^{\infty}b_nx^n$$

پس:

$$\big(f(x)-b_۰\big)=x\sum_{n=۰}^{\infty}\big(b_۰b_n+b_۱b_{n-۱}+...+b_nb_۰\big)x^n=x\big[f(x)\big]^۲\Rightarrow$$

$$x\big[f(x)\big]^۲-f(x)+b_۰=۰$$

چون $b_۰=۱$ داریم:

$$f(x)=\frac{۱\pm\sqrt{۱-۴x}}{۲x}$$

از بسط مک لورن داریم:

$$\sqrt{۱-۴x}=(۱-۴x)^{۱/۲}=\binom{۱/۲}{۰}+\binom{۱/۲}{۱}(-۴x)+\binom{۱/۲}{۲}(-۴x)^۲+...$$

که ضریب x^n در آن برابر است با:

$$\binom{۱/۲}{n}(-۴)^n=\frac{\left(\frac{۱}{۲}\right)\left(\frac{۱}{۲}-۱\right)\left(\frac{۱}{۲}-۲\right)....\left(\frac{۱}{۲}-n+۱\right)}{n!}(-۴)^n=$$

$$(-۱)^{n-۱}\frac{\left(\frac{۱}{۲}\right)\left(\frac{۳}{۲}\right)....\left(\frac{۲n-۳}{۲}\right)}{n!}(-۱)^n=$$

$$\frac{(-1)2^n(1)(3)....(2n-3)}{n!}=\frac{(-1)^n 2^n(n!)\,(1)(3)....(2n-3)(2n-1)}{(n!)(n!)(2n-1)}$$

$$=\frac{(-1)(2)(4)....(2n)(1)(3)....(2n-1)}{(2n-1)(n!)(n!)}=\frac{(-1)}{2n-1}\binom{2n}{n}$$

$$\Rightarrow f(x)=\frac{1}{2x}\left[1-\left[1-\sum_{n=1}^{\infty}\frac{1}{(2n-1)}\binom{2n}{n}x^n\right]\right]$$

پس b_n برابر ضرب x^{n+1} در سری زیر است:

$$\sum_{n=1}^{\infty}\frac{1}{(2n-1)}\binom{2n}{n}x^n$$

یعنی:

$$b_n=\frac{1}{2}\left[\frac{1}{2(n+1)-1}\right]\binom{2(n+1)}{n+1}=\frac{(2n)!}{(n+1)!(n!)}=\frac{1}{n+1}\binom{2n}{n}$$

پس :

$$T(n)=b_{n-1}=\frac{1}{(n-1)+1}\binom{2(n-1)}{n-1}=\frac{1}{n}\binom{2n-2}{n-1}$$

۵- ثابت کنید تعداد فراخوانی های بازگشتی در روال محاسبه ترکیب k مؤلفه برابر با $2\binom{n}{k}-2$ است.

پاسخ :

تعداد فراخوانی های بازگشتی برای محاسبه k مؤلفه از n از مؤلفه را با $M(n,k)$ نشان می‌دهیم. می خواهیم ثابت کنیم:

$$M(n,k)=2\binom{n}{k}-2$$

این حکم را با استقراء روی n ثابت می کنیم:

برای $n=1$ (چون اجباراً $k=1$) حکم بدیهی است.

فرض کنیم برای $n-1$ برقرار باشد. حال برای n با توجه به رابطه (۴-۶) در صفحه ۲۰۳ از کتاب باید دو فراخوانی $M(n-1,k)$, $M(n-1,k-1)$ را انجام داد.

پس داریم:

$$M(n,k)=M(n-1,k)+M(n-1,k-1)+2=$$
$$2\binom{n-1}{k}-2+2\binom{n-1}{k-1}-2+2=2\binom{n}{k}-2$$

۶- الگوریتم وارشال در الگوریتم فلوید کوتاهترین مسیرها بین هر زوج رأس یافت می گردد. در الگوریتم وارشال (*War Shalls algorithm*) وجود یا عدم وجود مسیر بین هر زوج راس مورد نظر مشخص می شود. یعنی در این الگوریتم به ازای کلیه راس ها اگر یک مسیری میان آنها وجود داشته باشد در ماتریس مجاورت یا *D, Ture* قرار داده می شود. یعنی *D(i,J)=ture* وگرنه *False* با اعمال تغییرات لازم در الگوریتم فلوید، الگوریتم وارشال را بنویسید.

پاسخ :

کافیست عمل منطقی *OR* را به جای *min* و عمل منطقی *AND* را به جای + قرار دهیم. ماتریس *W* در اینجا چنین است که هر کجا یالی بود عدد ۱ می گذاریم و کجا یالی نبود، عدد صفر می‌گذاریم.

```
Void  varshal (n , W [ ][ ] , D [ ][ ])
  {
    D ← W
    for (k = 1 to n ; k++)
      for (i = 1 to n ; i++)
        for (j = 1 to n ; j++)
          D[i][j] = ((D[i][j] OR (D[i][k] AND D[k][j])))
```

در واقع هر کجا مسیری باشد، با ۱ نشان می دهد و هر کجا مسیری نباشد، با صفر نشان می‌دهد.

۷- ثابت کنید که تعداد فراخوانیها برای محاسبه *P(n,n)* در تابع *world series* برابر $2\binom{2n}{n}-2$ است.

پاسخ :

راه اول: از استقرا کمک می گیریم و دقیقاً مانند تمرین ۵، حکم را ثابت می کنیم.

راه دوم: تعداد فراخوانی های مورد نیاز برای محاسبه $P(n,n)$ برابر است با تعداد فراخوانی های مورد نیاز برای محاسبه ترکیب n از $n+n$. یعنی باید تعداد فراخوانی های لازم برای محاسبه $\binom{۲n}{n}$ را محاسبه کنیم، که طبق مسئله ۵ برابر خواهد بود با $۲\binom{۲n}{n}-۲$.

۸- ثابت کنید که مرتبه زمانی دقیق روال *world series* برای محاسبه $P(n,n)$ برابر $\theta\left(۴^n/\sqrt{n}\right)$ است.

پاسخ :

با توجه به تمرین ۷، زمان دقیق از مرتبه $\theta\left(\binom{۲n}{n}\right)$ است. ولی از فرمول استرلینگ می‌دانیم $n!\approx\sqrt{۲\pi n}\left(\frac{n}{e}\right)^n$: پس :

$$\binom{۲n}{n}=\frac{(۲n)!}{n!\,n!}\approx\frac{\sqrt{۴\pi n}(\frac{۲n}{e})^{۲n}}{۲\pi n(\frac{n}{e})^{۲n}}=\frac{۴^n}{\sqrt{\pi n}}.$$

پس زمان دقیق از مرتبه $\theta\left(\frac{۴^n}{\sqrt{\pi n}}\right)$ می باشد.

۹- سری اعداد کاتالان به صورت زیر تعریف می شود:

$$\begin{cases}T(۱)=۱\\ T(n)=\sum_{i=۱}^{n-۱}T(i)T(n-i)\end{cases}$$

نشان دهید که تعداد روش های که می توان با رسم قطره های غیر متقاطع یک n ضلعی محدب را به n-۲ مثلث تبدیل کرد برابر n-۱ بعنی عدد کاتالان، یعنی $T(n)=\sum_{i=۱}^{n-۲}T(i)T(n-i-۱)$ است.

پاسخ :

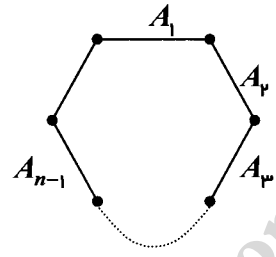
این سؤال غلط چاپی دارد. شکل صحیح آن چنین است.

نشان دهید که تعداد روش هایی که می توان با رسم قطرهای غیر متقاطع یک n ضلعی محدب را به $n-۲$ مثلث تبدیل کرد ، برابر است با $T(n-۱)$، که T یعنی عدد کاتالان، یعنی

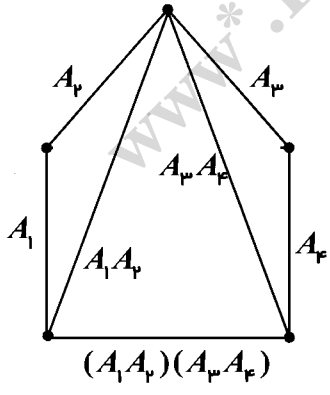
$$T(n-۱)=\sum_{i=۱}^{n-۲}T(i)T(n-i-۱)$$

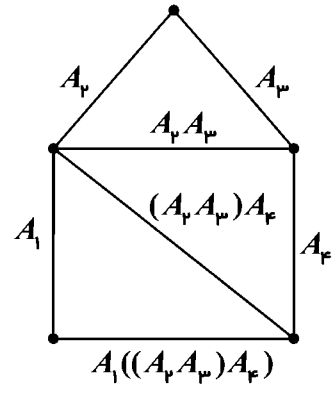
برای اثبات این مسئله، کافیست ثابت کنیم تعداد این مثلث بندیهای مطلوب، برابر است با تعداد پرانتزگذاری های $n-1$ ماتریس $A_1A_2A_3....A_{n-1}$ به منظور ضرب آنها، چون در آن صورت طبق مسئله تکمیلی ۴ از همین فصل، حکم ثابت است.

فرض کنیم می خواهیم ضرب ماتریس های $A_1A_2....A_{n-1}$ را به همین ترتیبی که قرار گرفته اند انجام دهیم، یک n ضلعی محدب رسم می کنیم و ضلعهای آن را به ترتیب A_1 , A_2 , A_3 ,..... , A_{n-1} نامگذاری می کنیم و ضلع آخر را نامگذاری نمی کنیم.



در پرانتز گذاری برای ضرب، ابتدا دو ماتریس مجاور مانند A_kA_{k+1} را پرانتز گذاری می‌کنیم. متناظر با آن مثلثی را رسم می کنیم که اضلاع آن A_k , A_{k+1} و یک قطر باشد، این قطر اضافه شده را A_kA_{k+1} می نامیم. به همین روال ادامه می دهیم تا نهایتاً ضلع بدون نام، $A_1A_2...A_n$ با پرانتز گذاری های متناظر آن نامگذاری خواهد شد. مثلاً برای پنج ضلعی دو حالت را به عنوان نمونه رسم می کنیم.



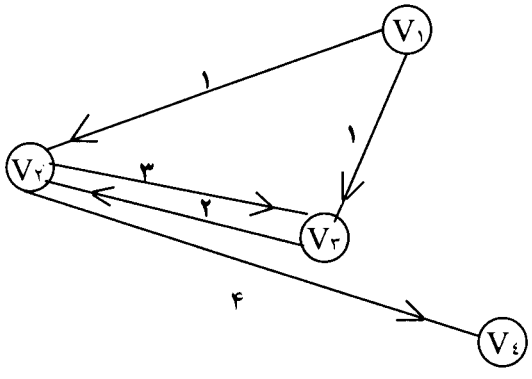


۱۰- یک مسأله بهینه سازی پیدا کنید که در آن اصل بهینگی صدق نکند و بنابراین حل بهینه با استفاده از برنامه نویسی پویا قابل حصول نباشد، پاسخ خود را توضیح دهید.

پاسخ :

پیدا کردن طولانی ترین مسیر بین دو رأس از یک گراف وزن دار، در اصل بهینگی صدق نمی‌کند. برای روشن شدن این مطلب، به مثال زیر توجه می کنیم.

در گراف زیر طولانی ترین مسیر از V_1 به V_4 به صورت $V_1 \to V_3 \to V_2 \to V_4$ می باشد، اما طولانی ترین مسیر از V_1 به V_3 به صورت $V_1 \to V_3$ نمی باشد بلکه به صورت $V_1 \to V_2 \to V_3$ می باشد. پس حل یک نمونه از مسئله، شامل حل هر زیر نمونه ای از آن نیست.



۱۱- تساوی زیر را اثبات کنید:

$$\sum_{digonal=1}^{n-1} \left[(n - diagonal) \times diagonal \right] = \frac{n(n-1)(n+1)}{6}$$

پاسخ :

diagonal را با d نمایش می دهیم.

$$\sum_{d=1}^{n-1} (n-d)d = (n-1) + (n-2)2 + (n-3)3 + \ldots + (n-(n-1))(n-1)$$

$$= n - 1 + 2n - 4 + 3n - 9 + \ldots + (n-1)n - (n-1)^2$$

$$= n + 2n + 3n + \ldots + (n-1)n - (1 + 9 + \ldots + (n-1)^2)$$

$$= n(1 + 2 + \ldots + (n-1)) - \left(1^2 + 2^2 + 3^2 + \ldots + (n-1)^2\right)$$

$$= n\frac{(n-1)n}{2} - \frac{(n-1)n\left(2(n-1)+1\right)}{6}$$

$$= \frac{n(n-1)\left(3n-2n+1\right)}{6} = \frac{n(n-1)\,(n+1)}{6}$$

۱۲- الگوریتمی کارآمد بنویسید که ترتیب بهینه را برای ضرب n ماتریس $A_1 \times A_2 \times \ldots \times A_n$ پیدا کند که در آن ابعاد هر یک از ماتریس ها 1×1 , $1 \times d$, $d \times 1$, $d \times d$ است و d یک عدد صحیح مثبت است. الگوریتم خود را تحلیل کنید.

پاسخ :

به عنوان نمونه، برنامه کامل این مسئله را می نویسیم:

```
Void  matrix – chain – order (int r)
{
    int  i=۱ ,  j=۱, L=۱ ;
    for (i=۱ , i<=n ; i++)
      { if  ( r [i-۱]==r[i]==۱)
            P[j++]=i ;
        if (r [i-۱]==۱&& r[i]==d)
            q [k++]=i ;
        if (r[i-۱]==d && r [i]==۱)
            s[L++]=i ;
      }
    P[j]=S[۱]; j++ ;
    for (i=۱; i<=n ; i++)
        if (r[i-۱]==r[i]==d)
            P [j++]=i ;
    for (i=۱ ; i<k ; i++)
      {d  [k++]=q[i];
       d[k++]=S[i+۱] ;
}
```

واضح است که پیچیدگی این الگوریتم برابر $\theta(n)$ است.

۱۳- چند درخت جستجوی دودویی می توان با شش کلید متمایز درست کرد.

پاسخ :

تعداد درختهای جستجوی دودویی با n کلید متمایز، توسط فرمول زیر محاسبه می شود.

$$\frac{۱}{n+۱}\binom{۲n}{n}$$

اثبات فرمول فوق در تمرین ۱۶ همین بخش، خواهد آمد. با توجه به فرمول فوق تعداد درختهای جستجوی دودویی با ۶ کلید متمایز برابر است با:

$$\frac{۱}{۷}\binom{۱۲}{۶}=۱۳۲$$

۱۴- یک مدار بهینه برای گراف جهت دار و موزون نشان داده شده توسط ماتریس W پیدا کنید. عملیات را قدم به قدم نشان دهید.

$$W=\begin{bmatrix} ۰ & ۸ & ۱۳ & ۱۸ & ۲۰ \\ ۳ & ۰ & ۷ & ۸ & ۱۰ \\ ۴ & ۱۱ & ۰ & ۱۰ & ۷ \\ ۶ & ۶ & ۷ & ۰ & ۱۱ \\ ۱۰ & ۶ & ۲ & ۱ & ۰ \end{bmatrix}$$

پاسخ :

با توجه به مثال ۵-۶ در صفحه ۲۰۲ کتاب، جواب واضح است.

۱۵- تعداد ترتیب های ممکن برای ضرب n ماتریس $A_1, A_2, \ldots, A_N$ را تعیین کنید.

پاسخ :

مفهوم سؤال مشخص نیست، اگر واقعاً منظور سؤال، تعداد ترتیبها باشد، چون در رابطه با تعداد سطرها و ستونهای ماتریسهای $A_1, A_2, \ldots, A_n$ چیزی نمی دانیم، پس نمی توان هیچ قضاوتی کرد. ولی اگر همه ماتریسها را مربعی و هم مرتبه فرض کنیم آنگاه تعداد ترتیب‌های مختلف برابر است با $n!$.

۱۶- نشان دهید که تعداد درختهای جستجوی دودویی با n کلید متمایز توسط فرمول زیر محاسبه می‌شود:

$$\frac{۱}{n+۱}\binom{۲n}{n}$$

پاسخ :

به اختصار، دو راه حل را بیان می کنیم و جزئیات بیشتر را به دانشجویان وامی گذاریم.

راه اول: چون n رأس داریم و هر رأس دو فرزند می تواند داشته باشد، پس در کل به تعداد $۲n$ تا جای داریم. در این $۲n$ تا جای، باید n کلید را قرار دهیم، پس به تعداد $\binom{۲n}{n}$ حالت می توان کلیدها را قرار داد. ولی به سادگی قابل رؤیت است که از هر $n+۱$ مدل چیدمان تنها

یکی از آنها خاصیت درختهای جستجوی دودویی را دارد. پس تعداد کل درختهای جستجوی دودویی برابر است با: $\frac{1}{n+1}\binom{2n}{n}$.

راه دوم : همانند آنکه در تمرین ۹، تعداد مثلث بندیها را با تعداد پرانتز گذاری ها، معادل کردیم، می توان نشان داد که تعداد درختهای جستجوی دودویی با n کلید برابر است با تعداد پرانتز گذاری ها برای $n+1$ ماتریس. پس با توجه به فرمول ثابت شده در مسئله ۹، تعداد درختهای جستجوی دودویی برابر است با:

$$T(n+1)=\frac{1}{n+1}\binom{2(n+1)-2}{(n+1)-1}=\frac{1}{n+1}\binom{2n}{n}$$

۱۷- آیا می‌دانید یک الگوریتم زمانی درجه دوم برای مساله درخت جستجوی بهینه بنویسید ؟

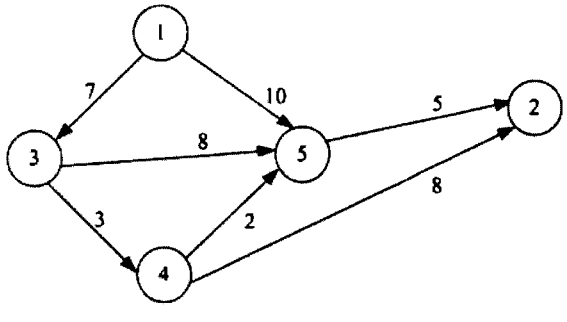
پاسخ :

۱۸- اصل بهینه سازی برای هر مسأله ای که جواب آن می تواند به عنوان یک توالی انتخاب به دست آید، برقرار نیست. دو مسأله که برای آنها اصل بهینه سازی برقرار نیست، پیدا کنید و علت برقرار نبودن این اصل را برای این مسائل بیان کنید.

پاسخ :

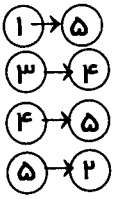
پیدا کردن طولانی ترین مسیر در یک گراف همبند، و پیدا کردن پرانتز گذاریی که بیشترین تعداد ضرب را داشته باشند، از جمله مسائلی هستند که در اصل بهینگی صدق نمی کند. برای توضیح بیشتر می توان به حل مسئله ۱۰ مراجعه کرد.

۱۹- در گراف زیر کوتاهترین مسیر بین گره ۱و۲ را پیدا کنید:



پاسخ :

این مسئله را هم از الگوریتم فلوید می توان حل کرد، هم از الگوریتم دیکسترا. چون الگوریتم دیکسترا سریعتر است، پس از آن استفاده می کنیم. از گره ۱ شروع می کنیم. ابتدا یال ۳ ← ۱ انتخاب می شود. در مراحل بعد، یالهای انتخاب شده به ترتیب برابرند با:



پس کوتاهترین مسیر از ۱ به ۲ برابر است با:

۲۰- الگوریتمی برای ساختن درخت جستجوی دودویی بهینه با ریشه های معلوم $r(i,J)$, $0 \le i \le J \le n$ بنویسید. نشان دهید که این کار می تواند در زمان $O(n)$ انجام شود.

پاسخ :

۲۱- نشان دهید فرمول بهینه بازگشتی زیر برای مسئله پرانتز گذاری ضرب n ماتریس به کار می رود.

$$m[i,J]=\begin{cases}0 & if \quad i=J \\ \min\{m[i,k]+m[k+1,J]+P_{i-1}P_KP_J\} & if \ i<J\end{cases}$$

پاسخ :

فرض کنیم در بین A_i , A_j اولین دسته بندی بعد از A_k واقع شود. پس می توان چنین تصور کرد (فرض می کنیم $i<j$) .

$$(A_i....A_k)\ (A_{k+1}....A_j)$$

پرانتز اول یک ماتریس $P_{i-1}\times P_k$ و پرانتز دوم یک ماتریس $P_k\times P_j$ است. پس برای ضرب این دو پرانتز به $P_{i-1}P_k\ P_j$ ضرب نیاز داریم. داخل پرانتز اول و دوم را نیز باید پرانتز گذاری کرد و در بهترین حالت به ترتیب باید $m[i,k]$, $m[k+1,j]$ ضرب انجام شود. پس در کل به تعداد $m[i,k]+m[k+1,j]+P_{i-1}P_kP_j$ ضرب نیاز است.

