

merge-sort (A, p, r) $T(n)$

if p < r

q ← $\lfloor \frac{p+r}{2} \rfloor$ c

merge-sort (A, p, q) $\rightarrow T(n/2)$

merge-sort (A, q+1, r) $\rightarrow T(n/2)$

merge (A, p, q, r) $\rightarrow C(n)$

تکلیف زمانی:

combine $\rightarrow C(n)$ $\theta(n)$ $D(n)$
 $T(n) = C(n) + C + 2T(n/2) \rightarrow$ هزینه اجرای merge-sort
 مجموع هزینه‌ها n

تجزیه و تحلیل پس از زمان اجرا هم بازگشتی

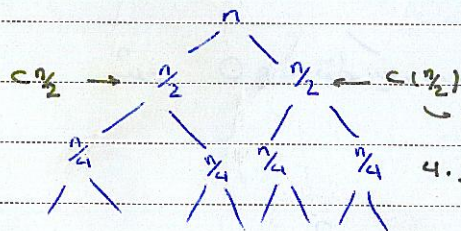
هزینه تقسیم

هزینه ترکیب

فرمول کلی

$$T(n) = aT(n/b) + D(n) + C(n)$$

هرگاه بازگشتی به صورت $T(n) = aT(n/b) + D(n) + C(n)$ باشد و $a, b, D(n), C(n)$ مشخص باشند می‌توانیم آن را حل کنیم.



$$C_n = C_{n/2} + C_{n/2}$$

$$4 \cdot \frac{n}{4} = C_n$$

برای تقسیم هزینه ثابت C داریم

$$C_1 n + C_2 \approx C_n$$

$$3C_1 n \leq C_n \leq 5C_1 n$$

هزینه ترکیب و تقسیم constant، بوسیله n هست.

تجزیه و تحلیل تقسیم و ترکیب، هر دو C_n می‌شوند.

تجزیه و تحلیل هر دو

$$2 \cdot \frac{C_n}{2}$$

زیر عملی هزینه تقسیم و ترکیب در هر سطح

دقتی که به دست می‌آید به ازای هر سطح به دست می‌آید.

لازمه‌های بازگشتی را می‌توانیم به این شکل بنویسیم: $T(n) = aT(n/b) + D(n) + C(n)$

جمع، که هزینه کل می‌شود

جمع هزینه سطح + هزینه برگها = هزینه اجرای کل

$$= C_n + \sum_{i=0}^{n-1} C_n$$

مجموع هزینه‌ها

جمع کل

P4PCO

$$n/2 = 1 \Rightarrow i = \log_2 n$$

$$2^i = 2^{\log_2 n} = n$$

$$C_n + C_n(\log n - 1)$$

جمله به ازای n می‌شود بیشتر

Subject:

Year. Month. Date. ()

2-3-7

merge-sort $T(n) = \theta(n \log n)$

تجزیه (2,3,4) الگوریتم زمان اجرای $n \log n$ به سبب برای اینست:

5 |
مدرصیح
مدرصیح x

درک اینست که مجموعش x شود.

$$\exists a, b \in S \Rightarrow a + b = x$$

رشد تابع:

صداها می توانند خود تابع هم باشد برای صداها می توانیم ()

صداها می توانیم ()

همه چیزها می توانند ()

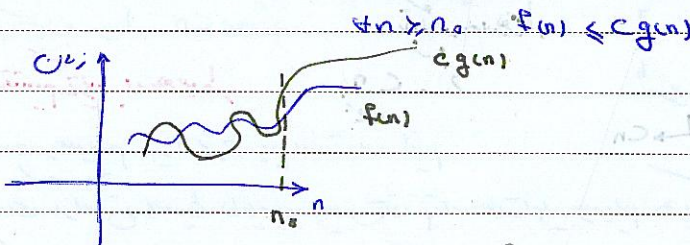
همه چیزها می توانند ()

بزرگ O و ω اینها صداها می توانند خود تابع باشند ()

()

تراجاع غیر منفی هستند ()

$$f(n) = O(g(n)) \iff \exists c, n_0 \text{ s.t. } \forall n > n_0, f(n) \leq c g(n)$$