چون نمی دانیم بهترین موقعیت این دسته بندی کجاست، پس نمی دانیم k چند است؟ پس برای تمام k های بین j,i عدد فوق را محاسبه کرده و کمترین آنها را برمی گزینیم.

اگر $i=j$ بدیهی است که برای محاسبه حاصلضرب A_i تا A_j به هیچ ضربی نیاز نیست.

۲۲- حداقل ممکن تعداد ضرب چهار ماتریس

$D(۱۰\times۲۰)$, $C(۴\times۱۰)$, $B(۱\times۴۰)$, $A(۳۰\times۱)$

را به صورت $ABCD$ در هم ضرب می‌شوند را پیدا کنید. حداکثر تعداد ضرب چقدر خواهد بود؟

پاسخ :

ماتریس C باید ۱۰×۴۰ باشد تا ضرب $ABCD$ ممکن باشد.

با توجه به مسئله ۴ تعداد حالات ممکن برابر

$$\frac{۱}{۴}\binom{۶}{۳}=۵$$

می باشد.

همه حالتها را با تعداد ضربهایشان محاسبه می کنیم:

$((AB)(CD))$ $۳۰\times۴۰+۴۰\times۱۰\times۲۰+۳۰\times۴۰\times۲۰=۳۳۲۰۰$

$(((AB)C)D)$ $۳۰\times۴۰+۳۰\times۴۰\times۱۰+۳۰\times۱۰\times۲۰=۱۹۲۰۰$

$((A(BC))D)$ $۴۰\times۱۰+۳۰\times۱۰+۳۰\times۱۰\times۲۰=۷۱۰۰$

$(A(B(CD)))$ $۴۰\times۲۰\times۱۰+۴۰\times۲۰+۳۰\times۲۰=۹۴۰۰$

$(A((BC)D))$ $۴۰\times۱۰+۱۰\times۲۰+۳۰\times۲۰=۱۲۰۰$

پس کمترین تعداد ضرب ۱۲۰۰ و بیشترین تعداد ضرب ۳۳۲۰۰ می باشد.

۲۳- فرض کنید سه ماتریس زیر را داریم:

ماتریس	ابعاد
$N_۱$	$W\times X$
$N_۲$	$X\times y$
$N_۳$	$Y\times Z$

ثابت کنید اگر $\frac{۱}{x}+\frac{۱}{z}<\frac{۱}{W}+\frac{۱}{y}$ باشد ضرب $(N_۱N_۲)N_۳$ سریعتر از ضرب $N_۱(N_۲N_۳)$ انجام خواهد شد.

پاسخ :

برای محاسبه $(N_۱N_۲)N_۳$ به $wxy+wyz$ ضرب نیاز است.

برای محاسبه $N_۱(N_۲N_۳)$ به $xyz+wxz$ ضرب نیاز است.

می دانیم که $\frac{۱}{x}+\frac{۱}{z}<\frac{۱}{w}+\frac{۱}{y}$. با ضرب طرفین در $xyzw$ داریم:

$$yzw+xyw<xyz+xzw$$

پس حکم برقرار است.

تمرینات فصل هفتم

۱- در مسأله n وزیر برخی از جوابها انعکاسی یا چرخشی هستند. مثلاً در مسأله ۴ وزیر دو جواب زیر انعکاسی یکدیگر هستند. این گونه جوابها را هم ارز می نامیم.

		۱	
۲			
			۳
	۴		

	۱		
			۲
۳			
		۴	

الگوریتمی بنویسید که جوابهای انعکاسی را پیدا کند. مرتبه زمانی الگوریتم خود را تحلیل کنید.

پاسخ :

```
Void   queens   (int i)
 {
    int    j ;
    if  (promi sin g (i))
       if (i = = n)
       {cout << col [۱] , col [۲], ...., col [n];
           Sym queens  ( ) ;
           cout << S col[۱] , S col [۲], .... , S col [n];
        }
    else {
         if (i = =۱)
            k = n/۲ ;
         else
            k = n ;
          for (j =۱ ; j <= k  ; j++)
             {col  [i+۱]= j ;
                   queens (i+۱) ;
              }
       }
 }
```

راه حلی که در بالا ارائه شد، به صورت یک برنامهٔ کامل بیان شد. یک راه حل هم فقط به صورت اشاره، بیان خواهیم کرد. سعی کنید تابع این الگوریتم را نیز بنویسید.

اگر هر جواب را به صورت یک ماتریس در نظر بگیریم، و $A=\lfloor a_{ij} \rfloor$, $B=\lfloor b_{ij} \rfloor$ جواب انعکاسی باشند، آنگاه برای هر $1 \le i \ , \ j \le n$ داریم.

$$a_{ij} = b_{i(n+1-j)}$$

مثلاً برای مثال رسم شده در کتاب داریم:

$$a_{12} = b_{13} \quad , \quad a_{14} = b_{11} \quad , \quad a_{31} = b_{34} \quad , \quad a_{43} = b_{42}$$

البته درایه های صفر هم که به وضوح در رابطهٔ فوق صدق می کنند. با استفاده از رابطهٔ فوق پس از n^2 مقایسه می توان فهمید که آیا دو جواب، انعکاس یکدیگر هستند یا خیر. البته چون می‌دانیم در هر سطر تنها یک عدد ۱ قرار دارد، پس می‌توان رابطهٔ فوق را فقط برای یکها امتحان کرد و با این کار با n مقایسه به انعکاس بودن دو جواب پی برد.

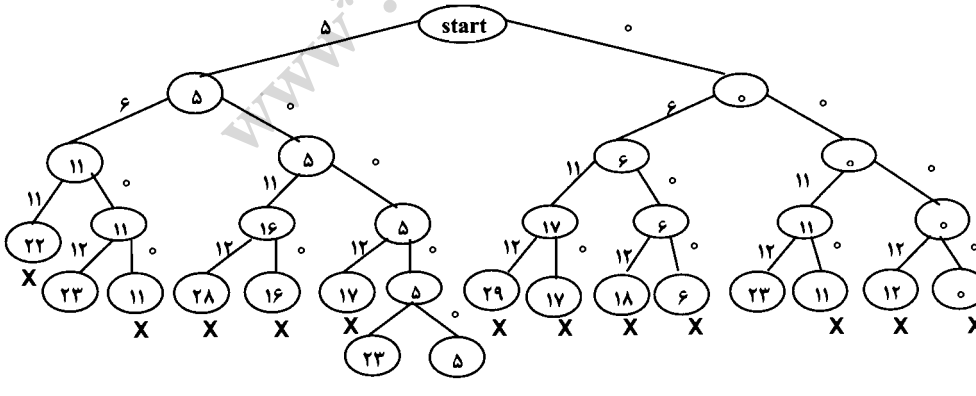
۲- با ترسیم درخت حل مسأله مجموع زیرمجموعه ها، برای یک مثال نمونه، تعداد دقیق دفعاتی را که عمل *backtrack* انجام خواهد شد، را شمارش کنید.

پاسخ :

فرض کنیم

$$n=5 \qquad w=23$$

$$w_1=5 \qquad w_2=6 \ , \ w_3=11 \ , \ w_4=12 \qquad w_5=18$$



۳- ثابت کنید که تعداد زیر مجموعه های یک مجموعه با n مؤلفه 2^n است.

پاسخ :

این مسئله راه حلهای گوناگونی دارد. به عنوان نمونه دو راه حل را بیان می کنیم.

الف) به کمک استقراء

برای $n=1$ واضح است چون مجموعهٔ $\{a\}$ فقط دو زیر مجموعهٔ تهی و $\{a\}$ را دارد.

فرض کنیم برای $n=k$ برقرار باشد.

حال اگر $n=k+1$ ، پس می توان فرض کرد $A=\{a_1, a_2, \ldots a_k . a_{k+1}\}$. هر زیر مجموعه از A یا شامل a_{K+1} هست یا نیست. اگر شامل a_{K+1} نباشد پس زیر مجموعه ای از $\{a_1, \ldots a_k\}$ است. که تعداد حالات آن طبق فرض استقراء 2^k است. اگر شامل a_{k+1} باشد، در کنار آن می‌تواند هر یک از 2^k زیر مجموعهٔ $\{a_1, \ldots a_k\}$ نیز قرار بگیرد. پس تعداد حالات کلی برابر می شود با $2^{k+1}=2\times 2^k=2^k+2^k$ و حکم ثابت است.

ب) می دانیم تعداد زیر مجموعه های m عضوی از یک مجموعهٔ n عضوی برابر است تعداد انتخابهای m عضو از n عضو، یعنی $\binom{n}{m}$ پس تعداد کل زیر مجموعه ها برابر است با:

$$\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\ldots+\binom{n}{n}$$

از طرفی هم طبق اتحاد دو جمله ای نیوتن، برای هر a,b داریم:

$$(a+b)^n=\binom{n}{0}a^n+\binom{n}{1}a^{n-1}b+\binom{n}{2}a^{n-2}b^2+\ldots.+\binom{n}{n}b^n$$

با قرار دادن $a=b=1$ در رابطهٔ فوق داریم:

$$2^n=\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\ldots+\binom{n}{n}$$

و حکم ثابت است.

۴- فرض کنید $W=\{5, 7, 10, 12, 15, 18, 20\}$, $S=35$ است. همه زیر مجموعه های ممکن W که مجموع آنها S می باشد، را پیدا کنید. قسمتی از درخت فضای حالت را رسم کنید.

پاسخ :

درخت آنرا مانند مسئلهٔ ۲ می توان رسم کرد. حالتهای ممکن عبارتند از :

۵+۱۲+۱۸=۳۵

۷+۱۰+۱۸=۳۵

۵+۱۰+۲۰=۳۵

۱۵+۱۸=۳۵

۵- تابع *mcoloring* یا رنگ آمیزی گراف را نوشته و آن را برای گراف های کامل با اندازه ۶، ۵، ۴، ۳، ۲=*n* اجرا کنید. تعداد رنگ ها را $k = n$, $k = n/2$ نیز محاسبه کنید. زمان محاسباتی را برای *n,k* به صورت جدول رسم کنید.

پاسخ :

تابع آن در صفحهٔ ۲۶۸ کتاب نوشته شده است. منظور این سؤال این است که تابع فوق را روی کامپیوتر برای چندین گراف مختلف اجرا کنید و با هم مقایسه کنید.

۶- *MAZE* (۱:*n*, ۱:*n*) یک آرایه دو بعدی با مقادیر صفر و یک است که یک مارپیچ را نشان می‌دهد. ۱ به معنای بسته بدون مسیر و صفر به معنای بازبودن آن است. الگوریتمی بنویسید که مسیری از *MAZE*(۱،۱) :*MAZE*(*n*,*n*) پیدا کند. استفاده از بازگشت به عقت در اینجا لازم است. زمان الگوریتم را تجزیه و تحلیل کنید.

پاسخ :

```
int    r[ ] = {۰ , ۱ , ۰ , −۱} ;
int    c[ ] = {۱ , ۰ , −۱ , ۰} ;   راست ، پایین ، چپ ، بالا
Void   maze (int i , int j , int L)
  {
    int k,
    if ( promising (i,j) )
        if (i = = n && j = = n)
            for (int q = ۱ ; path [q] ! = ۰ : q ++)
                cout << path [q] ;
      else
            for (k = ۱ ; k < ۴ ; k ++)
            {  path [L + ۱] = k ;
               maze (i + r[k] , j + c[k] , L + ۱) ;
            }
  }
```

که در آن تابع *promising* چنین تعریف می شود.

```
Void   promising  (int i , int  j)
  {
     bool  switch = true ;
     if (S[i][j]==۱)
         Switch = false.
     return  switch
  }
```

راه دیگری هم برای این مسئله وجود دارد که به آن اشاره می کنیم.

ابتدا برای هر $1 \le i \le n$ تعریف می کنیم:

$$maze(0,i) = maze(i,0) = maze(i, n+1) = maze(n+1, i) = 0$$

هدف مسئله عبارت است از یافتن یک مسیر از یکها از $maze(1,1)$ تا $maze(n,n)$. پس از $maze(1,1)$ شروع می کنیم($i=j=1$) و برای هر i,jی که انتخاب شده، نقاط زیر را به ترتیب بررسی می کنیم:

$maze(i+1, j)$

$maze(i, j+1)$

$maze(i-1, j)$

$maze(i, j-1)$

و هر کدام که برابر ۱ بود و قبلاً هم انتخاب نشده است را i و j جدید فرض کرده دوباره مراحل را تکرار می کنیم. اگر هر چهار نقطه برابر صفر بودند و یا قبلاً انتخاب شده بودند، آخرین نقطه را به انتخاب بعدیش منتقل می کنیم.

۷- الگوریتم عقبگرد برای مسأله n وزیر را برای نمونه مسأله ای به کار ببرید که در آن $n=8$ است و عملیات را مرحله به مرحله نشان دهید. درخت فضای حالت هرس شده ای را نشان دهید که این الگوریتم تا نقطه رسیدن به حل نخست ایجاد می کند.

پاسخ :

باید درختی همانند درخت صفحهٔ ۲۵۵ کتاب رسم کرد، با این تفاوت که رأس ۲ دارای هفت فرزند باشد، و هر کدام از آن فرزندها دارای ۶ فرزند باشند، و الی آخر به علت زیاد بودن تعداد رأسها، رسم درخت آن در یک ورق کاغذ، خوانا نخواهد بود، ولی روش کار چنین است که بیان شد.

۸- نشان دهید در مسأله ۴ وزیر بدون عقبگرد پیش از یافتن نخستین حل باید ۱۰۰ گره چک شود.

پاسخ :

غلط چاپی دارد. به جای ۱۰۰ باید ۱۵۵ باشد.

درخت صفحهٔ ۲۵۵ را در نظر بگیرید که کامل باشد.
در درخت کامل آن قابل تحقیق است که رأس ۲ چهار فرزند دارد و هر کدام از آنها چهار فرزند و مجدداً هر کدام از اینها نیز چهار برگ. پس در سمت چپ رأس ۱ تعداد ۸۵=۱+۴+۱۶+۶۴ رأس داریم که با خود رأس۱ می‌شود ۸۶.
رأس ۱۱ با چهار فرزندش می شود ۵ رأس. سه فرزند سمت چپ۱۱ هر کدام چهار فرزند دارند و آنها نیز هر کدام ۴ برگ دارند.
پس تا اینجا می شود:

۸۶+۵+۱۲+۴۸=۱۵۱

رأس ۱۵ با دو فرزند چپ آن و همچنین رأس ۱۶ مجموعاً می شوند چهار رأس.
پس تا رسیدن به اولین جواب (که همان رأس ۱۶ است)، باید ۱۵۵=۴+۱۵۱ رأس چک شود.

۹- الگوریتم عقبگرد برای مسأله n وزیر را طوری اصلاح کنید که به جای تولید همه حل های ممکن فقط یک حل را بیاید.

پاسخ :

تابع

```
bool  queens (int i)
{
  bool S = true ;
  if (promising (i))
    if (i == n)
      { cout << col [1], col [2],..., col [r];
        return false ;
      }
    else
      for (int j = 1 , j <= n && S == true , j++1
          { col [i+1] = j
            S = quens (i+1) ;
          }
  return S ;
}
```

که در آن تابع *promising* همان تابعی است که در تمرین ۶ تعریف شده است.

۱۰- فرض کنید حل از مسأله n وزیر را که در آن $n=۴$ است، در اختیار داریم. آیا می توان این حل را برای یافتن حلی جهت نمونه $n=۵$ بسط داد و سپس از مایه های مربوط به $n=۵$ و $n=۴$ برای نمونه $n=۶$ استفاده کرده و سپس این روش برنامه سازی پویا را برای یافتن حل هر نمونه ای که در آن $n>۴$ باشد به کار برد؟ برای پاسخ خویش دلیل بیاورید.

پاسخ :

خیر. زیرا روش برنامه نویسی پویا برای مسائلی به کار می رود که اولاً از نوع بهینه سازی باشد، ثانیاً در اصل بهینه سازی صدق کند. واضح است که مسئلهٔ n وزیر فاقد این دو شرط اساسی می‌باشد.

۱۱- حداقل دو نمونه از مسأله n وزیر را پیدا کنید که حلی ندارند.

پاسخ :

برای $n=۲$ و $n=۳$ مسئلهٔ n– وزیر جواب ندارد.

۱۲- یک الگوریتم عقبگرد برای مسأله حاصل جمع زیر مجموعه ها بنویسید که اوزان را از قبل مرتب بکند. کارایی این الگوریتم را با الگوریتم بیان شده در فصل مقایسه کنید.

پاسخ :

فرض کنیم:

r=مجموع کلیه وزنهای باقیمانده

s=مجموع وزنهای انتخاب شده

k=شماره وزن

x=یک آرایه سراسری

w=یک آرایه سراسری که به صورت نزولی مرتب است

m=متغیر سراسری وزن مجموع مورد نظر

```
Void   SumSet (S , r , k)
  {
      x[k]=۱ ;
      if (S+W [k]= = m)
          cout << x[۱] , x[۲] ,..., x[n]
       else
          if (S + w[k]+ w[k+۱]<= m)
             SumSet  (s + w[k] , r – w[k] , k+۱);
```

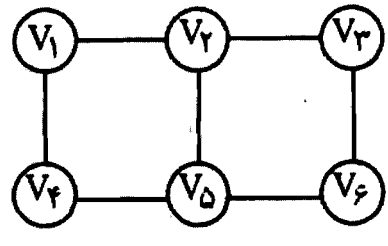
```
if (S + r − w[k] > = m) & &(S + w[k + ۱] <= m)
    {  x[k] = ۰
       SumSet(S , r − w[k] , k + ۱)
    }
}
```

باید دقت کرد که فرض می کنیم $\sum_{i=۱}^{n} w_i \geq m$, $w_۱ \leq m$ و در ابتدا الگوریتم بـه صـورت $SumSet\left(۰ , \sum_{i=۱}^{n} w_i , ۱\right)$ فراخوانی می شود.

۱۳- الگوریتم عقبگرد برای حل مسأله حاصل جمع زیر مجموعه ها را چنان اصلاح کنید کـه بـه جـای تولید همه حل های ممکن، فقط یک حل را پیدا کند. این الگوریتم را از لحاظ کارایی با الگوریتم مسأله ۱۳ و الگوریتم بیان شده در فصل مقایسه کنید.

پاسخ :

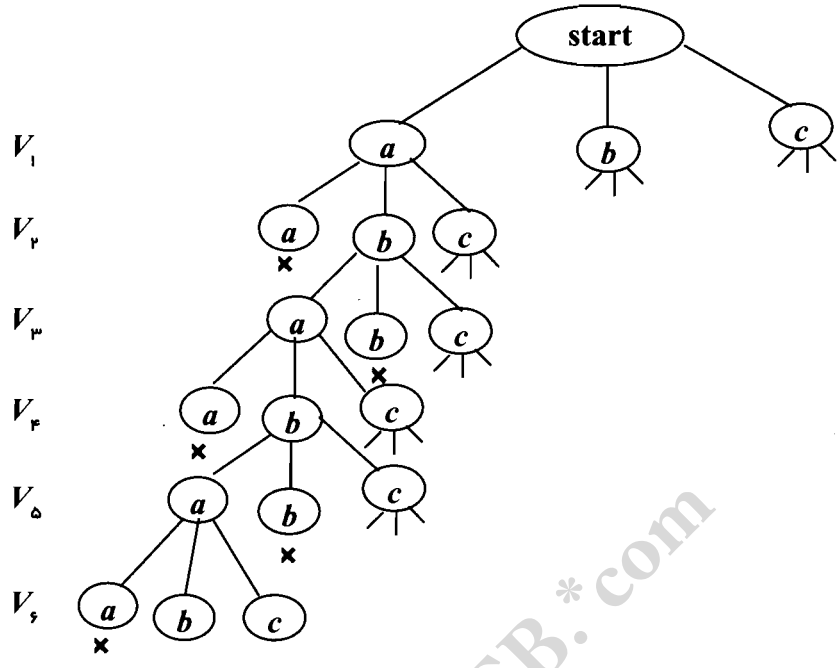
۱۴- از الگوریتم عقبگرد برای مسأله رنگ آمیزی m برای یافتن همه رنگ آمیزی های ممکن گراف زیر با استفاده از سه رنگ قرمز، سبز و آبی استفاده کنید. عملیات را مرحله به مرحله نشان دهید.



پاسخ :

رنگها را c,b,a می نامیم.

علامت ضربدر، گره های غیر امید بخش را مشخص می کنند و هر گرهی که به سه خط خـتم می شود یعنی مانند گره هم تراز خود ادامه می یابد.



یعنی شکل رسم شده، دو جواب زیر را نشان می دهد:

$V_1 \to a \quad V_2 \to b \quad V_3 \to a \quad V_4 \to b \quad V_5 \to a \quad V_6 \to b$

$V_1 \to a \quad V_2 \to b \quad V_3 \to a \quad V_4 \to b \quad V_5 \to a \quad V_6 \to c$

چون درخت کامل حجیم می شود، به رسم قمستی از آن اکتفا کرده ایم.

۱۵- فرض کنید برای رنگ آمیزی مناسب یک گراف یک رأس آغازی و یک رنگ انتخاب کرده هر تعداد رأس ممکن را رنگ آمیزی می کنیم. سپس رنگ جدیدی را انخاب کرده هر تعداد رأس ممکن از رئوس رنگ نشده را رنگ آمیزی می کنیم. این فرآیند را چندان ادامه می دهیم که همه رئوس گراف رنگ آمیزی شوند. الگوریتمی حریصانه جهت رنگ آمیزی یک گراف با n رأس بنویسید. این الگوریتم را تحلیل کنید و نتیجه را با نماد مرتبه نشان دهید.

پاسخ :

رأسهای گراف را $v_1, v_2, v_3, \ldots v_n$ فرض می کنیم. الگوریتم را به زبان ساده توضیح می دهیم. ابتدا رنگ اول را برداشته و v_1 را رنگ می کنیم. بعد به سراغ v_2 می رویم و همهٔ رأسهای مجاور آنرا چک می کنیم، اگر هیچکدام رنگ اول را نداشتند، v_2 را با رنگ اول رنگ آمیزی می کنیم. به سراغ v_3 می رویم و همهٔ رأسهای مجاور آنرا چک می کنیم و... تا رأس n ام هم به همین روال.

حال رنگ دوم را برداشته و سراغ اولین رأسی می رویم که بی رنگ است و آنرا رنگ می‌کنیم.

دوباره اولین اندیسی که بی رنگ را انتخاب کرده و تمام رأسهای مجاور آنرا چک می کنیم، اگر هیچگدام رنگ دوم را نداشتند، آنرا با رنگ دوم رنگ آمیزی می کنیم، الی آخر. روال را آنقدر ادامه می دهیم تا همهٔ رأسها رنگ شوند.

دقت شود که در هر مرحله، برای هر رأس به تعداد درجهٔ آن مقایسه انجام می شود. پس در کل به اندازهٔ مجموع درجه ها مقایسه انجام می شود. اگر m تعداد یالها و Δ بزرگترین درجه و k عدد رنگی گراف باشد، با توجه به فرمول:

$$\sum_{i=1}^{n}\deg(v_i)=2m\le n(n-1)$$

در هر مرحله از مرتبهٔ $O(n^2)$ مقایسه انجام می شود. چون به تعداد k مرتبه، مرحلهٔ فوق تکرار می شود و می دانیم $k\le\Delta+1\le n$ ، پس کل الگوریتم از مرتبهٔ $O(n^3)$ است.

۱۶- فرض کنید می خواهیم تعداد رنگ های به کار رفته در رنگ آمیزی یک گراف را کمینه سازی کنیم. آیا روش حریصانه یک حل بهینه را تضمین می کند؟ برای پاسخ خود دلیل بیاورید.

پاسخ :

بله- با برهان خلف می توان حکم را ثابت کرد. یعنی اگر تعداد رنگهای ما کمینه نباشد، حتماً می‌توان دو رأسی پیدا کرد مثل v_i وv_j که از دو رنگ مختلف باشند ولی قابلیت تبدیل به یک رنگ را داشته باشند. این موضوع با اینکه در هر مرحله ما بیشترین تعداد رأسی که قابلیت رنگ شدن در آن مرحله را دارند را رنگ می کنیم، تناقض دارد.

۱۷- کارایی الگوریتم عقبگرد برای مسأله رنگ آمیزی m و الگوریتم روش حریصانه مقایسه کنید.

پاسخ :

این دو الگوریتم دو کار مختلف انجام می دهند.

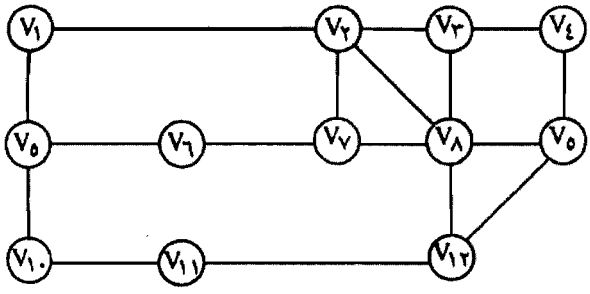
الگوریتم عقبگرد، برای هر m ی همهٔ گونه های رنگ آمیزی با m رنگ را نشان میدهد ولی الگوریتم حریصانه، یک نوع رنگ آمیزی با حداقل تعداد رنگ را پیدا می کند. برای همین هم الگوریتم حریصانه خیلی سریعتر از الگوریتم عقبگرد عمل می کند.

۱۸- چند کاربرد عملی بنویسید که برحسب مسأله رنگ آمیزی m قابل ارائه باشد.

پاسخ :

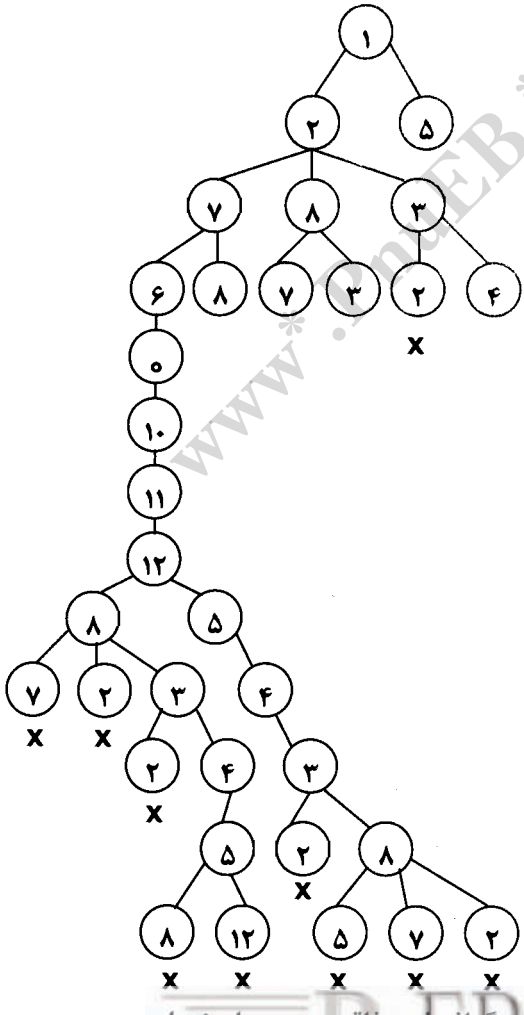
تعیین عدد رمزی یک گراف – قفسه بندی مواد شیمیایی با تشعشعات رادیو اکتیوی و غیره.

۱۹- از الگوریتم عقبگرد برای مسأله مدارهای هامیلتونی جهت یافتن همه مدارهای هامیلتونی ممکن برای گراف شکل زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید. الگوریتم فوق روی کامپیوتر پیاده سازی کنید.



پاسخ :

دو رأس v_5 وجود دارد. سمت چپی را v_0 می نامیم. قسمتی از درخت آنرا میکشیم (منظـور از هر عددی، اندیسی آن است، یعنی منظور از ۳، V_3 است).



با تکمیل درخت فوق دیده می شود گراف فوق فاقد دور همیلتونی است.

۲۰- الگوریتم عقبگرد برای مساله مدارهای هامیلتونی را چنان اصلاح کنید که به جای تولید همه حل‌های ممکن ، فقط یک حل را پیدا کند . کارایی این الگوریتم را با الگوریتم بیان شده حرکت ب مقایسه کنید .

پاسخ :

۲۱- سه کاربرد دیگر برای راهبرد عقبگرد بیان کنید.

پاسخ :

۱) در یک گراف مفروض G، همهٔ مسیرهای موجود بین دو رأس داده شده را بیابد.

۲) یک رمز n رقمی که هر یک از ارقام آن می تواند صفر یا یک باشند را بیابد، با این شرط که می دانیم تعداد ارقامی از رمز صحیح که برابر یک می باشند، حداقل $\frac{n}{۲}$ تا است.

۳) در حالت کلی، اکثر الگوریتمهای هوشمند (مثل پیدا کردن مسیر توسط رباتها و...) از تکنیک عقبگرد استفاده می کنند.

۲۲- الگوریتم عقبگرد برای مسأله وزیر را طوری اصلاح کنید که فقط راه حل هایی را تولید نماید که در اثر انعکاس و چرخش ثابت می مانند.

پاسخ :

```
Void   Queen (int i)
  {
    int  j ;
      if  (prom(i) )
        if (i = = n)
          cout << col [۱] , col [۲] , ... , col[n] ;
        else
        { if  (i = =۱)
                k = n/۲ ,
            else
                k = n ;
            for (j =۱ ; j < = k ; j + +)
                { col [i +۱] = j
                  Queen (i +۱) ;
                }
        }
  }
```

که در آن تابع *prom* همان تابع *promising* کتاب است با این تغییر که شرط زیر نیز به آن اضافه می شود.

$$if\ (col[i]!=(c-col[i]+1);\ then\ Switch\ =\ false.)$$

۲۳- با یک مکعب $n\times n\times n$ که شامل n^3 خانه است. می خواهیم n وزیر را در یک مکعب قرار دهیم، به طوری که هیچ دو وزیری به یکدیگر گارد ندهند (به طوری که هیچ دو وزیری در یک سطر، یک ستون یا یک قطر نباشند). آیا الگوریتم n وزیر می تواند برای حل این مسأله بسط داده شود؟ اگر ممکن است آن الگوریتم را بنویسید و آن را برای مسأله های n=۴ و n=۸ اجرا کنید.

پاسخ :

باید n آرایهٔ $n\times n$ در نظر گرفت و براساس معیارهای مسئلهٔ n وزیر، برای هر صفحه، و یک سری معیارهای جدید برای بُعد سوم (ارتفاع) و قطرهای مکعب، روش چیدمان وزیرها را مشخص کرد.

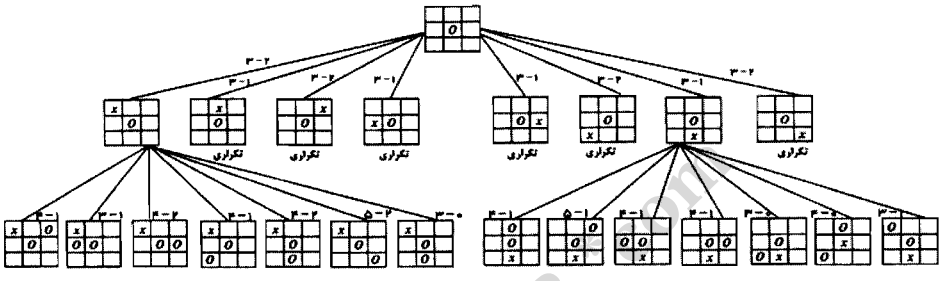
۲۴- الگوریتم عقبگرد مربوط به مساله هامیلتون را طوری اصلاح کنید که مدارهای هامیلتونی با کمترین هزینه را برای گراف موزون پیدا نماید . این الگوریتم چگونه عمل خواهد کرد ؟

پاسخ :

تمرینات فصل هشتم

۱- در بازی سه به سه قطار یک مربع به ۹ مربع کوچکتر تقسیم می شود. دو بازیکن به نوبت بازی می‌کنند و در هر نوبت یک بازیکن یک مربع کوچک را با علامت ویژه خود علامتگذاری می کند. بازیکنی که ابتدا سه مربع یک ردیف یا یک ستون یا یک قطر را با علامت ویژه خود علامتگذاری کند برنده خواهد بود. این بازی را تحلیل کنید و روش پیشنهادی خود را برای برنده شدن یکی از بازیکن ها بر اساس اصول انشعاب و تحدید بنویسید.

پاسخ :



و به همین ترتیب برخی از مراحل طبق روش انشعاب و تحدید حذف میگردند. توجه شود که در مواقعی که X بازی می کند بهتر است که کمترین نرخ را انتخاب کند و در مواقعی که0 بازی می‌کند بهتر است که بیشترین نرخ را انتخاب کند، و این امر با توجه به فرمول نرخ که به صورت زیر تعریف می شود واضح است.

نرخ هر مرحله = (حالات برد 0) - (حالات برد X)

۲- با استفاده از روش انشعاب و تحدید برای مسأله کوله پشتی صفر و یک سود ماکزیمم قابل حصول از نمونه زیر را پیدا کنید. عملیات را مرحله به مرحله نشان دهید.

n=۵ W=۲۵

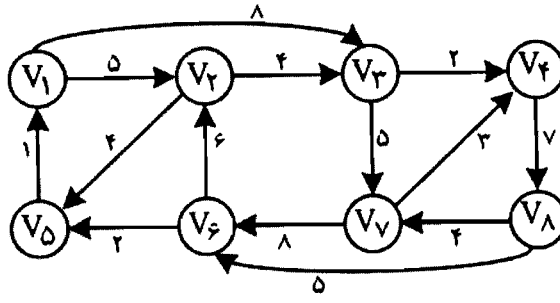
i	P_i	W_i	P_i/W_i
۱	$ ۲۷	۳	۹
۲	$ ۳۰	۶	۵
۳	$ ۳۵	۷	۵
۴	$ ۱۸	۹	۲
۵	$ ۳	۳	۱

پاسخ :

با دنبال کردن مراحل مثال ارائه شده در صفحهٔ ۲۹۶ کتاب، به جواب زیر می‌رسیم.

$x_1 = x_2 = x_3 = x_4 = 1$, $x_5 = 0$

۳- با استفاده از روش انشعاب و تحدید برای مسأله فروشنده دوره گرد به یک تور بهینـه و طـول آن را برای گراف زیر پیدا کنید. مرحله به مرحله حل مسئله را نمایش دهید.



پاسخ :

پس از دنبال کردن مراحل مثال صفحهٔ ۲۹۰ کتاب، به جواب زیر می رسیم.

$$V_1 \to V_2 \to V_3 \to V_7 \to V_4 \to V_8 \to V_6 \to V_5 \to V_1$$

۴- یک الگوریتم انشعاب و تحدید برای مسأله زمان بندی با مهلت معین بنویسید.

پاسخ :

```
Struct   node
{
   int level ;
   Sequence – of – integer k  ;
    int  profit ;
    int  bound ;
    }
//
Void schedule (int n , Sequence – of –integer  j , number and  max profit)
{
   priority – queue –of – node  PQ ;
                          node  U , V ;
   initialize  (PQ) ;
   V . Level = ۰ ;
   V .  k = [۱] ;
   V .  profit =  profit  [V .level + ۱] ;
   V . bound = bound (V) ;
   max profit = V .profit ;
   insert (PQ , V) ;
   while  (!empty (PQ))
   {  remove  (PQ , V) ;
```

```
if (V.bound > max profit)
   {u.level = V. level + ۱;
     for (all i such that ۲ ≤ i ≤ n and i is not inV.k)
          {u.profit = V. profit + profit[i] ;
             u.k = V.K with i added according to
                          nondecreasing value of
                          dead line[i]
           if (u.k is feasible and U.level < n and U.profit > max profit)
              {max profit = u.profit ;
                  J = u.k ;
               }
           if (u.k is feasible)
              {u.bound = bound (u) ;
               if (u.bound > max profit)
                   insert (PQ , U);
               }
             }
             }
          }
       }
//
int bound (node u)
 {
      int i ;
Sequence − of − integer q,k ;
q = u.k ;
result = u.profit ;
for (int i = ۲ , j ≤ n and i is not in q ; i++)
      {k = q with i added according to nondecreasing
                                value of deadline [i];
         if (k is feasible)
            {q = k
               result = result + profit [i] ;
               }
            return result ;
           }
```

۵- کارایی برنامه نویسی پویا و بازگشت به عقب را برای مسأله کوله پشتی مقایسه کنید.

پاسخ :

همانطوری که قبلاً دیده شد روش برنامه نویسی پویا برای مسئلهٔ کوله پشتی از مرتبهٔ $O(min(۲^x, nw))$ است و روش عقبگرد از مرتبهٔ $O(۲^n)$. پس به نظر میرسد که روش برنامه نویسی پویا بهتر است. البته باید توجه داشت که معمولاً روش عقبگرد، به بدترین حالت

نمی‌رسد. به همین دلیل دیده می شود که پس از اجرای این دو الگوریتم روی چندین نمونهٔ عددی، معمولاً روش عقبگرد زمان واقعی کمتری لازم دارد.

۶- الگوریتمی به روش انشعاب و تحدید برای حل مسئله چهار وزیر ارائه دهید، سپس آنرا برای حالت n توسعه دهید.

پاسخ :

برای محاسبهٔ *bound* هر گره، باید تعداد شاخه هایی که به گره ختم می شوند را شمارش کرد.

۷- الگوریتم ارائه داده شده در مسئله ۶ را تحلیل زمانی کنید.

پاسخ :

دیده می شود که الگوریتم فوق از مرتبهٔ نمایی است.

۸- درخت فضای حالات را برای مسئله ۶ ترسیم نموده سپس مراحل آن را توصیف کنید.

پاسخ :

درخت صفحهٔ ۲۵۵ کتاب را به صورت سطری پیمایش کنید.

۹- جستجو با هرس کردن انشعاب و تحدید را برای مسئله n وزیر به کار گرفته سپس آن را با الگوریتم عقبگرد مقایسه نمایید.

پاسخ :

پس از اجرای هر دو الگوریتم دیده می شود که روش انشعاب و تحدید زودتر به جواب می‌رسد.

۱۰- الگوریتم انشعاب و تحدید برای مساله مدارهای هامیلتونی ارائه دهید ، سپس آن را با روش عقبگرد مقایسه نمایید .

پاسخ :

۱۱- الگوریتم انشعاب و تحدید برای مساله مدارهای هامیلتونی را چنان اصلاح کنید که به جای تولید همه حل‌های ممکن ، فقط یک حل را پیدا کند .

پاسخ :

۱۲- الگوریتمی به روش انشعاب و تحدید برای حل مسئله رنگ آمیزی با m رنگ ارائه دهید، سپس کارایی آن را نسبت به الگوریتم ارائه شده به روش عقبگرد مقایسه نمایید.

پاسخ :

```
bool   bound    (nod   V)
{
   int    i, j ;
    bool   switch = true ;
    i = u.level ;
    j = ۱
          while  (j < i   and   switch)
                  {if    w[i][j] and u.color [i] == u.color  [j]
                          switch = false ;
                      j++;
                  }
                return   switch
              }
```

دقت شود که ساختار *u.level* همانی است که در مسئلهٔ ۴ ارائه شد.

البته راه دیگری هم موجود است. می توان تابع *promising* را به عنوان تابع *bound* تعریف کرد و الگوریتم را چنین نوشت.

```
struct
{
    int  level ;
    int    color [n] ;
    bool  bound  ;
  };
Void    m – coloring  (int  n ,  int  m)
 {
    queue – of – nodes  Q  ;
     node    U,V  ;
     intialize  (Q) ;
     v.level = ۰
     v .bound = true
     v. color [ ] = {۰,۰ ,...., ۰}
     insert  (Q,V)
      while (! empty (Q))
```

```
{ remove    (Q,V)
   if    (V.level = = n)
        cout << v. color[۱] , V.color[۲] ,..., V.color [n] ;
    else {u.level = V. level +۱ ;
            for (color =۱; color < = m , color++)
            {u.color [V.level +۱] = color ;
              u.bound = bound (u) ;
              if (u.bound = = true)
                  insert (Q,U) ;
            }
         }
    }
}
```

۱۳- فرض کنید برای رنگ آمیزی مناسب یک گراف یک رأس آغازی و یک رنگ انتخاب کرده هر تعداد رأس ممکن را رنگ آمیزی می کنیم. سپس رنگ جدیدی را انتخاب کرده هر تعداد رأس ممکن از رئوس رنگ نشده را رنگ آمیزی می کنیم. این فرآیند را چندان ادامه می دهیم که همه رئوس گراف رنگ آمیزی شوند. الگوریتمی به روش انشعاب و تحدید جهت رنگ آمیزی یک گراف با n رأس بنویسید. این الگوریتم را تحلیل کنید و نتیجه را با نماد مرتبه نشان دهید.