(مثال) $f(n) = 2n^2 + 10$

$g(n) = n^2$

$f(n) = O(n^2)$

$g(n) = n^3$

$\forall n > n_0, 2n^2 + 10 \leq cn^2$

$C = 4, n_0 = 3$

Subject: 2

Year. Month. Date. ()

$$f(n) \neq O(n)$$

$\cdot C_1 n^2, n^2, n \log n, n^2, n \log n, n^2, n \log n$ ←

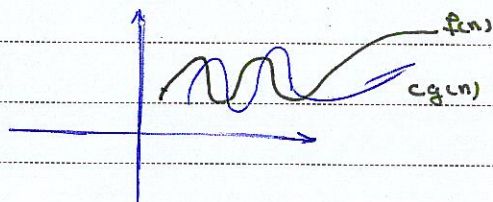
$$f(n) \neq O(n \log n)$$

$$f(n) = \Omega(g(n)) \iff \exists c > 0$$

$$n_0 > 0$$

$$\forall n > n_0$$

$$f(n) \geq c g(n)$$



d2) $f(n) = 2n^2 + 10$

$$g(n) = n^2 \Rightarrow f(n) = \Omega(n^2)$$

$$g(n) = n \log n \Rightarrow f(n) = \Omega(n \log n)$$

$$g(n) = n \Rightarrow f(n) = \Omega(n)$$

invis

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n))$$

(g, f) in the same order of growth

$$f(n) = \Theta(g(n)) \iff \exists c_1, c_2 > 0, n_0 > 0$$

$$\forall n > n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

with constant c_1, c_2

d2) $f(n) = 2n^2 + 10$

$$g(n) = 3n^2 + 100 \Rightarrow 2n^2 + 10 = \Theta(3n^2 + 100)$$

$$g(n) = 100n^2 - 50 \Rightarrow 2n^2 + 10 = \Theta(100n^2 - 50)$$

$$2n^2 + 10 \neq \Theta(n^3)$$

$$\neq \Theta(n)$$

Subject:

Year. Month. Date. ()

$$f(n) = O(g(n)) \iff \forall c > 0, \exists n_0 > 0$$

$$\forall n \geq n_0, f(n) \leq c g(n)$$

ف(ن) = O(گ(ن)) یعنی برای هر c > 0 و n > n_0، f(n) ≤ c g(n) است. constant

Big O notation

$$f(n) = 2n^2 + 10$$

$$f(n) = O(n^2 \log n)$$

$$f(n) = O(n^3)$$

$$f(n) \neq O(n^2)$$

$$f(n) = \omega(g(n)) \iff \forall c > 0, \exists n_0 > 0$$

$$\forall n \geq n_0, c g(n) < f(n)$$

$$f(n) = 2n^2 + 10$$

$$f(n) = \omega(n \log n)$$

$$f(n) \neq \omega(n^2)$$

$$= \omega(n)$$

$$\rightarrow f(n) = O(g(n))$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$n \rightarrow \infty$$

$$f(n) = \omega(g(n))$$

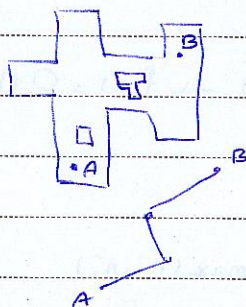
$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

$$n \rightarrow \infty$$

computational Geometry

هندسه حسابی

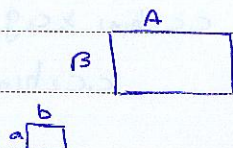
مسئله های هندسی و الگوریتم ها



برای حرکت یک robot مسیر پیدا کردن در محیط مختصات
 مختصات برای روبات مختصات است و تغییر مسیر را مشخص می کند

maven.smith.edu/~nourourke/ToPP

pallet loading



مسئله چگونگی بارگیری در یک روبات برای بارهای مختلف (انتقال)
 عبور از این (مسئله) برای بار (قرار دهیم)

انتقال اجسام هندسی در محیط مختصات

نقاط و خطوط در محیط مختصات

upper bound - lower bound

Subject:

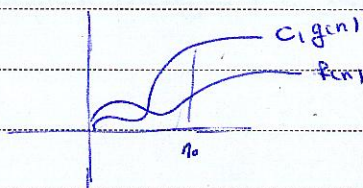
Year. Month. Date. ()

خاصیت تراندزی:

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n))$$

$$\Rightarrow f(n) = \Theta(h(n))$$

$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$



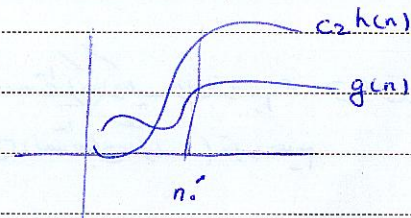
$$n_0 = \max(n_0, n'_0)$$

$$c' = c_1 c_2$$

$$c' h(n) \geq f(n)$$

$$c_1 c_2 h(n) \geq c_1 g(n) \geq h(n)$$

$$c_1 c_2 h(n) \geq c_1 g(n)$$



برای هر 5 notation خاصیت تراندزی برقرار است.