پاسخ :

۱۴- الگوریتمی به روش انشعاب و تحدید برای مساله مجموع زیر مجموعه‌ها ارائه دهید. حل حاصل را با حل بدست آمده برای مسئله به روش عقبگرد مقایسه نمایید . آیا زمان الگویتم جدید بهتر شده است؟

پاسخ :

```
struck  node
{
   int  level ;    ← سطح درخت
   int  weight ;  ← مجموع وزن گره
   int  bound ;   ← حد
   bool  include [n] ; ← وزنهای انتخاب شده
   } ;
```

در ابتدا $total = \sum_{i=۱}^{n} w_i$

```
Void    sum -of- subsets (int n)
{
    queue -of- Queue  Q ;
    node   u , V;
     int ialize (Q);
     V. level = ۰
     V. weight = ۰
     V. include [ ]={false , false ,..., false},
     V. bound = total ;
     insert (Q,V)
     while (! empty (Q))
      { remove  (Q,V)
         U. level = V.level + ۱
         U. weight = V.weight + w [u. level];
         U .include [V.level + ۱]= true
          total = total - w[u.level]
         U.bound = bound (u)
          if(u.weght = = w)
              cout << u.include [ ۱ ], u.include [ ۲ ],..., u.include [n]
else
    if( (u.bound > w[u.level] || u.bound = = ۰) and
        W [u.level]>= w )
         insert (Q,U) ;
    U.weight = V.weight
    U. include [V.level + ۱]= false
    U.bound = bound (u)
    if (u.bound > w [u.level] || u.bound == ۰)
          insert  (Q,U)
     }
 }
```

```
int   bound  (node u)
{
    int  S  ;
    S = total + U.weight ;
      return (S – w);
     }
```

۱۵- از الگوریتم انشعاب و تحدید برای مسأله مدارهای هامیلتونی جهت یافتن همه مدارهای هامیلتونی ممکن برای گراف شکل زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید. الگوریتم فوق روی کامپیوتر پیاده سازی کنید.

پاسخ :

با اجرای الگوریتم فوق روی این گراف، ملاحظه می شود که گراف داده شده فاقد دور همیلتونی می باشد.

تمرینات فصل نهم

۱- سه مسأله NP پیدا کرده و به دقت تشریح کنید.

پاسخ :

اولین مسئله: مسئلهٔ پیدا کردن کوتاهترین دور همیلتونی (یا همان مسئلهٔ فروشندهٔ دوره‌گرد).

دومین مسئله: مسئلهٔ SAT که چنین تعریف می شود: یک عبارت منطقی داریم. ممکن است با یک ارزشگذاری، ارزش کل عبارت ۱ شود و ممکن است هیچ ارزشگذاری ارزش آنرا یک نکند. مثلاً

$$(x \wedge y) \wedge (x \vee \neg y)$$

با ارزشگذاری x=۱ و y=۱ به ارزش ۱ می رسد، ولی عبارت $x \wedge \neg x$ با هیچ ارزشگذاری یک نمی‌شود.

مسئلهٔ SAT این است که آیا می توان یک ارزشگذاری پیدا کرد که عبارت داده شده یک شود یا خیر.

سومین مسئله: یک گراف را بگیرد و بزرگترین زیر گراف کامل آنرا مشخص کند. این مسئله را *Clique* می گویند.

۲- برای مرتب سازی داده های آرایه ای از اعداد یک الگوریتم نامعین طراحی کنید.

پاسخ :

یک ترتیب از اعداد را حدس بزن و امتحان کن که آیا مرتب شده هست یا نه؟

اگر بود، چاپش کن و اگر نبود یک حدس معقول دیگر بزن.

```
Void  nd – sort (A[ ], n)
{
   for (i = ۰ , i < n , i++)
         B[i] = ۰
   for (i = ۰ , i < n , i++)
```

```
{ j = choice (۰, n−۱) ;   انتخاب تصادفی اندیس آرایه
   if (B [j] ! = ۰)        failure ( ) ;
   B [j] = A[j]
   }
for (i = ۰ , i < n−۱ , i++)
     if (B[j] > B[j+۱])      failure ( );
cout << B[۱] , B[۲ ],...., B[n]
success( ) ;
}
```

۳- یک الگوریتم مرتبه زمانی چند جمله ای بنویسید که چک کند آیا یک گراف بدون جهت دارای مدارهای هامیلتونی هست یا خیر. فرض کنید این گراف هیچ رأسی با درجه بیشتر از ۲ ندارد.

پاسخ :

اگر رأسی با درجهٔ صفر یا یک وجود داشته باشد، گراف دور همیلتونی ندارد. اگر همهٔ رأسها درجهٔ دو داشته باشند و گراف همبند باشد، آنگاه دور همیلتونی دارد و اگر ناهمبند بود، دور همیلتونی ندارد.

۴- تابع حدس اولیه برای مسائل NP بنویسید و آنرا تحلیل نمایید.

پاسخ :

سؤال ناقص است، چون هر مسئله ای، تابع حدس اولیهٔ مربوط به خودش را دارد و نمی توان به شکل کلی یک تابع حدس نوشت، ولی همگی بر حسب اعداد تصادفی کار می کنند.

۵- مسائل فروشنده دوره گرد، رنگ آمیزی گراف و مجموع زیر مجموعه ها را به شکل مسائل تصمیم‌گیری ارائه دهید.

پاسخ :

فروشندهٔ دورگرد: آیا گراف دارای دور همیلنی با طول کمتر از d هست یا نه؟

رنگ آمیزی گراف: آیا گراف را می توان با d رنگ، رنگ آمیزی کرد یا نه؟

مجموع زیر مجموعه ها: آیا مجموع زیر مجموعهٔ $\{x_1,..x_d\}$ برابر w هست یا نه؟

۷- مسائل *NP-hard* را تعریف نموده، سپس رابطه آن را با مسائل *NP* مشخص نمایید.

پاسخ :

مسئله ای را *NP-hard* می گوییم که اگر بتوان برای آن یک الگوریتم قطعی چند جمله ای یافت، آنگاه می توان برای همهٔ مسائل *NP* هم یک الگوریتم قطعی چند جمله ای پیدا کرد. مسائل *NP-hard* با *NP* اشتراک دارد. در واقع

$$(NP - hard) \cap (NP) = (NP - complet)$$

۸- مسائل *NP-complete* را تعریف نموده، سپس رابطه آن را با مسائل *NP* مشخص نمایید.

پاسخ :

مسئله ای را *NP-complet* می گوئیم که اولاً *NP* باشند، ثانیاً *NP-hard* باشد.

$$(NP - complet) \subset NP$$

۹- رابطه بین مسائل از نوع *NP-complete, NP-hard, P* را مشخص نمایید.

پاسخ :

می دانیم *(NP-hard)* ⊂ *(NP-complet)*. همچنین می دانیم

$$P \subseteq NP$$

بین *P* و *(NP-complet)* هنوز رابطهٔ مستقلی پیدا نشده ولی گزارهٔ شرطی برقرار است.

$$if \quad P \cap (NP - complet) \neq \varnothing \Rightarrow P = NP$$

سوالات چهارگزینه ای

۱- مرتبه اجرای الگوریتم زیر چیست؟

```
For j:=۱  to  m  do
  For  k:=۱  to j  do
      x:= x+۱ ;
```

۱) $O\left(\frac{m+۱}{۲}\right)$ ۲) $O(m^۲)$ ۳) $O(\log_۳ m)$ ۴) $O(m^۳)$

پاسخ :

گزینهٔ ۲. چون دو حلقهٔ تودرتو دارد که هر کدام حداکثر n می شوند.

۲- در الگوریتم زیر در صورتی که $n=m$ باشد مرتبه اجرایی کدام است؟

```
For  i:=۱  to  n  do
  For  j:=۱  to  m  do
    For  k:=۱  to  j  do
        X:=x+۱ ;
```

۱) $O\left(\frac{m+۱}{۲}\right)$ ۲) $O\left(\frac{m(m+۱)}{۲}\right)$ ۳) $O(n^۲)$ ۴) $O(n^۳)$

پاسخ :

گزینهٔ ۴. چون سه حلقهٔ تودرتو دارد که هر کدام حداکثر n می شوند.

۳- زمان اجرای الگوریتم $T(n)$ به صورت زیر برابر کدام گزینه است؟

$$T(n)=\begin{cases}۱ & n=۱\\ n+T(n-۱) & n\geq ۲\end{cases}$$

۱) $O(n)$ ۲) $O(n^{۳/۲})$ ۳) $O(n\log n)$ ۴) $O(n^۲)$

پاسخ :

گزینهٔ ۴. ابتدا n ها را به $n+۱$ تبدیل می کنیم و از رابطهٔ اصلی کم می کنیم.

$$\left.\begin{aligned}T(n)-T(n-۱)&=n\\ T(n+۱)-T(n)&=n+۱\end{aligned}\right\}\Rightarrow T(n+۱)-۲T(n)+T(n-۱)=۱$$

یکبار دیگر n را به $n+۱$ تبدیل کنید و از رابطهٔ فوق کم کنید.

$$T(n+۲)-۳T(n+۱)+۳T(n)-T(n-۱)=۰\Rightarrow (x-۱)^۳=۰\Rightarrow$$

$$T(n)=c_۱+c_۲n+c_۳n^۲\in O(n^۲)$$

۴- مرتبه اجرای برنامه زیر کدام است؟

```
i = n;
while (i > ۱) {
    i = i / ۲ ; j = n;
    while (j > ۱)
    j = j / ۳
}
```

۱) $O(\log_{۲} n)$ ۲) $O(\log_{۳} n)$ ۳) $O(\log_{۶} n)$ ۴) $O(\log_{۲} n \times \log_{۳} n)$

پاسخ :

گزینهٔ ۴. چون دو حلقهٔ تودوتو دارد که یکی به تعداد $\log_{۲}^{n}$ و دیگری $\log_{۳}^{n}$ تکرار می شوند.

۵- تابع بازگشت زیر را در نظر بگیرید:

```
int   recursive (int n)
{
    if ( n == ۱)
        return ۱;
    else
        return(recursive(n-۱) + recursive(n-۱));
}
```

۱) ۱۶ ۲) ۸ ۳) ۲۳ ۴) ۱۴

پاسخ :

سؤال ناقص است.

۶- در برنامه زیر مقدار $F(۳و۶)$ برابر است با:

```
int   F(int m , int n)
{
    if (m == ۱ || n == ۰ || m == n)
        return ۱;
    else
        return (F(m-۱, n) + F (m-۱, n-۱) );
}
```

۱) ۲۰ ۲) ۱۰ ۳) ۱۸ ۴) ۴

پاسخ :

گزینهٔ ۴.

$$F(۳,۶)=F(۲,۶)+F(۲,۵)=F(۱,۶)+F(۱,۵)+F(۱,۵)+F(۱,۴)=۴$$

۷- تابع بازگشتی زیر را در نظر بگیرید:

```
int test (int n)
{
  if (n <= ۲)
      return ۱ ;
  else
      return test (n − ۲) * test (n − ۲) ;
}
```

زمان اجرای تابع فوق برابر است با:

۱) $O(n^۲)$ ۲) $O(n\log n)$ ۳) $O(۲^{n/۲})$ ۴) $O(۲^n)$

پاسخ :

گزینهٔ ۳ . چون در هر مرحله ۲ واحد از n کم می شود، پس بعد از $\frac{n}{۲}-۱$ مرحله اجرا به $test(۱)$ یا $test(۲)$ می رسیم، و چون در هر مرحله، تعداد ضربها ۲ برابر می شود، پس زمان کل می‌شود $\theta(۲^{\frac{n}{۲}-۱})$ که از مرتبهٔ $\theta(۲^{n/۲})$ است.

۸- تابع ACK به صورت زیر تعریف می شود. مقدار ACK (۱،۱) برابر است با:

```
int ACK (int m , int n)
{
      if (m < ۰ || n < ۰)
          return ۰ ;
      else if (m = = ۰)
          return n + ۱ ;
      else if (n = = ۰)
          return ACK (m − ۱ , ۱) ;
      else
          return ACK (m − ۱ , ACK (m , n − ۱)) ;
}
```

۱) ۵ ۲) ۴ ۳) ۳ ۴) ۶

پاسخ :

گزینهٔ ۳.

$$A(۱,۱)=A(۰,A\ (۱,۰)\ )=A(۰,A(۰,۱)\ )=A(۰,۲)=۳$$

۹- مجموع مراحل خطوط در برنامه زیر چند است؟

```
float sum (int num [ ], int n)
{
    int i , temp= ۰ ;
    for (i = ۰ ; i < n ; i ++)
        temp + = num [i];
    return temp ;
}
```

۱) $۲n+۳$ ۲) $۲n+۱$

۳) تعداد مراحل بستگی به n دارد و نا مشخص است ۴) $n+۱$

پاسخ :

گزینهٔ ۲.

$$۱+n+(n-۱)+۱=۲n+۱$$

۱۰- تعداد مراحل کل خطوط در برنامه زیر چقدر است؟

```
float rsum( float list[ ],int n)
    if (n)
        return( rsum(list ,n-۱) + list[n-۱]) ;
    return list[ ۰ ];
}
```

۱) $۲n+۴$ ۲) $۲n+۲$ ۳) $۲n^۲$ ۴) $۲(n-۱)^۲$

پاسخ :

سؤال ناقص است.

۱۱- تعداد مجموع مراحل خطوط در برنامه زیر چند است؟ (MS ثابتی است که حداکثر اندازه ماتریسها را مشخص می سازد).

```
void add(int a[ ][ MS ], int b[ ][MS ], int c[ ][MS ], int r, int c)
{
    int i, j ;
    for (i = ۰ ; i < r ; i++)
        for (j = ۰ ; j < c ; j++)
        c [i ][ j ] = a[i ][j ]+ b[i ][j ] ;
```

۱) $r*c$ ۲) $۲r*c+r+c$ ۳) $۲r*c+۲r+۱$ ۴) $MS*r*c$

پاسخ :

گزینهٔ ۳.

$$(r+۱)+r(c+۱)+rc=۲rc+۲r+۱$$

۱۲- کدامیک از روابط زیر نشان دهنده رابطه صحیح زمان محاسبه الگوریتم های مختلف است؟

۱) $O(\log n)<O(n)<O(n\log n)<O(۲^n)<O(n^۲)$

۲) $O(n)<O(\log n)<O(n\log n)<O(۲^n)<O(n^۲)$

۳) $O(n)<O(\log n)<O(n\log n)<O(n^۲)<O(۲^n)$

۴) $O(\log n)<O(n)<O(n\log n)<O(n^۲)<O(۲^n)$

پاسخ :

گزینهٔ ۴.

۱۳- تابع زیر روی عدد طبیعی X چه عملی انجام می دهد؟

```
int g(int X)
{
    if (X >۱)
        return X * g(X -۱);
    else
        return ۱;
}
```

۱) ۱ ۲) X^X ۳) $X!$ ۴) $\sum_{i=۱}^{X} i$

پاسخ :

گزینهٔ ۳. چون $0!=1!=1$ ، $n!=n(n-1)!$

۱۴- با توجه به تابع روبرو $Func(100)$ چه خواهد بود؟

```
int Func (int n)
{
    if (n==0)
        return 0 ;
    return (n + Func(n-1) );
}
```

۱) ۱۹۹ ۲) ۲۰۰ ۳) ۵۰۵۰ ۴) ۱۰۰۰۰

پاسخ :

گزینهٔ ۳.

$$F(100)=100+F(99)=....=100+99+...+1=\frac{100\times101}{2}=5050$$

۱۵- می‌خواهیم تابعی بازگشتی برای بیان تابع زیر که روی اعداد صحیح تعریف می گردد بنویسیم گزینه صحیح کدام است؟

$$F(n)=2n-1+2^n \qquad n>=0$$

۱)
$$F(n)=4F(n-1)-3F(n-2)-5$$
$$F(0)=0$$
$$F(1)=3$$

۲)
$$F(n)=3F(n-1)-2F(n-2)-5$$
$$F(1)=3$$
$$F(2)=7$$

۳)
$$F(n)=4F(n-1)-5F(n-2)-2F(n-3)$$
$$F(0)=0$$
$$F(1)=3$$
$$F(2)=7$$

۴)
$$F(n)=4F(n-1)-5F(n-2)-5$$
$$F(0)=0$$
$$F(1)=3$$

پاسخ :

گزینهٔ ۳. با توجه به اینکه سه ضریب داریم، پس احتیاج به سه مقدار اولیه می‌باشد.

۱۶- کدامیک از مجموع توابع زیر بر حسب افزایش مرتبه (*order*) از چپ به راست مرتب هستند؟

۱) $\left(۱.۰۰۵^{n}, n^{۱۰۰۰}, n!\right)$ ۲) $(۱.۰۰۵)^{n}, n!, n^{۱۰۰۰}$

۳) $n^{۱۰۰۰}, (۱.۰۰۵)^{n}, n!$ ۴) $n^{۱۰۰۰}, n!, (۱.۰۰۵)^{n}$

پاسخ :

گزینهٔ ۲.

۱۷- در برنامه زیر تعداد دفعات تکرار دستورالعمل شماره ۳ برابر است با....

۱) $for\left(k=۰\,;\, k<=n-۱\,;k++\right)$

۲) $for\left(i=۱\,;\, i<=n-k\,;i++\right)$

۳) $a[i][i+k]=k\,;$

۱) $n^{۲}$ ۲) $\frac{n(n+۱)}{۲}$ ۳) $\frac{n^{۲}}{۲}$ ۴) $\frac{n(n-۱)}{۲}$

پاسخ :

گزینهٔ ۴. چون برای k=۰ خط۲، مرتبه و خط ۳، n-۱ مرتبه تکرار می شود. با ادامهٔ روش بالا دیده می شود جواب برابر است با:

$$(n-۱)+(n-۲)+...+۱=\frac{n(n-۱)}{۲}$$

۱۸- می خواهیم n خط را در یک صفحه رسم کنیم. با فرض اینکه هیچ دو خطی، موازی همدیگر نیستند و همچنین بیشتر از دو خط همدیگر را در یک نقطه قطع نمی کنند، تعداد نواحی تولید شده توسط این خطوط چیست؟

۱) $۲n+۱$ ۲) $\frac{n(n+۱)}{۲}+۱$ ۳) $۳n-۱$ ۴) $n^{۲}$

پاسخ :

گزینهٔ ۲. با روش امتحان چند عدد می شود جواب را پیدا کرد. ولی راه کلی آن به صورت زیر است. خط اول دو ناحیه ایجاد می کند. خط دوم، دو ناحیه اضافه میکند. خط سوم سه ناحیه اضافه می کند و... و خط n ام، n ناحیه اضافه می کند. پس:

$$۲+۲+۳+۴+...+n=۱+۱+۲+۳+...+n=۱+\frac{n(n+۱)}{۲}$$

۱۹- تابع زیر چه کاری را انجام می دهد و $L(۲۵)$ را محاسبه نمایید.

$$L(n)=\begin{cases}۰ & if \quad n=۱\\ (\lfloor n/۲ \rfloor)+۱ & if \quad n>۱\end{cases}$$

۱) نصف عدد داده شده + ۱و ۱۳

۲) بزرگترین عدد صحیح به طوری که $n \le ۴,۲^{L}$

۳) $۴, L=\lceil \log_{۲} n \rceil$

۴) ۲و۳ صحیح است.

پاسخ :

گزینهٔ ۱ درست‌تر از بقیه به نظر می رسد(به شرط آنکه منظورمان از نصف، معرفی شود).

۲۰- اگر $F(n)=۵n+۱۰۰$ و $C=۶$ و $g(n)=n$ باشد به ازای کدام مقدار n. رابطه $F(n)=O(g(n))$ برقرار است؟

۱) ۱۲۰ ۲) ۶۳ ۳) ۵۸ ۴) ۱۰۰

پاسخ :

گزینه های ۱و۴ می توانند درست باشند زیرا می خواهیم رابطهٔ زیر برقرار باشد.

$$n \ge n_{۰} \Rightarrow ۵n+۱۰۰ \le ۶n \Rightarrow ۱۰۰ \le n$$

پس $n_{۰}$ می تواند هر عددی بزرگتر یا مساوی ۱۰۰ باشد ولی جواب درستتر، گزینه ۴ است.

۲۱- زمان جمع دو چند جمله ای یک متغیر که بر اساس توان X به صورت نزولی مرتب شده اند و یکی M جمله و دیگری n جمله دارد از چه مرتبه ای است؟

۱) $O(m \times n)$ ۲) $O(m+n)$ ۳) $O(۲^{m \times n})$ ۴) $O(۲^{m+n})$

پاسخ :

گزینهٔ ۲. البته می توان الگوریتمی ارائه داد که از زمان $O(max\{m,n\})$ باشد.

۲۲- خروجی تابع زیر که به صورت $F(۱و۲)$ صدا زده شده است کدام است؟

```
int  F(int m , int n)
{
    if (m==۰)
        return ++n ;
    if (n==۰)
        return F(m-۱ , ۱) ;
    return F(m-۱ , F (m ,n-۱)) ;
}
```

۱) ۷ ۲) ۵ ۳) ۸ ۴) ۳

پاسخ :

گزینهٔ ۲.

$$F(۲,۱)=F(۱,F(۲,۰))=F(۱,F(۱,۱))$$
$$F(۱,۱)=F(۰,F(۱,۰))=F(۰,F(۰,۱))=F(۰,۲)=۳$$
$$\Rightarrow F(۲,۱)=F(۱,۳)=F(۰,F(۱,۲))$$
$$F(۱,۲)=F(۰,F(۱,۱))=F(۰,۳)=۴$$
$$\Rightarrow F(۲,۱)=F(۰,۴)=۵$$

۲۳- اگر تابع A به صورت بازگشتی زیر تعریف شده باشد:

$$A(m,n)\begin{cases} n+۱ & \text{اگر } m=۰ \\ A(m-۱,۱) & \text{اگر } m\neq ۰ \ , \ n=۰ \\ A(m-۱,\ A(m,n-۱)) & \text{اگر } m\neq ۰ \ , \ n\neq ۰ \end{cases}$$

آنگاه مقدار خروجی تابع $A(۲و۲)$ کدام است؟

۱) ۹ ۲) ۱۵ ۳) ۴ ۴) ۷

پاسخ :

گزینهٔ ۴.

$$A(۲,۲)=A(۱,A(۲,۱))=$$
$$A(۲,۱)=A(۱,A(۲,۰))=A(۱,A(۱,۱))$$
$$A(۱,۱)=A(۰,A(۱,۰))=A(۱,۰)+۱=A(۰,۱)+۱=۲+۱=۳$$
$$\Rightarrow A(۲,۱)=A(۱,۳)=A(۰,A(۱,۲))=A(۱,۲)+۱=A(۰,A(۱,۱))+۱$$
$$=A(۰,۳)+۱=۴+۱=۵$$
$$\Rightarrow A(۲,۲)=A(۱,۵)=A(۰,A(۱,۴))=A(۱,۴)+۱=A(۰,A(۱,۳))+۱$$
$$=A(۱,۳)+۱+۱=۵+۲=۷$$

۲۴- خروجی این تابع به ازای n=۲۰ چیست؟

```
float F(int n)
{
    if (n==۱)
        return Sqrt (۱۲) ;
    else
        return Sqrt (۱۲ + F(n-۱)) ;
}
```

۱) ۴.۰ ۲) ۳.۰ ۳) ۱۲.۰ ۴) ۶/۰

پاسخ :

گزینهٔ ۱.

$$F(۲۰)=\sqrt{۱۲+F(۱۹)}=\sqrt{۱۲+\sqrt{۱۲+\sqrt{.....+\sqrt{۱۲}}}}$$

$$F^{۲}(۲۰)\approx ۱۲+F(۲۰)\Rightarrow F^{۲}(۲۰)-F(۲۰)-۱۲=۰\Rightarrow F(۲۰)\approx\frac{۱+\sqrt{۴۸+۱}}{۲}=۴$$

۲۵- الگوریتم های بازگشتی چه معایبی دارند؟

۱) اتلاف حافظه، سرعت اجرای کمتر

۲) اتلاف حافظه، طولانی بودن سورس

۳) سرعت اجرای کمتر، طولانی بودن سورس

۴) طولانی بودن سورس، اتلاف حافظه، سرعت اجرای کمتر

پاسخ :

گزینهٔ ۱.

۲۶- با توجه به دو تابع زیر $F_{۱}(۴)$ و $F_{۲}(۴)$ چیست؟

```
void F1(intX)
{
    if (X) F2(X-1);
    printf (X);
}
```

```
void F2(intY)
{
    if (Y) {
        Printf(Y+1);
        F1(Y-1);
    }
}
```

۱) به ترتیب از چپ به راست برای $F_{۱}(۴)$ خروجی ۴۴۲۲۰ و برای $F_{۲}(۴)$ خروجی ۵۳۱۱۳ است.

۲) به ترتیب از چپ به راست برای $F_{۱}(۴)$ خروجی ۲۰۲۴۴ و برای $F_{۲}(۴)$ خروجی ۱۳۳۵ است.

۳) به ترتیب از چپ به راست برای $F_{۱}(۴)$ خروجی ۴۲۲۰۴ و برای $F_{۲}(۴)$ خروجی ۵۳۳۱ است.

۴) به ترتیب از چپ به راست برای $F_{۱}(۴)$ خروجی ۴۲۰۲۴ و برای $F_{۲}(۴)$ خروجی ۵۳۱۳ است.

پاسخ :

گزینهٔ ۴.

۲۷- در روال (*routine*) بازگشتی زیر مقدار *Rec* (۵،۳) کدام است؟

```
int Rec (int P , int q)
{
        int R ;
        if (q <= ۰) return ۱ ;
        R = Rec (P , q / ۲) ;
        R = R * R ;
        if (q%۲ = = ۰)
           return R ;
        else
           return R * P ;
}
```

۱) ۱۵ ۲) ۲۵ ۳) ۷۵ ۴) ۱۲۵

پاسخ :

گزینهٔ ۴. برای محاسبهٔ $F_۱(۴)$، ابتدا باید $F_۲(۳)$ را محاسبه کرد و پس از آن ۴ گذاشت .پس به صورت $۴F_۲(۳)$ در می آید. برای$F_۲(۳)$، ابتدا ۴ می گذاریم و سپس $F_۱(۲)$ پس میشود ۴ $۴F_۱(۲)$ با ادامهٔ این روش دیده می‌شود. $F_۱ (۲) = ۲۰۲$، پس گزینهٔ چهار صحیح است.

۲۸- خروجی تابع زیر با *F*(*a*،۵) کدام است؟

```
# define Max ۱۰
int a[max];
void   F (int *a , int n)
{
        Static int i = ۰ ; int k = ۰ ;
        if (i < n) {
           a[i] = k++ ;
           printf ("%d" , a[i++] );
           F(a,n) ;
        }
}
```

۱) ۰،۱،۲،۳،۴ ۲) ۴،۳،۲،۱،۰ ۳) ۱،۲،۳،۴،۵ ۴) ۰،۰،۰،۰،۰

پاسخ :

غلط چاپی دارد.

۲۹- فرض کنید $\binom{n}{k}=\frac{n!}{(n-k)!k!}$ کدامیک از تعاریف زیر یک تعریف بازگشتی برای $\binom{n}{k}$ می‌باشد؟

۱)
$$\binom{n}{k}=\begin{cases}۱ \text{ اگر } k=۰\ ,\ k=n\\ \binom{n-۱}{k-۱}+\binom{n}{k-۱}\\ ۰ \text{ در غیر اینصورت}\end{cases}$$

۲)
$$\binom{n}{k}=\begin{cases}۱ \text{ اگر } k=۰\ ,\ k=n\\ \binom{n-۱}{k}+\binom{n}{k-۱}+\binom{n-۱}{k-۱}\\ ۰ \text{ در غیر اینصورت}\end{cases}$$

۳)
$$\binom{n}{k}=\begin{cases}۱ \text{ اگر } k=۰\ ,\ k=n\\ \binom{n-۱}{k-۱}+\binom{n-۱}{k}\\ ۰ \text{ در غیر اینصورت}\end{cases}$$

۴)
$$\binom{n}{k}=\begin{cases}۱ \text{ اگر } k=۰\ ,\ k=n\\ \binom{n-۱}{k-۱}+\binom{n-۲}{k-۲}\\ ۰ \text{ در غیر اینصورت}\end{cases}$$

پاسخ :

گزینهٔ ۳. این همان فرمول پاسکال است.

۳۰- در ضرب سه آرایه $A(۳،۴)$ و $B(۴،۶)$ و $C(۶،۲)$ به ترتیب $A*B*C$ چند عمل ضرب انجام می شود؟

۱) ۲۵ ۲) ۱۰۸ ۳) ۲۵۹۲ ۴) ۳۴۵۶

پاسخ :

گزینهٔ ۲. اگر پانتز گذاری به صورت $(AB)C$ باشد، برای محاسبهٔ تعداد ضربها داریم:

۳×۴×۶+۳×۶×۲= ۷۲+۳۶=۱۰۸

۳۱- برای یافتن یک عنصر درون آرایه N عنصری به چه تعداد مقایسه نیاز است (روش جستجوی خطی)؟

۱) N ۲) $N+۱$ ۳) $\frac{N+۱}{۲}$ ۴) $\frac{N-۲}{۲}$

پاسخ :

گزینهٔ ۱.

۳۲- افزودن یک عنصر به یک آرایه به طور متوسط چند جابه جایی نیاز دارد؟

۱) $N-1$ ۲) $\frac{N-1}{2}$ ۳) $N+1$ ۴) $\frac{N+2}{2}$

پاسخ :

سؤال گنگ است.

۳۳- برای حذف عنصر kام از یک آرایه N عنصری چند جابه جایی لازم است؟

۱) $N-K-1$ ۲) K ۳) $N-K$ ۴) $N-K+1$

پاسخ :

گزینه ۳، چون تمام جملات از k ام به بعد که تعدادشان $n-k$تا است، جابه جا می شوند.

۳۴- در یک آرایه، n عدد به ترتیب نزولی قرار دارد. اگر از روش جستجوی دودوئی برای یافتن عددی استفاده کنیم، حداکثر تعداد مقایسه چقدر خواهد بود؟

۱) $n\log_2 n$ ۲) $\lfloor \log_2 n \rfloor + 1$ ۳) $\log_2(n-1)$ ۴) $n-\log_2 n$

پاسخ :

گزینه ۲، با حل رابطه بازگشتی $T(n)=T\left(\left[\frac{n}{2}\right]\right)+1$ به گزینه ۲ می رسیم.

۳۵- مرتبه اجرایی الگوریتم جمع ماتریس ها و ضرب ماتریس ها به ترتیب از راست به چپ می شود؟

۱) $O(n^2)$, $O(n^3)$ ۲) $O(n\log n)$, $O(n^3)$

۳) $O(n^2)$, $O(n\log n)$ ۴) $O(n^2\log n)$, $O(n^2)$

پاسخ :

گزینه ۱، برای جمع ماتریس ها دو حلقه تو در تو و برای ضرب ماتریس ها سه حلقه تو در تو داریم.

۳۶- در یک جستجوی خطی حداکثر تعداد جستجو برابر کدام است؟

۱) n ۲) $n-1$ ۳) $\frac{n}{2}$ ۴) n^2

پاسخ :

گزینه ۱، در بدترین حالت، با تمام عناصر آرایه (که تعدادشان n است) مقایسه انجام می‌شود.

۳۷- کار تابع F بر روی رشته S با n کاراکتر چیست؟ تابع $Sub(S,I,n)$ تعداد n کاراکتر از موقعیت I در رشته S را بر می گرداند.

$$F(S,n)=\begin{cases} S & \text{اگر } n=1 \\ F\left(Sub(S,1,n-1),\ n-1\right)+Sub(S,n,1) & \text{اگر } n>1 \end{cases}$$

۱) رشته S را برمی گرداند.
۲) معکوس رشته S را برمی گرداند.
۳) یک کاراکتر از انتها به رشته S اضافه می کند.
۴) یک کاراکتر به ابتدای رشته S اضافه می کند.

پاسخ :

گزینه۱، مثلاً اگر $S=abc$ آنگاه:

$$F(a,b,c)=F(Sub(a,b))c=F(sub\ (a))bc=abc$$

۳۸- فرض کنید یک آرایه ۲۰۰ عنصری مرتب شده باشد. زمان اجرای بدترین، برای پیدا کردن عنصر معلوم X در آرایه A با استفاده از جستجوی دودوئی چیست؟

۱) ۲۰۰ ۲) ۸ ۳) ۷ ۴) ۱۲

پاسخ :

گزینه ۲ .

$$\lfloor \log_2^{200} \rfloor + 1 = 7 + 1 = 8$$

۳۹- رشته $ABCD$ داده شده است. این رشته چند زیر رشته دارد؟

۱) ۴ ۲) ۵ ۳) ۱۱ ۴) ۱۰

پاسخ :

گزینه ۳ .

A AB ABC $ABCD$
B BC BCD
C CD
D
$\varnothing$

۴۰- می خواهیم حاصلضرب $ABCD$ (A یک ماتریس ۵×۱۳، B یک ماتریس ۸۹×۵، C یک ماتریس ۳×۸۹ D یک ماتریس ۳۴×۳ می باشد) را پیدا کنیم به طوری که کمترین تعداد عمل ضرب انجام گیرد. ترتیب ضرب ماتریسها عبارت است از:

۱) $(A(BC))D$ ۲) $A((BC)D)$ ۳) $(AB)(CD)$ ۴) $((AB)C)D$

پاسخ :

گزینه ۱، ابتدا ۸۹ که بزرگترین عدد است را محور قرار می دهیم و BC به دست می آید. پس با محاسبه گزینه های ۱ و۲ به جواب ۱ می رسیم.

۴۱- در صورتیکه آرایه مورد جستجو در جستجوی دودوئی به صورت ۱،۰،۱،۲،۳،۴،۵،۶،۷- باشد، متوسط تعداد مقایسه ها برای جستجوی موفق چیست؟

۱) $\frac{۲۷}{۹}$ ۲) $\frac{۲۵}{۹}$ ۳) $\frac{۳۱}{۹}$ ۴) هیچکدام

پاسخ :

گزینه ۲. برای رسیدن به پنجمین عنصر یک مقایسه + برای رسیدن به دومین و هفتمین عنصر دو مقایسه + برای رسیدن به عنصرهای اول و سوم و ششم و هشتم سه مقایسه + برای رسیدن به عنصرهای چهارم و نهم چهار مقایسه لازم است. پس:

$$۱+۲\times۲+۴\times۳+۲\times۴=۲۵$$

پس میانگین آن می شود $\frac{۲۵}{۹}$.

۴۲- فرض کنید آرایه مورد جستجو توسط جستجوی دودوئی به صورت (۳۰،۲۰،۹،۷،۰،۶،۵۴،۸۲،۱۰۱-) باشد، متوسط تعداد مقایسه های مورد نیاز برای حالت جستجوی موفق چیست؟

۱) $\frac{۲۸}{۹}$ ۲) $\frac{۱۸}{۹}$ ۳) $\frac{۲۵}{۹}$ ۴) $\frac{۲۶}{۹}$

پاسخ :

گزینه ۳، روش تست ۴۱ را به کار بگیرید.

۴۳- تابع بازگشتی زیر چه می کند؟

```
int X (int a[ ], int k , int m , int n)
{
        intf ;
        if (m <= n) {
            f = (m+n)/۲ ;
            Swich (Compare(a[ p ],k ){
             Case −۱: return X (a, k, f +۱,n) ;
           Case ۰ : return f ;
           Case ۱ : return X(a,k,m , f −۱) ;
            }
         }
return−۱ ;
}
```

۱) جستجوی عنصری در آرایه
۲) مرتب کردن آرایه
۳) پیدا کردن اولین عنصر بزرگتر یا کوچکتر از عدد معین در آرایه
۴) پیدا کردن اولین عنصر کوچکتر از عددی معین، در آرایه

پاسخ :

گزینه ۱.

۴۴- A یک آرایه مرتب شده با ۱۰ عنصر می باشد، کدامیک از پاسخ های زیر زمان موفقیت متوسط برای یافتن عنصر دلخواه X در A، با استفاده از جستجوی دودوئی می باشد؟ (با این فرض که برای تعیین رابطه X با $A[i]$ تنها یک مقایسه لازم است و هر مقایسه یک واحد زمانی به طول می انجامد)

۱) ۱.۶ ۲) ۲.۹ ۳) ۴.۲ ۵) ۵.۵

پاسخ :

گزینه ۲، به توضیحات تست ۴۱ مراجعه شود.

۴۵- اگر X بیانگر یک رشته باشد، X^R نشان دهنده معکوس X است. اگر Y,X رشته باشند، حاصل $(XY)^R$ برابر است با:

۱) $X^R Y^R$ ۲) $Y^R X^R$ ۳) XY ۴) XY^R

پاسخ :

گزینه ۲، رابطه $(xy)^R = y^R x^R$ همواره برقرار است.

۴۶- اگر A آرایه ای مرتب از اعداد صحیح ۱ الی ۱۰۲۴ باشد الگوریتم جستجوی دودوئی با چند بار تکرار عدد ۴ را پیدا می کند.

۱) ۸ ۲) ۷ ۳) ۹ ۴) ۱۰

پاسخ :

گزینه ۱ ابتدا با ۵۱۲، سپس با ۲۵۶، سپس با ۱۲۸، سپس با ۶۴، سپس با ۳۲، سپس با ۱۶، سپس با ۸، و نهایتاً با ۴ مقایسه می شود.

۴۷- می خواهیم چهار ماتریس A(۱×۳۰) و B (۳۰×۱) و C(۱×۴۰) و D (۴۰×۱۰) را در هم ضرب کنیم ($A\times B\times C\times D$) حداقل ممکن تعداد عمل ضرب عناصر این ماتریس ها چند تا است؟

۱) ۱۲۵۰ ۲) ۱۴۰۰ ۳) ۲۰۷۰۰ ۴) ۱۱۷۵۰

پاسخ :

گزینه ۲، در اینجا، ترتیب $(A((BC)D))$ بهینه است که تعداد ضرب ها مورد نیاز برای محاسبه آن برابر است با ۱۴۰۰.

۴۸- اگر $Pat = 'abcabcacab' = P_0P_1 \ldots P_{n-1}$ باشد و داشته باشیم:

$$F'(j) = \begin{cases} P_0P_1 \ldots P_k = P_{j-k}P_{j-k+1} \ldots P_j & k<j \text{ بزرگترین} \\ -1 & \text{در غیر اینصورت} \end{cases}$$

مقادیر تابع F برای Pat برابر است با ...

۱) ۱، ۰، ۰، ۳، ۲، ۱، ۰، ۰، ۰، ۱-
۲) ۱، ۰، ۰، ۳، ۲، ۱، ۰، ۰، ۰، ۰
۳) ۱، ۱، ۱، ۱، ۱، ۱، ۰، ۰، ۰، ۰
۴) ۱، ۰، ۱-، ۳، ۲، ۱، ۰، ۱-، ۱-، ۱-

پاسخ :

۴۹- مینیمم و ماکزیمم اعداد ذخیره شده در یک آرایه یک بعدی با n خانه ، با چند مقایسه بین اعداد ذخیره شده در این خانه‌ها بدست خواهد آمد؟

۱) $\frac{3n}{2}$
۲) $\frac{3n}{2} - 2$
۳) $\frac{n}{2}$
۴) $\frac{n+1}{2}$

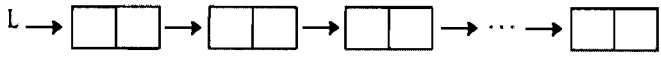
پاسخ :

۵۰- اگر $a=2, b=4, c=8, d=10$ باشد ، ارزش عبارت پسوندی $ab*c+da-/$ چیست؟

۱) ۲-
۲) ۱-
۳) ۱
۴) ۲

پاسخ :

۵۱- لیست خطی یک طرفه و با سر لیست L داده شده است :



فرض کنید :

```
typelist =↑cell type;
cell type=record
        element:int;
        next : List ;
end;
```

تابع زیر چه کاری انجام می‌دهد ؟

```
Function F(L:list):List;
{
  if(L==NULL)OR(L↑.next==NULL)
      F=L;
  else{
      F=F(L↑.next);
      L↑.next↑.next=NULL;
      L↑.next:=NULL;
    }
}
```

۱) لیست L را معکوس می‌کند
۲) در لیست L تغییری نمی‌دهد
۳) لیست L را مرتب می‌کند
۴) هیچکدام

پاسخ :

۵۲- مزیت لیست پیوندی نسبت به آرایه چیست ؟

۱) مصرف حافظه کمتر
۲) ساده‌تر بودن عملیات حذف و درج
۳) سریعتر بودن عمل پیمایش
۴) سریعتر بودن عمل جستجو

پاسخ :

۵۳- یک لیست خطی یکطرفه با دو اشاره‌گر F و R که به ترتیب اول و آخر لیست اشاره می‌کنند پیاده‌سازی شده است ؟ هزینه کدامیک از اعمال زیر وابسته به تعداد عناصر لیست است ؟

۱) حذف اولین عنصر
۲) حذف آخرین عنصر
۳) درج یک عنصر در انتهای لیست
۴) درج یک عنصر در ابتدای لیست

پاسخ :

۵۴- صف اولویت دار (*Priority Queue*) شامل کدامیک از اعمال زیر می‌شود ؟

۱) جستجو، درج، حذف کوچکترین عنصر
۲) درج ، حذف کوچکترین عنصر
۳) جستجو، درج، حذف
۴) درج ، حذف

پاسخ :

۵۵- ماکزیمم تعداد $NODE$ در یک دوتایی با ارتفاع h برابر است با:

۱) $۲^{n+۱}$ ۲) $۲^{n+۱}+۱$ ۳) $۲^{n+۱}-۱$ ۴) $۲^{n}+۱$

پاسخ :

گزینه ۳، ریشه ۱ رأس دارد که همان $۲^{۰}$ است. سطر اول $۲^{۱}$, ... و سطر nام ، ۲^{n} رأس دارد. پس در مجموع برابر است با:

$$۲^{۰}+۲^{۱}+۲^{۲}+...+۲^{n}=۲^{n+۱}-۱$$

۵۶- رویه زیر را برای یک درخت دودوئی در نظر بگیرید:

```
Function Count (tree: Pointer) : integer ;
{
    if (tree = = NULL)
        count = ۰ ;
    else if (tree -> left = = NULL && tree -> right = = NULL)
        Count = ۱ ;
    else
        Count = Count (tree -> left) + Count (tree -> right)
}
```

۱) تعداد گره های یک درخت دودوئی را محاسبه می کند.