خاصیت یکنوازی:

$$f(n) = \Theta(f(n))$$

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

constant کارایه در تقریب برقرار است

برای 0، تا کوچک برقرار نیست چون هیچ وجه پیدا کردنش از f نمیشد یا بیشتر باشد

خاصیت تقارنی:

$$f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$$

اگر f و g از هم بزرگتر باشند f از g بزرگتر است و برعکس

این فقط برای 0 است و برای مثال برای 0 برقرار نیست

خاصیت پارتیالی:

$$f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \iff g(n) = \omega(f(n))$$

① $f(n) = O(g(n))$ یعنی $f(n)$ از $g(n)$ کوچکتر است.
 ② $f(n) = \Omega(g(n))$ یعنی $f(n)$ از $g(n)$ بزرگتر است.
 ③ $f(n) = \Theta(g(n))$ یعنی $f(n)$ و $g(n)$ هم‌اندازه هستند.

$$O, \Omega, \Theta, o, \omega, \asymp, \sim$$

$$a, b \in \mathbb{R} \quad ① a > b$$

$$② a < b$$

$$③ a = b$$

در سیمپل برای notation $f(n)$ و $g(n)$ می‌توانیم بنویسیم $f(n) = O(g(n))$ یا $f(n) = \Omega(g(n))$ یا $f(n) = \Theta(g(n))$.

$$f(n) \neq O(g(n))$$

$$f(n) \neq \Omega(g(n))$$

$$f(n) \neq \Theta(g(n))$$

$$\begin{cases} f(n) = n \\ g(n) = n \sin n \end{cases}$$

برای $n \rightarrow \infty$ $f(n)$ و $g(n)$ هم‌اندازه هستند.

← برای notation $f(n)$ و $g(n)$ می‌توانیم بنویسیم $f(n) = O(g(n))$ یا $f(n) = \Omega(g(n))$ یا $f(n) = \Theta(g(n))$.

3.1-1 Θ

برای مقایسه از Θ می‌توانیم بنویسیم

$$\max(f(n), g(n)) = \Theta(f(n) + g(n))$$

$$h(n) = \begin{cases} f(n) & f(n) \geq g(n) \\ g(n) & g(n) > f(n) \end{cases}$$

3.1-2 Θ

$$(n+a)^b = \Theta(n^b)$$

$$\exists c_1, c_2 > 0, n_0 \in \mathbb{N} \text{ such that } n \geq n_0 \implies c_1 n^b \leq (n+a)^b \leq c_2 n^b$$

$$f(n) + g(n) \geq \max(f(n), g(n)) \geq \frac{1}{2}(f(n) + g(n))$$

$$f(n) + g(n) \geq g(n) \geq 0$$

$$f(n) + g(n) \geq h(n) = \max(f(n), g(n))$$

$$c_1 f(n) + c_2 g(n) \geq h(n) =$$

$$f(n) \leq h(n)$$

$$g(n) \leq h(n)$$

$$f(n) + g(n) \leq 2h(n) \implies h(n) \geq \frac{1}{2}(f(n) + g(n)) \quad c_1 = \frac{1}{2}, c_2 =$$

4. Θ

$$\frac{1}{2} n^b \leq (n+a)^b \leq (2n)^b = 2^b n^b$$

$$\frac{1}{2} n^b \leq (n+a)^b \leq (2n)^b = 2^b n^b$$

Subject:

Year. Month. Date. ()