۲) تعداد برگهای یک درخت دودوئی را محاسبه می کند.

۳) تعداده گره هایی که دارای دو فرزند می باشند را محاسبه می کند.

۴) تعداد گره هایی که دارای یک فرزند می باشند را محاسبه می کند.

پاسخ :

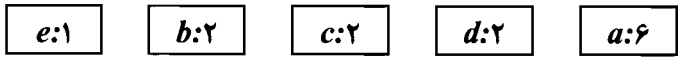
گزینه ۱، با اجرای الگوریتم روی یک درخت کوچک، این گزینه تأیید می شود.

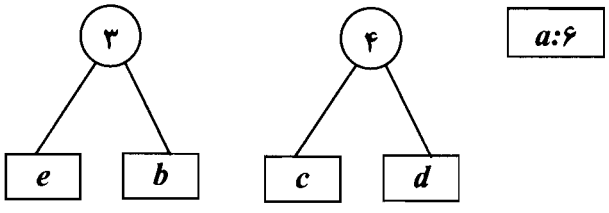
۵۷- رشته متنی ($abaabacadcade$) را در نظر بگیرید. کد هافمن حاصل برای هر یک از نویسه ها برابر است با.....

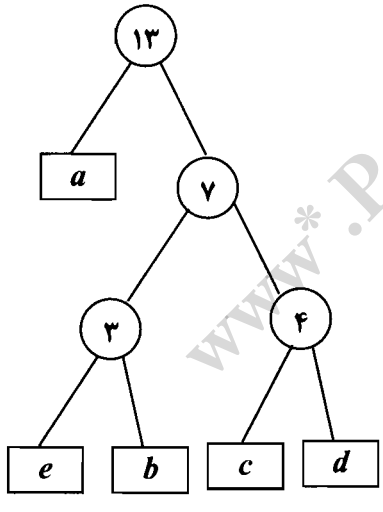
۱)	۲)	۳)	۴)
$a=۱$	$a=۰$	$a=۰۰$	$a=۰۰$
$b=۰۱$	$b=۱۰۰$	$b=۰۰۱$	$b=۰۱$
$c=۰۰۱$	$c=۱۰۱$	$c=۰۱۰$	$c=۱۰$
$d=۰۰۰۱$	$d=۱۱۰$	$d=۰۱۱$	$d=۱۱$
$e=۰۰۰۰$	$e=۱۱۱$	$e=۱۰۰$	$e=۱۰۰$

پاسخ :

هیچکدام از گزینه ها درست نیست.







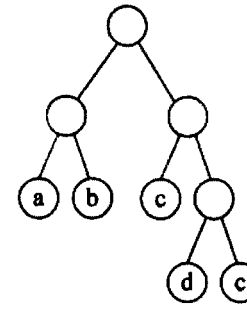
$$\Rightarrow \quad \begin{aligned} a &= ۰ \\ b &= ۱۰۱ \\ c &= ۱۱۰ \\ d &= ۱۱۱ \\ e &= ۱۰۰ \end{aligned}$$

۵۸- حروف a,b,c,d,e با جدول فراوانی زیر داده شده است:

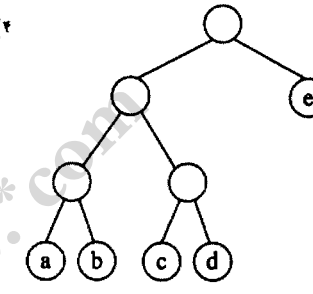
حروف	a	b	c	d	e
فراوانی	۰/۰۵	۰/۱	۰/۲۵	۰/۲۸	۰/۳۲

درخت هافمن وابسته به این حروف به قرار ..... است.

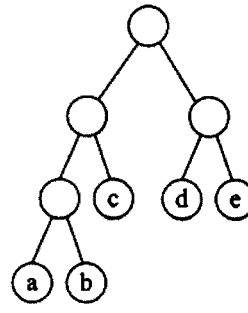
۱)



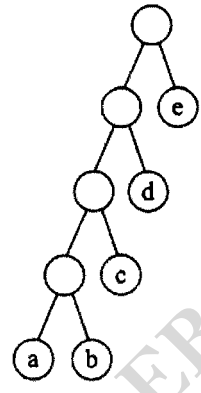
۲)



۳)

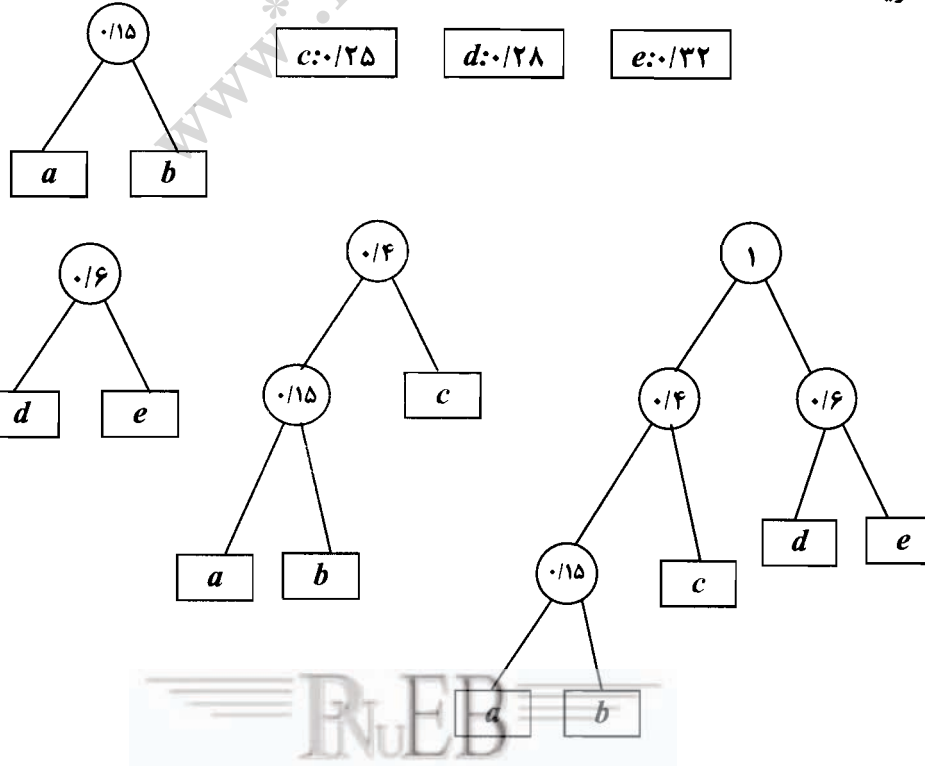


۴)



پاسخ :

گزینه ۳



۵۹- یک درخت دودویی جستجوی T با۵ گره با برچسب های $a_1 < a_2 < a_3 < a_4 < a_5$ را در نظر بگیرید. به گره a_i عدد X_i را نسبت می دهیم. فرض کنید X_1=۲ و X_i=۱ برای i=۲...۵ می خواهیم T را طوری بسازیم که عبارت $C_T = \sum_{i=1}^{5} X_i\big(depth(a_i)+1\big)$ مینیمم شود. $depth(a_i)$ عمق گره a_i در T است. کمترین مقدار C_T چقدر است؟

۱) ۱۳ ۲) ۱۴ ۳) ۱۵ ۴) ۱۶

پاسخ :

گزینه ۲.

۶۰- ساختمان داده ای را بر روی مجموعه A از اعداد صحیح در نظر بگیرید که اعمال درج (*Insert*) حذف (*Delete*) و پیدا کردن نزدیکترین (*Find closest*) را فراهم می آورد. منظور از *Find closest (X)* پیدا کردن $Y \in A$ است که $|X-Y|$ نسبت به بقیه Yها کمینه باشد. فرض کنید که T بیشترین زمان اجرای اعمال فوق در بدترین حالت باشد. در این صورت کدام ساختمان داده کمترین مقدار T را خواهد داشت؟

۱) *Heap* ۲) لیست نامرتب

۳) لیست مرتب ۴) درخت دودویی جستجوی متوازن

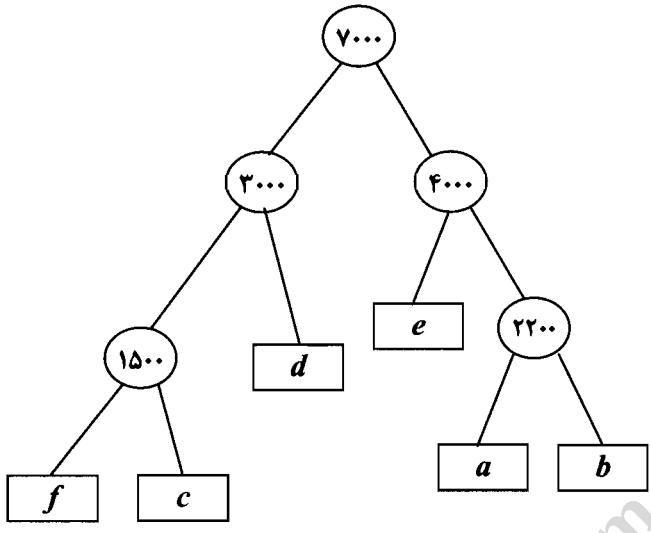
پاسخ :

۶۱- متنی شامل ۷۰۰ حرف از حروف f,d,c,b,a با تکرار a=۱۰۰۰، b=۱۲۰۰، c=۸۰۰، d=۱۵۰۰، e=۱۸۰۰ و f=۷۰۰ موجود است. چنانچه کدی بهینه برای حروف بالا انتخاب نماییم. تعداد کل بیتهای لازم برای تبدیل متن مذکور به مجموعه ای از بیتها چقدر است؟

۱) ۱۴۶۰۰ ۲) ۱۷۷۰۰ ۳) ۲۴۳۰۰ ۴) ۳۵۲۰۰

پاسخ :

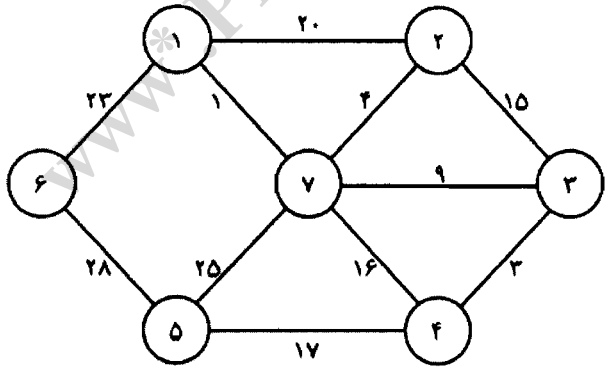
گزینه ۲، ابتدا الگوریتم هافمن را اعمال می کنیم و به درخت زیر می رسیم:



پس f,c,ba هر کدام سه رقم دارند و d,e هر کدام دو رقم. پس تعداد کل برابر است با:

$$۳(۱۰۰۰+۱۲۰۰+۸۰۰+۷۰۰)+۲(۱۵۰۰+۱۸۰۰)=۱۷۷۰۰$$

۶۲- هزینه درخت پوشای مینیمم (*Minimum Spanning Tree*) گراف زیر چیست؟



۱) ۶۸ ۲) ۴۱ ۳) ۸۱ ۴) ۵۷

پاسخ :

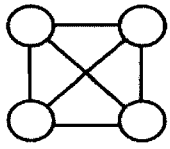
گزینه ۴، درخت مطلوب عبارت است از:

$$T=\{(۱,۷),(۷,۲),(۷,۳),(۳,۴),(۴,۵),(۱,۶)\}$$

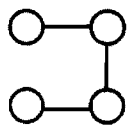
وزن آن عبارت است از:

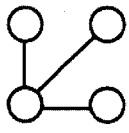
$$۱+۴+۹+۳+۱۷+۲۳=۵۷$$

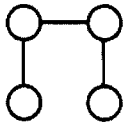
۶۳- گراف زیر داده شده است:



کدامیک از انتخاب های زیر *Spanning* این گراف است؟

۱) 

۲) 

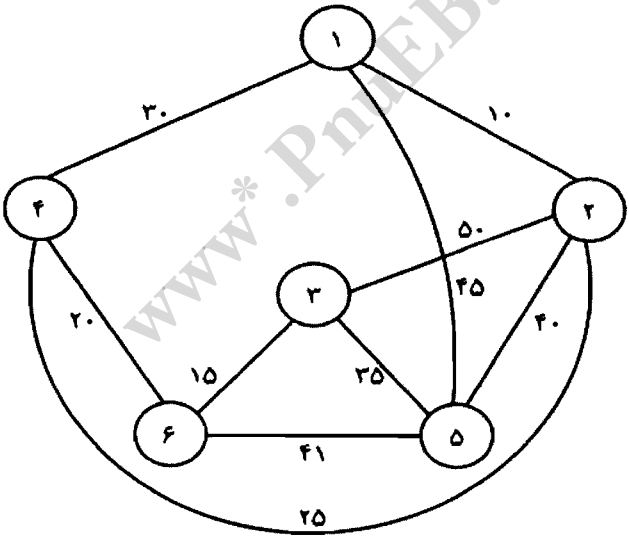
۳) 

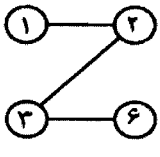
۴) هر سه

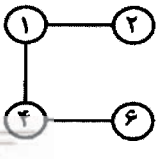
پاسخ :

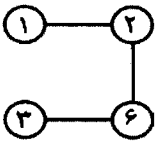
گزینه ۴. طبق تعریف *spanning tree*.

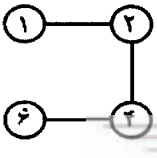
۶۴- اگر برای پیدا کردن درخت پوشای مینیمم از الگوریتم *Prim* استفاده کنیم کدامیک از گزینه‌های زیر درخت حاصله در انتهای مرحله سوم این الگوریتم را به ما می دهد؟



۱) 

۲) 

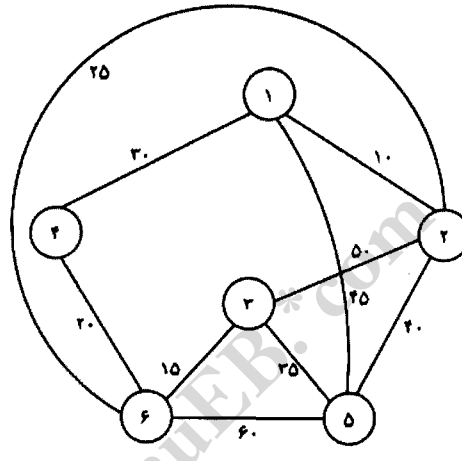
۳) 

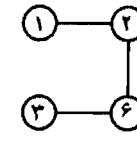
۴) 

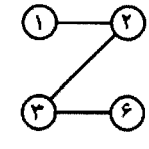
پاسخ :

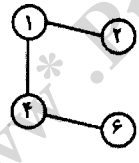
گزینه ۴، با اجرای الگوریتم پرایم تا سه مرحله، ابتدا (۲و۱) پس (۴و۲) و بعد (۶و۴) انتخاب می‌شوند.

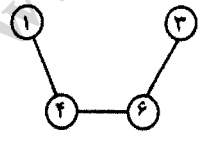
۶۵- اگر برای پیدا کردن درخت پوشای مینیمم گراف زیر از الگوریتم *Prim* استفاده کنیم کدامیک از گزینه های زیر درخت حاصله در انتهای مرحله سوم این الگوریتم را به ما می دهد؟



۱)


۲)


۳)


۴)


پاسخ :

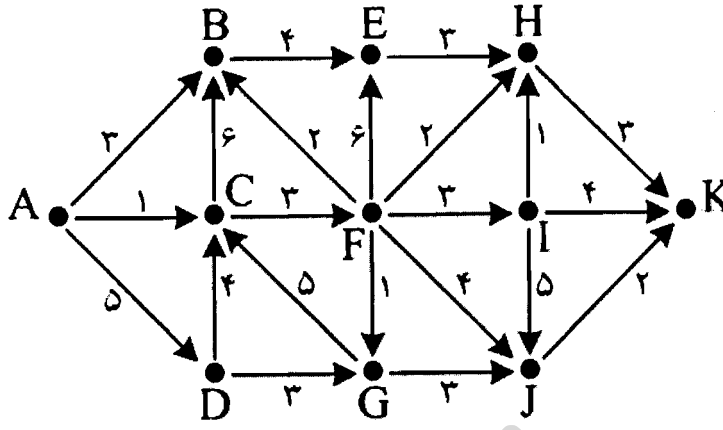
گزینه ۱، به ترتیب یالهای (۱،۲)، پس (۲،۶) و بعد (۶،۳) انتخاب می شوند.

۶۶- فضای مورد نیاز برای نمایش یک گراف *G(V,E)* به روش لیست همسایگی (*Adjacency List*) کدام است؟

۱) $O(|E|+|V|)$ ۲) $O(|E|)$ ۳) $O(|V|)$ ۴) $O(|E|.|V|)$

پاسخ :

۶۷- با توجه به گراف زیر کوتاهترین مسیر از A به K دارای چه طولی است؟



۱) ۸ ۲) ۹ ۳) ۱۰ ۴) ۱۱

پاسخ :

گزینه ۱، مسیر فوق چنین است: $A \to C \to F \to H \to K$

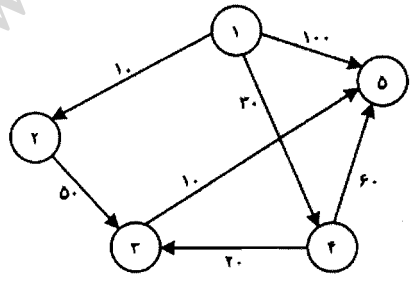
۶۸- برای گراف سؤال قبلی، درخت پوشای مینیمم آن دارای چه وزنی است؟

۱) ۱۸ ۲) ۱۹ ۳) ۲۰ ۴) ۲۱

پاسخ :

گزینه ۳، با اجرای الگوریتم پرایم، جواب فوق تأیید می شود.

۶۹- در گراف زیر کمترین هزینه از گره ۱ به ۵ دارای مسیری است با طول:



۱) ۱ ۲) ۲ ۳) ۳ ۴) ۴

پاسخ :

گزینه ۳، مسیر فوق عبارت است از: $۱ \to ۴ \to ۳ \to ۵$

۷۰- پیچیدگی کدامیک از الگوریتمهای مرتب کننده زیر (برحسب تابعی از اندازه ورودی) در حالت متوسط (*Average Case*) و در بدترین حالت (*Worst Case*) با هم متفاوت است؟

۱) *Quick Sort*　　۳) *Heap Sort*

۲) *Binary Insertion Sort*　　۴) *Merge Sort*

پاسخ :

گزینه ۱، در بدترین حالت n^2 و در حالت میانگین $nlogn$ است.

۷۱- آرایه A شامل n عنصر مرتب (از اندیس ۱ تا n) و k عنصر نامرتب (از اندیس $n+1$ تا $n+k$) است. کدام یک از الگوریتمهای زیر برای مرتب سازی A کمترین تعداد مقایسه را دارد؟ فرض کنید k مستقل از n و بسیار کمتر از آن است.

۱) *Insertion Sort*　　۳) *Quick Sort*

۲) *Quick Sort*　　۴) *Merge Sort*

پاسخ :

گزینه ۱، چون در مرتب سازی درجی، n عنصر مرتب، به هم ریخته نمی شوند.

۷۲- یک ماتریس $N \times M$ از اعداد صحیح دلخواه داده شده است. الگوریتم زیر را بر روی این ماتریس انجام می دهیم:

الف) هر کدام از سطرهای ماتریس را مستقلاً از چپ به راست به صورت صعودی مرتب می کنیم.

ب) هر کدام از ستون های ماتریس را مستقلاً از بالا به پایین به صورت صعودی مرتب می کنیم.

کدامیک از گزینه های زیر درست است؟

۱) سطرها لزوماً مرتب نیستند ولی اگر یک بار سطرها را مرتب کنیم همه سطرها و هم ستونها مرتب می‌شوند.

۲) سطرها لزوماً مرتب نیستند، ولی اگر یک بار دیگر هم سطرها را مرتب کنیم، ستونها لزوماً مرتب نمی‌شوند.

۳) سطرها و ستونها هر کدام به صورت صعودی مرتب می شوند.

۴) تکرار این الگوریتم لزوماً هم سطرها و هم ستونها را مرتب نمی کند.

پاسخ :

گزینه ۳،

۷۳- الگوریتم *Quick Sort* یک رشته nتایی را با چه سرعتی مرتب می کند؟

۱) $O(n \log n)$　　۲) $O(n)$　　۳) $O(n^2)$　　۴) $O(\log_2 n)$

پاسخ :

گزینه ۱

۷۴- اگر یک لیست مرتب شده با n خانه را با استفاده از الگوریتم *Binary Search* برای یک مقدار خاص جستجو کنیم، تعداد دفعات مقایسه چه خواهد بود؟

۱) $O(n/2)$ ۲) $O(n^2)$ ۳) $O(\log n)$ ۴) $O(\log_2 n)$

پاسخ :

گزینه ۴، البته گزینه ۳ هم صحیح است، چون همه لگاریتمها، هم مرتبه اند.

۷۵- متوسط زمانی که نیاز است یک جستجوی ترتیبی موفق برای یک عنصر از یک آرایه n تایی انجام شود عبارت است از:

۱) $\frac{n+1}{2}$ ۲) $\frac{n(n+1)}{2}$ ۳) $\log_2 n$ ۴) n^2

پاسخ :

گزینه ۱، برای جستجوی عنصر اول، یک مقایسه و برای جستجوی عنصر دوم، دو مقایسه و.... و برای جستجوی عنصر nام، n مقایسه لازم است. پس میانگین آنها می شود:

$$\frac{1}{n}(1+2+3+\ldots+n)=\frac{n(n+1)}{2n}=\frac{n+1}{2}$$

۷۶- در کدامیک از حالات زیر زمان اجرای الگوریتم *Quick Sort* برای منظم کردن یک لیست (از کوچک به بزرگ) بیشتر از حالات دیگر زمان نیاز دارد؟

۱) لیست کاملاً نامنظم باشد.

۲) لیست از بزرگ به کوچک منظم باشد.

۳) لیست از کوچک به بزرگ منظم و مرتب باشد.

۴) *Quick Sort* در همه حالات زمان بسیار سریع و ثابتی دارد.

پاسخ :

گزینه۲، این یکی از نکاتی است که در متن کتاب به آن اشاره شده است.

۷۷- برای مرتب سازی یک آرایه Nتایی به روش *Bubble=Sort* حداکثر چند تعویض لازم است؟

۱) $\frac{N(N-1)}{2}$ ۲) $\log_2(N-1)$ ۳) $\frac{N^2}{4}+3(N-1)$ ۴) $\frac{N^2-3N-4}{2}$

پاسخ :

گزینه ۱، $1+2+3+\ldots+(N-1)=\frac{N(N-1)}{2}$

۷۸- کدام کار روی همه ساختارهای داده ای انجام می شود؟

۱) ادغام (*Merge*)
۲) پردازش (*Process*)
۳) مرتب سازی (*Sort*)
۴) درج (*Insert*)

پاسخ :

گزینه ۴، طبق تعریف، گزینه ۴ صحیح است.

۷۹- نماد $O(\log n)$ نشان دهندگی پیچیدگی کدام الگوریتم است؟

۱) جستجوی دودویی (*Binary Search*)
۲) جستجوی خطی (*Linear Search*)
۳) مرتب سازی حبابی (*Bubble Sort*)
۴) مرتب سازی سریع (*Quick Sort*)

پاسخ :

گزینه ۱، جستجوی خطی $O(n)$ و دو مرتب سازی دیگر $O(n^2)$ می باشند.

۸۰- بدترین حالت در روش مرتب سازی سریع (*Quick Sort*) چیست؟

۱) عناصر لیست به ترتیب معکوس باشند.
۲) عناصر لیست از قبل مرتب باشند.
۳) یک نیمه لیست مرتب باشد.
۴) یک نیمه لیست معکوس باشد.

پاسخ :

به تست ۷۶ مراجعه شود.

۸۱- اگر آرایه مرتب A دارای r عنصر و آرایه مرتب B دارای P عنصر باشد، حداکثر تعداد مقایسه برای ترکیب (*Merge*) دو آرایه کدام است؟

۱) $\frac{r+P}{2}$
۲) $r+P$
۳) $Max(r,P)$
۴) $\log r * P$

پاسخ :

این حالت موقعی اتفاق می افتد که اولین عضو A بین $B[1]$, $B[2]$ باشد و آخرین عضو A از آخرین عضو B بزرگتر باشد.

۸۲- اگر N رکورد داشته باشیم تعداد کل مقایسه در روش *Linear Insertion Sort* برابر است با:

۱) $\frac{N^2}{4}$
۲) $\frac{N(N-1)}{2}$
۳) $\frac{N(N-1)}{4}$
۴) $\frac{N^2}{4}$

پاسخ :

گزینه ۲، $1+2+3+\ldots+(n-1)=\frac{n(n-1)}{2}$

۸۳- فرض کنید که لیست حروف زیر (n=۱۱) برای مرتب کردن با استفاده از *Quick sort* داده شده است. تعداد مقایسه ها دارای چه *order* است.

a b c d e f g h i j k

۱) $O(n)$ ۲) $O(n\log_2 n)$ ۳) $O(n^3)$ ۴) $O(o\log_2 n)$

پاسخ :

گزینه ۲.

۸۴- اگر در الگوریتم مرتب سازی سریع (*Quick sort*) به ترتیب صعودی، عنصر لولا (*Pivot*) را همان عنصر اول لیست بگیریم و با استفاده از آن یک بار لیست مرتب نزولی و یکبار دیگر لیست مرتب صعودی را مرتب کنیم گزینه صحیح برای تعداد عملیات اصلی (مقایسه و جابجایی) را در این دو حالت انتخاب کنید؟

۱) هر دو حالت از $O(n\log n)$

۲) هر دو حالت از $O(n^2)$

۳) برای لیست صعودی $O(n)$ و برای لیست نزولی $O(n^2)$

۴) برای لیست صعودی $O(n\log n)$ و برای لیست نزولی $O(n^2)$

پاسخ :

گزینه ۳ .

۸۵- لیست زیر را در نظر بگیرید. اگر عنصر اول لیست یعنی عدد ۹ را به عنوان لولا (*Privot*) اختیار کنیم کدام یک از گزینه های زیر می توانند خروجی مرحله اول الگوریتم مرتب سازی سریع (*Quick sort*) باشد؟

۱) (۷،۸،۹،۱۰،۳،۶،۱۵)

۲) (۷،۸،۹،۳،۶،۱۰،۱۵)

۳) (۶،۳،۸،۷،۹،۱۵،۱۰)

۴) (۶،۷،۸،۹،۳،۱۰،۱۵)

پاسخ :

سؤال ناقص است.

۸۶- N گلوله با وزن های مختلف را می خواهیم با یک ترازوی دو کفه ای بدون وزنه و با توزین های متوالی مرتب کنیم (یک توزین عبارت است از قرار دادن دو گلوله در دو کفه ترازو و مقایسه وزنهای آنها). کدام یک از گزینه های زیر درست است؟

۱) ۳ گلوله را می توان حداکثر با ۲ بار توزین مرتب کرد.

۲) ۴ گلوله را می توان حداکثر با ۴ بار توزین مرتب کرد.

۳) ۴ گلوله را می توان در مواردی با ۳ بار توزین مرتب کرد.

۴) هیچکدام

پاسخ :

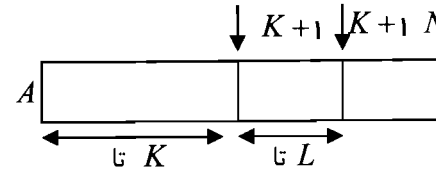
گزینه ۳، هر گاه گلوله ۱ از ۲، و ۲، از ۳، و ۳ از ۴ سنگینتر باشد.

۸۷- الگوریتم زیر برای مربت سازی صعودی آرایه A به N عنصر (با اندیس هـای از ۱ تـا N) پیـشنهاد شده است. در این الگوریتم، فرض کنید $Sort\ (A,I,j)$ عناصر i تا j از آرایه A را با یکی از الگوریتمهای شناخته شده به صورت صعودی مرتب می کند.

$Sort(A, K+1, N)$

$Sort(A, 1, K+L)$

$Sort(A, K+1, N)$



کدامیک از گزینه های زیر برای هر n صحیح است؟

۱) تنها اگر $K \le L$ باشد این الگوریتم درست است.

۲) تنها اگر $K>L$ باشد این الگوریتم درست است.

۳) تنها اگر $N = 3K = 3L$ باشد این الگوریتم درست است.

۴) تنها اگر $K=L$ باشد الگوریتم درست است.

پاسخ :

صورت سؤال غلط است .

۸۸- اگر یک لیست مرتب شده (با n خانه) را با استفاده از الگوریتم *Binary Search* برای یک مقدار خاص جستجو کنیم تعداد دفعات مقایسه چه خواهد بود؟

۱) $O(\log n)$ ۲) $O(n^2)$ ۳) $O(\log_2 n)$ ۴) $O\left(\frac{n}{2}\right)$

پاسخ :

گزینه ۳، البته گزینه ۱ نیز صحیح است.

۸۹- تابع $A(M,N)$ را به شکل زیر در نظر بگیرید. حاصل $A(1,3)$ کدام است؟

$$A(M,N)\begin{cases} N+1 & \text{اگر } M=0 \\ A(M-1,1) & \text{اگر } N=0 \\ A\big(M-1, A(M,N-1)\big) & \text{بقیه حالات} \end{cases}$$

۱) ۳ ۲) ۴ ۳) ۵ ۴) ۶

پاسخ :

این همان تست شماره ۸ است.

۹۰- اگر B,A دو عدد صحیح نامنفی باشند و تابع M به صورت زیر تعریف شده باشد، حاصل $M(۲۰,۲۸)$ چیست؟

$$M(A,B)=\begin{cases}M(B,A) & if\ A<B\\ A & if\ B=۰\\ M\big(B,MOD(A,B)\big) & else\end{cases}$$

۱) ۸ ۲) ۴ ۳) صفر ۴) ۱

پاسخ :

گزینه ۲، چون

$$M(۲۰,۸)=M(۲۸,۲۰)=M(۲۰,۸)=M(۸,۴)=M(۴,۰)=۴$$

۹۱- خروجی کدام الگوریتم، درخت پوشا برای گراف نیست؟

۱) *Prim* ۲) *Dijkestra* ۳) *Kruskal* ۴) *BFS*

پاسخ :

گزینه ۲.

۹۲- برای محاسبه $A_{m\times n}\times B_{n\times P}$ چند عمل جمع لازم است؟

۱) $mn^{۲}p$ ۲) n ۳) mn ۴) mp

پاسخ :

گزینه ۳.

۹۳- برای ادغام (*merge*) دو آرایه مرتب شده B,A که به ترتیب n,m عنصری هستند حداکثر چند مقایسه لازم است؟

۱) $n+m$ ۲) $n+m-۱$ ۳) $max(n,m)$ ۴) $\frac{(m+n)(m+n+۱)}{۲}$

پاسخ :

گزینه ۱، به توضیحات تست ۸۱ مراجعه شود.

۹۴- کدامیک از عبارات زیر غلط است؟

۱) $۱۰^{n}+n^{۲۰}\notin\theta(n^{n})$ ۳) $(\log_{۲}n)!\in\Omega(n!)$

۲) $\log_{L}^{n}\in\theta(\log_{۱۰}^{n})$ ۴) $۴n^{۳}+۵n^{۲}+۷n\in\Omega(\log_{۲}^{n})$

پاسخ :

گزینه ۳، چون $(\log_{۲}^{n})!\in O(n!)$

۹۵- تابع زیر را در نظر بگیرید:

```
int F (int x)
{
    if (x < ۱)
        return ۱ ;
    else return F(x − ۱) + g(x) ;
}
int g (int x)
{
    if (x < ۲)
        return ۱ ;
    else return F۹x − ۱) + g(x/۲) ;
}
```

کدام گزینه بهترین نمایش برای رشد $F(n)$ بر حسب n است؟

۱) خطی ۲) نمایی ۳) درجه ۲ ۴) لگاریتمی

پاسخ :

گزینه ۲، چون محاسبه $F(x)$ مشروط می شود به دو بار محاسبه $F(x-۱)$. پس

$$F(x) = ۲F(x-۱) \Rightarrow F(x) \in \theta(۲^n)$$

۹۶- N یک عدد صحیح ۱۶ بیتی است که در متغیر صحیح A قرار می گیرد. A که هدف آن بررسی اول (*Prime*) بودن متغیر A می باشد دارای چه درجه ای است؟

۱) N ۲) $N^۲$ ۳) $\log N$ ۴) نمایی

پاسخ :

گزینه ۱، چون n را بر اعداد $۱,۲,۳,...,\sqrt{n}$ تقسیم می کنیم و رشد $\sqrt{n}$ نسبت به n، خطی است.

۹۷- اگر الگوریتم جستجوی دودویی را برای جستجوی عناصر آرایه $A[۱...۸] = \{۵,۱\ ۰,۱\ ۵,۲\ ۰,۲\ ۵,۳\ ۰,۳\ ۵,۴\ ۰\}$ به کار ببریم: میانگین تعداد مقایسه ها برای جستجوی موفق تقریباً، کدام است؟

۱) ۲/۲ ۲) ۲/۴ ۳) ۲/۶ ۴) ۲/۸

پاسخ :

گزینه ۲، به توضیحات تست ۴۱ مراجعه شود.

$$۱+۲\times۲+۳\times۴+۴=۲۱\Rightarrow\frac{۲۱}{۸}=۲/۴$$

۹۸- یک ماشین انتزاعی که در مبنای ۱۰ کار می کند اعداد را به راحتی با یکدیگر جمع و تفریق می کند و ضرب در اعداد ۱۰^{x} را از طریق *Shift* به راحتی عمل می کند، اما در ضرب تنها قادر است اعداد یک رقمی را در هم ضرب نماید. به نظر شما برای ضرب دو عدد ۱۲۳۴*۵۶/۸ حداقل تعداد عملیات ممکن چه تعداد می باشد تا حاصل ضرب فوق به دست آید؟

۱) ۸ ۲) ۱۲ ۳) ۱۳ ۴) ۱۶

پاسخ :

گزینه ۲، چون با ۳ ضرب تبدیل می شود به ۳ تا ضرب ۲ رقم در ۲ رقم، و مجدداً برای محاسبه هر کدام از آنها نیاز به ۳ ضرب داریم. پس در مجموع ۱۲ ضرب نیاز داریم.

۹۹- فرض کنید $T(n)$ برای تعداد پرانتزبندی های مختلف برای ضرب کردن n ماتریس در هم باشد. در این صورت: $T(۱)=T(۲)=۱$:

۱) $T(n)=\sum_{i=۱}^{n}T(i)\times T(n-i)$

۲) $T(n)=\sum_{i=۱}^{n-۱}T(i)\times T(n-i)$

۳) $T(n)=\sum_{i=۱}^{n}T(i)\times T(n-i+۱)$

۴) $T(n)=\sum_{i=۱}^{n-۱}T(i)\times T(n-i+۱)$

پاسخ :

گزینه ۲، این همان عدد کاتالان است.

۱۰۰- کدام گزینه در مورد الگوریتمهای کروسکال و پرایم برای ایجاد درخت پوشای کمینه درست است؟

۱) زمان اجرای هر دو الگوریتم روی گرافهای یکسان، مساوی است.

۲) هر دو الگوریتم روی گرافهای یکسان، درخت پوشای یکسان تولید می کنند.

۳) مجموع طول اضلاع درخت پوشا در هر دو الگوریتم یکسان است.

۴) هر دو الگوریتم با رشد و به هم پیوستن یک جنگل از درختها درخت پوشا را تولید می کنند.

پاسخ :

گزینه ۳، چون هر دو الگوریتم، درخت پوشای کمینه تولید می کنند، و مجموع طول اضلاع درختهای کمینه با هم برابر است.

۱۰۱- پنج فایل مرتب شده به سایزهای ۵، ۱۰، ۲۰، ۲۵و ۳۰ داریم. می خواهیم از ادغان دو به دو آنها یک فایل مرتب شده واحد شامل همه رکوردها به دست آوریم در هر ادغام رکوردهای فایل های ورودی ممکن است چند بار از یک فایل خوانده و در یک فایل دیگر نوشته شوند به هر کدام از این نوشتن و خواندن یک جابه جایی می گوییم. حداقل تعداد کل این جابه جایی برای ادغام همه فایل ها چقدر است؟

۱) ۱۹۵ ۲) ۲۰۰ ۳) ۱۸۵ ۴) ۲۱۵

پاسخ :

گزینه ۲

$۵+۱۰=۱۵$

$۱۵+۲۰=۳۵$

$۳۵+۲۵=۶۰$

$۶۰+۳۰=۹۰$

$۹۰+۶۰+۳۵+۱۵=۲۰۰$

۱۰۲- حداکثر ول یک کد برای n عنصر که به روش هافمن کدگذاری می شوند کدام است؟

۱) $n-۲$ ۲) $\lceil \log n \rceil$ ۳) $n-۱$ ۴) $\left\lceil \frac{n}{۲} \right\rceil$

پاسخ :

گزینه ۳ .

۱۰۳- آرایه $S[۱..n]$ از اعداد صحیح داده شده است با داشتن k می خواهیم I و jای را پیدا کنیم که $S[i]+S[j]=k$ برای این کار الگوریتم زیر پیشنهاد شده است؟

```
i ← ۱ , j ← n ;
while i ≤ j do
{
    if S[i]+S[j]=k then
        return ij
    if S[i]−S[j]<k then
        i ← i+L
    if S[i]+S[j]>k then
        j ← j−۱
}
return not − Possible
```

۱) همواره درست کار می کند.
۲) اگر S به صورت نزولی مرتب شده باشد درست کار می کند.
۳) اگر S به صورت صعودی مرتب شده باشد درست کار می کند.
۴) در هر صورت برای هر حالت ورودی ممکن است هیچگاه جواب پیدا نشود.

پاسخ :

به نظر می رسد الگوریتم شامل غلط چاپی باشد.

۱۰۴- برای به دست آوردن بزرگترین و کوچکترین عنصر در یک لیست غیر مرتب بهترین الگوریتمی که می توان ارائه داد چند مقایسه خواهد داشت (برای n حالت خروجی)؟

۱) $2(n-1)$ ۲) $\frac{3n}{2}-2$ ۳) $\frac{n-1}{2}$ ۴) $(n-1)+(n-2)$

پاسخ :

اگر منظور بدترین حالت است گزینه ۴ و اگر منظور حالت میانگین است گزینه ۲ صحیح است. صفحه ۱۱۷ کتاب ملاحظه شود.

۱۰۵- الگوریتم زیر برای پیدا کردن اندیس های کوچکترین و بزرگترین عناصر در یک آرایه N تایی A داده شده است:

```
Min ← 1 , max ← 1;
For i = 2 to N do
    Choose B = True or B = False with equal probablities
    if B then
        if A[i] < A[min] then
            min ← i;
        else if A[i] > A[max] then
            max ← i;
        elseif A[i] < A[min] then  min ← i;
```

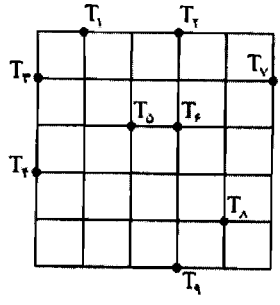
متوسط تعداد مقایسه های دو عنصر از A چندتاست؟

۱) n ۲) $2(n-1)$ ۳) $\frac{3n}{2}$ ۴) $\frac{3(n-1)}{2}$

پاسخ :

به نظر می رسد الگوریتم شامل غلط چاپی باشد.

۱۰۶- یک مهندس برق مداری را طراحی نموده است دارای ۹ ترمینال، که باید ولتاژی معادل ۵ ولت به آنها متصل گردد فرض کنید ولت به یکی از ترمینال ها وصل است. برای اینکه کمترین سیم بندی در مدار به کار رود بگویید حداقل چند واحد سیم لازم دارد؟ فاصله هر سطر و ستون را یک سانتی متر در نظر بگیرید.



۱) $۸ + ۲\sqrt{۲} + ۲\sqrt{۵}$

۲) $۶ + ۲\sqrt{۲} + ۲\sqrt{۵}$

۳) $۵ + ۲\sqrt{۲} + ۳\sqrt{۵}$

۴) $۷ + ۲\sqrt{۲} + ۲\sqrt{۵}$

پاسخ :

گزینه ۴،

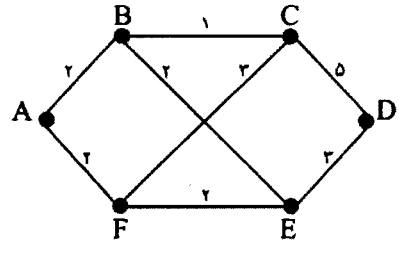
$$T_۷ \;\; \big|\sqrt{۵}$$

$$T_۴ \underline{\;۲\;} T_۳ \underline{\;\sqrt{۲}\;} T_۱ \underline{\;۲\;} T_۲ \underline{\;۲\;} T_۶ \underline{\;۱\;} T_۵$$

$$\big|\sqrt{۵}$$

$$T_۸ \;\; \underline{\sqrt{۲}} \;\; T_۹$$

۱۰۷- گراف بدون جهت و وزن دار روبه رو را در نظر می گیریم. اگر نقطه A را به عنوان مبدأ انتخاب کنیم و الگوریتمهای کروسکال و پریم را جداگانه بر روی این گراف اجرا کنیم لبه (یال)های انتخاب شده به ترتیب مطابق با کدام گزینه می باشد؟



۱) BC,BE,FE,AF,ED کروسکال AF,FE,BE,BC,ED پریم

۲) BC,BA,AF,FE,ED کروسکال AF,FE,EB,BC,ED پریم

۳) BC,FE,AF,AB,ED کروسکال AB,AF,FE,BC,ED پریم

۴) BC,BE,AB,ED,FE کروسکال AF,FE,ED,AB,BC پریم

پاسخ :

هر دو گزینه ۱ و ۲ درستند.

۱۰۸- سه آرایه A,B,C هر کدام با n عنصر را در نظر بگیرید. به قسمتی که A,C مرتب شده صعودی و B مرتب شده، نزولی است همچنین A,B عنصر تکراری ندارند ولی C دارد. به منظور مرتب سازی صعودی الگوریتمهای *Selection sort* و *Straight inseriom sort* و *quick sort* را بر روی هر کدام از این آرایه ها اجرا می کنیم. در این صورت می توان گفت:

۱) برای B پیچیدگی زمانی *heap sort, quick sort* یکسان است.

۲) برای C برنامه *quick sort* سریع تر از *heap sort* به پایان می رسد.

۳) برای C پیچیدگی زمانی *selection sort* و *Straight insertion sort* یکسان است.

۴) برای A اجرای *Straight isertion sort* سریع تر از *selection sort , quick sort* به پایان می‌رسد.

پاسخ :

۱۰۹- کدامیک از عبارات زیر صحیح است؟

۱) برای حل هر مسئله می توان یک الگوریتم با استفاده از روش برنامه ریزی پویا (*Dynamic Programming*) طراحی کرد.

۲) روش حریصانه (*Greedy*) همیشه یک راه حل بهینه را به دست می دهد.

۳) روش تقسیم و حل ی روش بالا به پایین می باشد در صورتی که روش برنامه ریزی پویا یک روش پایین به بالا می باشد.

۴) برای اینکه روش تقسیم و حل برای حل یک مسأله مورد استفاده قرار گیرد اصل *Optimality* بایستی برقرار باشد.

پاسخ :

گزینه ۳ . این یک قاعده کلی است.

۱۱۰- اگر دو ماتریس ۴×۴ با روش ضرب استرسن در یکدیگر ضرب شوند برای ضرب این دو ماتریس چند ضرب عددی صورت می گیرد؟

۱) ۲۸ ۲) ۴۹ ۳) ۷ ۴) ۱۱

پاسخ :

گزینه ۲، چون تبدیل می شود به هفت ضرب ماتریس های ۲×۲ و هر کدام از ضربهای ماتریسهای ۲×۲ تبدیل می شوند به هفت ضرب عددی.

۱۱۱- یک مسأله با استفاده از الگوریتم تقسیم و حل، حل شده است. در این مسأله هر مسأله بزرگ به پنج مسأله کوچکتر شکسته شده و اندازه هر مسأله شکسته شده $\frac{۱}{۳}$ مسأله بزرگتر می باشد. الگوریتم شکستن و ترکیب دارای زمان $O(n)$ می باشند. مرتبه بزرگی این الگوریتم چیست؟

۱) $\theta\left(n\log_{۳}^{۵}\right)$ ۲) $\theta\left(n^{۳}\right)$ ۳) $\theta\left(n\log_{۳}^{n}\right)$ ۴) $\theta\left(n^{۵/۳}\right)$

پاسخ :

گزینه ۱، منتها باید $\theta\left(n^{\log_{۳}^{۵}}\right)$ باشد.

چون به رابطه $T(n)=۵T\left(\frac{n}{۳}\right)+O(n)$ می رسیم که با توجه به قضیه صفحه ۶۵ کتاب جواب ۱ به دست می آید.

۱۱۲- کدامیک از گزینه های زیر غلط است؟

۱) $if\ F(n)=O(g(n))and\ F(n)=\theta(g(n))then\ F(n)=\Omega(g(n))$

۲) $if\ F(n)=\Omega(g(n))and\ F(n)=\theta(g(n))then\ F(n)=O(g(n))$

۳) $if\ F(n)=\Omega(g(n))and\ F(n)=O(g(n))then\ F(n)=\theta(g(n))$

۴) $if\ F(n)=O(g(n))and\ F(n)=\Omega(g(n))then\ F(n)=\theta(g(n))$

پاسخ :

گزینه ۴. چون اگر $F\in O(g)$ آنگاه $g\in\Omega(F)$.

۱۱۳- مینیمم و ماکیمم اعداد ذخیره شده در یک آرایه یک بعدی با n خانه، با چند مقایسه بین اعداد ذخیره شده در این خانه ها به دست خواهد آمد؟

۱) $\frac{۳n}{۲}$ ۲) $\frac{۳n}{۲}-۲$ ۳) $\frac{n}{۲}$ ۴) $\frac{n+۱}{۲}$

پاسخ :

گزینه ۲ در بدترین حالت $(n-۱)+(n-۲)$ مقایسه و در بهترین حالت $n-۱$ مقایسه می‌خواهد. پس در حالت میانگین

$$\frac{n-۲+n-۱+n-۱}{۲}=\frac{۳n}{۲}-۲$$

۱۱۴- می خواهیم حاصلضرب $ABCD$ (A یک ماتریس ۵×۱۳، B یک ماتریس ۸۹×۵، C یک ماتریس ۳×۸۹ و D یک ماتریس ۳۴×۳ می باشد) را پیدا کنیم به طوری که کمترین تعداد عمل ضرب انجام گیرد. ترتیب ضرب ماتریس ها عبارت است از:

۱) $(A(BC))D$ ۲) $A((BC)D)$ ۳) $(AB)(CD)$ ۴) $((AB)C)D$

پاسخ :

این تست تکرار تست ۴۰ است.

۱۱۵- در صورتی که آرایه مورد جستجو در جستجوی دودوئی به صورت ۷، ۶، ۵، ۴، ۳، ۲، ۱،۰و۱- باشد متوسط تعداد مقایسه ها برای جستجوی موفق چیست؟

۱) $\frac{۲۷}{۹}$ ۲) $\frac{۲۵}{۹}$ ۳) $\frac{۳۱}{۹}$ ۴) هیچکدام

پاسخ :

این تست تکرار تست ۴۱ است.

۱۱۶- فرض کنید $\lfloor a \rfloor$ قسمت صحیح یک عدد حقیقی a باشد یعنی بزرگترین عدد صحیحی که از a بزرگتر نباشد. دنباله T به صورت بازگشتی زیر تعریف می شود:

۱) $T(۱) = ۱$

۲) $T(n) = ۲.T\left(\left\lfloor n/۲ \right\rfloor\right), n \geq ۲$

و آنگاه مقدار $T(۷۳)$ برابر است با:

۱) ۳۲ ۲) ۱۲۸ ۳) ۶۴ ۴) ۶۸

پاسخ :

گزینه ۳.

$$T(۷۳) = ۲T(۳۶) = ۴T(۱۸) = ۸T(۹) = ۱۶T(۴) = ۳۲T(۲) = ۶۴T(۱) = ۶۴$$

۱۱۷- دنباله فیبوناچی به صورت بازگشتی زیر تعریف می شود:

۱) $FIB(۰) = FIB(۱) = ۱$

۲) $FIB(N) = FIB(n-۱) + FIB(n-۲)\ ,\ n \geq ۲$

مقدار $FIB(۸)$ برابر است با:

۱) ۳۴ ۲) ۲۱ ۳) ۵۵ ۴) ۱۳

پاسخ :

گزینه ۱، دنباله مطلوب از چپ به راست برابر است با:

۱،۲،۳،۵،۸،۱۳،۲۱،۳۴،....

۱۱۸- با توجه به تعاریف علامات Ω, σ, θ کدامیک از گزینه های زیر غلط است؟

۱) $\sigma\left(\sum_{i=1}^{m} F_i(n)\right) = \sigma\left(\underset{i=1}{\overset{m}{Max}}\{F_i\}\right)$

۲) $If\ \ g\ (n) \leq C\ \ For\ \ n \geq n_۰ then\ \ g\ (n)F(n) = \theta(F(n))$

۳) $\sigma(F(n) + g(n)) = \sigma(F(n)) + \sigma(g(n))$

۴) $If\ F\ (n) = \sigma(g(n))\ and\ \ g(n) = \Omega(F(n)) = \theta(g(n))$

پاسخ :

سؤال خیلی غلط چاپی دارد، ولی اگر منظور از σ همان O باشد گزینه ۳ غلط است.

۱۱۹- برای یافتن درخت پوشای حداقل یک گراف خلوت کدام از الگوریتم های زیر مناسب تر است؟

۱) *Floyd* ۲) *Prim* ۳) *Kruskal* ۴) *Dijkstra*

پاسخ :

گزینه ۳ .

۱۲۰- مرتبه بزرگی (*order of magnitude*) قطعه کد زیر چیست؟

```
k = ۰ ;
for i:=۱ to n do
begin
        for j :=۱ to m do
              k := k + ۱
         J := ۱ ;
         while J < n do
          begin
               k := k + ۱ ;
               J := ۲ ;
          End ;
end ;
```

۱) $\theta(n^2)$ ۲) $\theta(nm)$ ۳) $\theta(nm+n^2)$ ۴) $\theta(nm+n\log n)$

پاسخ :

گزینه ۳

۱۲۱- در الگوریتم *Mergesort* اگر به جای آنکه هر بار لیست به دو قسمت مساوی تقسیم شود به چهار قسمت مساوی تقسیم گردد و در مرحله ترکیب با چهار لیست در یک دیگر ادغام شوند پیچیدگی زمانی الگوریتم چه خواهد شد؟

۱) $\theta(n^{3/4})$ ۲) $\theta(n\log_2 n)$ ۳) $\theta(n^2)$ ۴) $\theta(n^2\log_4 n)$

پاسخ :

گزینه ۲، به رابطه $T(n)=4T\left(\frac{n}{4}\right)+O(n)$ می رسیم که طبق قضیه صفحه ۶۵ کتاب جواب به دست می آید.

۱۲۲- در الگوریتم زیر $F(3,6)$ چقدر است؟

```
int F(int m, int n)
{
     if (m = = ۱ or n = = ۰ or m = = n)
          retun (۱) ;
     else
          return (F(m-۱,n) + F(m-۱,n-۱))
}
```

۱) ۲۰ ۲) ۱۰ ۳) ۱۸ ۴) ۱۵

پاسخ :

این تست، تکرار تست ۶ است.

۱۲۳- در الگوریتم *Insetion sort* بهترین شرایط و بدترین شرایط به ترتیب از راست به چپ عبارت است از:

۱) مرتب شده نزولی، مرتب شده صعودی

۲) مرتب شده صعودی، مرتب شده نزولی

۳) مرتب شده صعودی، مرتب شده صعودی

۴) توالی عناصر ورودی اثری ندارد.

پاسخ :

مشخص نیست که می خواهیم آرایه صعودی مرتب شود، یا نزولی.

۱۲۴- اگر $A_{۱۳\times۵}, B_{۵\times۸۹}, C_{۸۹\times۳}, D_{۳\times۳۴}$ باشد حداقل تعداد عمل ضرب لازم برای محاسبه $ABCD$ چقدر است؟

۱) ۲۵۸۶ ۲) ۴۰۵۵ ۳) ۲۸۵۶ ۴) ۵۴۲۰

پاسخ :

این تست تکرار تستهای ۱۱۴ و ۴۰ است.

۱۲۵- الگوریتم زیر درخت پوشا با کمترین هزینه را به دست می آورد. منطق الگوریتم کدامیک از گزینه‌های ذیل است؟

```
T ← ۰ ;
while T conrains less than n − ۱ edge and E not empty do
begin
        choose an edge (V,W) From E of lowest cost ;
        delete (V,W) from E ;
        if (V,W) to T ;
            add (V,W) to T ;
        else
            discard (V,W);
end;
f T contains fewer than n − ۱ edge then
        Print ("no Spanning tree") ;
```

۱) پریم ۲) کروسکال ۳) راشال ۴) سولین

پاسخ :

گزینه ۲.

۱۲۶- گراف ساده غیر جهت دار و بدون حلقه $G(V,E)$ از ده مؤلفه همبند که هر کدام یک درخت است تشکیل شده است. اگر مجموعه درجه های رئوس گراف برابر صد (۱۰۰) در این صورت تعداد رئوس این گراف برابر است با:

۱) ۱۰ ۲) ۶۰ ۳) ۵۰ ۴) ۴۰

پاسخ :

گزینه ۴

$$۱۰۰ = \sum_{u \in v} \deg(u) = ۲q = ۲(p-۱۰) = ۲p - ۲۰ \Rightarrow ۲p = ۸۰ \Rightarrow p = ۴۰$$

۱۲۷- روی مجموعه X با n عنصر متمایز چند عمل دوتایی متمایز می توان تعریف کرد؟

۱) n^n ۲) $n \times n$ ۳) $n^{n^۳}$ ۴) $۲^{n^۲}$

پاسخ :

چون $X \times X$ دارای $n^۲$ عضو است، پس تعداد توابع از $X \times X$ به X برابر است با $n^{n^۲}$.

۱۲۸- یک آرایه از اعداد صحیح به صورت $A[۱..m]$ مفروض است، به طوری که $\sum_{i=۱}^{m} A[i] = S$ میباشد. در این صورت درجه اجرای الگوریتم زیر کدامیک از گزینه های زیر است؟

```
T := ۰;
For i := ۱ to m do
    For j ;= ۱ to A[i] do
        T := T + ۱ ;
```

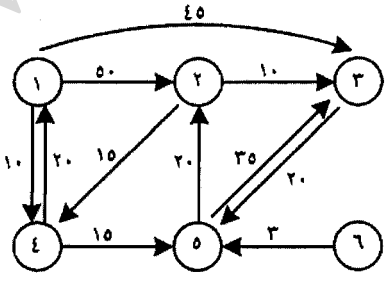
۱) $O(۵^۲)$ ۲) $O(m+۵)$ ۳) $O(m^۲)$ ۴) ۰

پاسخ :

غلط چاپی دارد.

۱۲۹- گراف زیر را در نظر بگیرید. کوتاهترین مسیر از گره ۱ به گره های ۲ و۳ به ترتیب چند است؟

۱) ۶۰،۴۵ ۲) ۶۰،۵۰ ۳) ۴۵،۴۵ ۴) ۳۵،۴۵



پاسخ :

گزینه ۳ مسیرها عبارتند از: ۲ → ۵ → ۴ → ۱ و ۳ → ۱ .

۱۳۰- بهترین زمان برا ترانهاده گرفتن از یک ماتریس خلوت (*Sparse*) چقدر است اگر n تعداد سطرها، m تعداد ستون ها و t تعداد عناصر ماتریس باشد.

۱) $O(nt)$ ۲) $O(mt)$ ۳) $O(mnt)$ ۴) $O(nm)$

پاسخ :

ماتریس های اسپارس و اعمال روی آنها مربوط به درس ساختمان داده ها می باشد.

۱۳۱- ۶ کار به شرح ذیل داریم: g_i نشان دهنده سود حاصل از اجرای کار iام است اگر و فقط اگر بعد از زمان d_i انجام نشود، فرض کنید هر کار در واحد زمان انجام می شود.

i	۱	۲	۳	۴	۵	۶
G_i	۲۰	۱۵	۱۰	۷	۵	۳
P_i	۳	۱	۱	۳	۱	۳

حداکثر سود حاصل از اجرا چقدر است؟

۱) ۴۲ ۲) ۴۵ ۳) ۳۷ ۴) ۳۲

پاسخ :

گزینه ۱. ۴۲ = ۷ + ۱۵ + ۲۰

۱۳۲- گراف همبندی بدون جهت با n گره و n-۲ لبه داریم. کدامیک از الگوریتمهای زیر برای تولید درخت پوشا با حداقل هزینه بر روی گراف مناسب تر است؟

۱) *Prim* ۲) *Kruskal* ۳) *Sollin* ۴) هیچکدام

پاسخ :

اگر منظور از لبه همان یال باشد گزینه ۲ می شود زیرا گراف خلوت است.

۱۳۳- بهترین زمان برای انتخاب k امین عنصر کوچک (k-۱ عنصر کوچکتر و بقیه بزرگترند) و عدد میانه از بین n عدد نامرتب به ترتیب از راست به چپ چقدر است؟

۱) $O(n \log n)$, $O(n \log n)$ ۲) $O(n)$, $O(n)$
۳) $O(n)$, $O(n \log n)$ ۴) $O(n \log n)$, $O(n)$

پاسخ :

گزینه ۱. ابتدا آرایه را مرتب می کنیم و بعد انتخاب را انجام می دهیم. البته الگوریتم های سریعتری هم وجود دارند که چون در کتاب عنوان نشده اند، به نظر نمی رسد منظور طراح آنها باشند).

۱۳۴- کدام یک از موارد زیر صحیح است؟

۱) $n! = O(n^n)$ ۲) $n^2 \log n = \theta(n^n)$
۳) $\frac{n^2}{\log n} = O(n^2)$ ۴) $n^{2n} + 6(2^n) = \theta(2^{2n})$

پاسخ :

گزینه های ۱ و ۳ صحیح هستند.

۱۳۵- رویه زیر را در نظر بگیرید:

```
Procedure f (a: String ; k ,n: integer );
{
    if (k = = n)
        print (a) ;
```

```
else
    for i := k to n do
    {
        Swap (a[k].a[i]) ;
        F (a, k+1, n) ;
}
```

تابع *print* رشته a را چاپ و تابع *Swap* جای دو عنصر را عوض می کند. کدام رابطه زیر درباره زمان F درست است؟

۱) $O(n!)$ ۲) $O(n^k)$ ۳) $O(n!)$ ۴) $O(n.n!)$

پاسخ :

گزینه های ۱ و ۳ صحیح هستند.

۱۳۶- اگر در مسأله فروشنده دوره گرد ماترس همجواری شهرها به صورت زیر باشد، حداقل مسیر کدام است؟

	V_1	V_2	V_3	V_4	V_5
V_1	۰	۱۴	۴	۱۰	۲۰
V_2	۱۴	۰	۷	۸	۷
V_3	۴	۵	۰	۷	۱۶
V_4	۱۱	۷	۹	۰	۲
V_5	۱۸	۷	۱۷	۴	۰

۱) ۳۰ ۲) ۳۱ ۳) ۴۳ ۴) ۳۴

پاسخ :

گزینه ۱ این همان مثال حل شده در صفحه ۲۹۰ کتاب است.

۱۳۷- کدامیک از جملات زیر صحیح نیست؟

۱) پیدا کردن بهترین روش ضرب (کمترین تعداد ضرب های لازم) برای ضرب n ماتریس $\theta(n^3)$ است.

۲) پیچیدگی الگوریتم *prim* ، $\theta(n^2)$ است.

۳) بدترین حالت الگوریتم *kruskal* $\theta(n^2 \log n)$ است.

۴) بهترین درجه برای الگوریتم کوله پشتی $O(2^n)$ و با روش حریصانه است.

پاسخ :

گزینه ۱، چون روش استراسن $O(n^{2/73})$ است.

۱۳۸- الگوریتم زیر به کدام روش عمل *Sort* را انجام می دهد؟

```
X Sort( )
{
   For i = ۲ to n do
   {
      X = A[i];
      j = i − ۱;
      while (j > ۰and A[i] > X) d ۰
      {
         A [j + ۱] = A[j];
         j = j − ۱;
       }
       A[j + ۱] = X ;
     }
} // end X Sort
```

۱) *radix* ۲) *shell* ۳) *insertion* ۴) *selection*

پاسخ :

گزینه ۲، مرتب سازی های *shell* , *radix* در سر فصل این درس نیستند.

۱۳۹- در بهترین حالت و بدترین حالت برای الگوریتم های مرتب سازی بهترین زمان ها کدام هستند؟

الف) بهترین حالت: *bobble sort-n* و بدترین حالت: *Merge sort-n log n*

ب) بهترین حالت: *insertion sort-n* و بدترین حالت: *Merge sort-n log n*

ج) بهترین حالت: *Merge sort –n log n* و بدترین حالت: *Merge sort-n log n*

د) بهترین حالت: *heap sort-n log n* و بدترین حالت: *heap sort- n log n*

۱) ج و د ۲) د ۳) الف ۴) الف و ب

پاسخ :

گزینه۱، در متن کتاب به این روابط اشاره شده است.