یہ سب سے زیادہ ریاضی ہے

$$x \in \mathbb{R}$$

$$x-1 < [x] \leq x \leq [x] < x+1$$

$$n \in \mathbb{N}$$

$$\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil = n$$

$$n \in \mathbb{R} \quad a, b > 0, \quad a, b \in \mathbb{N}$$

$$\lceil \frac{\lceil \frac{n}{a} \rceil}{b} \rceil = \lceil \frac{n}{ab} \rceil$$

$$\lfloor \frac{\lfloor \frac{n}{a} \rfloor}{b} \rfloor = \lfloor \frac{n}{ab} \rfloor$$

$$\lceil \frac{n}{b} \rceil \leq (a + (b-1))/b$$

$$\lfloor \frac{n}{b} \rfloor \geq (a - (b-1))/b$$

$$\lim_{n \rightarrow \infty} \frac{n^b}{a^n} = 0 \Rightarrow n^b = o(a^n) \quad \text{اگر } a, b \in \mathbb{R}$$

$$a \in \mathbb{R}, \quad a > 1$$

سب سے زیادہ ریاضی ہے

$$\lim_{n \rightarrow \infty} \frac{\log^n n}{n^a} = 0 \Rightarrow (\log^n n) = o(n^a)$$

$$n \rightarrow \infty$$

$$\log n \quad n \quad 0, \infty, \dots, \infty$$

سب سے زیادہ ریاضی ہے

$$f^{(i)}(n) = \begin{cases} n & \text{if } i=0 \\ f(f^{(i-1)}(n)) & \text{if } i>0 \end{cases}$$

سب سے زیادہ ریاضی ہے

$$\log^* n = \min \{ i > 0 : \log^{(i)} n \leq 1 \}$$

سب سے زیادہ ریاضی ہے

$$\log^* 2 = 1$$

$$\log^* 4 = 2$$

$$\log^* 2^{(65536)} = 5$$

$$\log^* 2^{80} \approx 10 < 2$$

سب سے زیادہ ریاضی ہے

اعداد فیبوناچی:

$$F_0 = 0$$

$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2} \quad \text{if } i \geq 2$$

↪ recursive

نسبت طلایی $\phi = \frac{1+\sqrt{5}}{2} \approx 1.6$
 $\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.6$

↪ این جمله تقریباً برابر صفر می شود

recursive غیر

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

فیبوناچی در صورتی که i بزرگ باشد

$$n! = O(n^n)$$

$$\lim_{n \rightarrow \infty} \frac{n!}{n^n} = 0$$

تعیین 3.2.3

$$n! = \omega(2^n)$$

$$\lim_{n \rightarrow \infty} \frac{n!}{2^n} = \infty$$

تعیین 3.2

A	B	O	o	Ω	ω	Θ
$\lg^n n$	n^k			X	X	
n^k	c^n			X	X	
$\sqrt{n}$	$\sin n$					
2^n	$2^{n/2}$	X		X		X
$\lg c$	$c \lg^n$			X	X	
$\lg(n!) \lg n$		X	X			

$\epsilon > 0, k > 1, c > 1$

تعیین 1: برای n از n_0 به بالا $n!$ تقریباً $\Theta(n!)$ می شود.

if $n=1$

return S

else

$S \neq \text{perm}(S, n-1)$

for (int i=0; i < n; i++)

{ for (int j=i; j < n; j++)

Subject:

Year. Month. Date. ()

رابطه بازگشتی:

۱. روش جایگزینی: داریم این معادله را استفاده از استقرا اثبات می‌کنیم.
 ۲. روش درخت بازگشت: این روش را برای حل مسائل بازگشتی (تجزیه و تحلیل) می‌توانیم استفاده کنیم.
 ۳. استفاده از قضیه اصلی (قضیه کارسکی-اسکول): این روش را می‌توانیم استفاده کنیم.
- رابطه بازگشتی: $T(n) = 2T(\frac{n}{2}) + n$

روش جایگزینی:

فرض کنیم در روش جایگزینی را از $T(1) = 1$ شروع می‌کنیم و می‌خواهیم رابطه بازگشتی را اثبات کنیم.

$$T(n) = 2T(\frac{n}{2}) + n \quad \text{merge sort} \quad T(1) = 1$$

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

$$T(n) \leq C n \log n$$

$$T(\frac{n}{2}) \leq C \frac{n}{2} \log \frac{n}{2}$$

$$T(n) = 2T(\frac{n}{2}) + n \leq 2C \frac{n}{2} \log \frac{n}{2} + n$$

$$\leq C n \log \frac{n}{2} + n$$

$$\leq C n (\log n - \log 2) + n$$

$$\leq C n \log n - C n + n \leq C n \log n$$

$$C n \log n \leq C n \log n \quad \text{I}$$

$$n=2 \Rightarrow T(2) = 2T(\frac{2}{2}) + 2 = 2T(1) + 2 = 2 \times 1 + 2 = 4$$

$$T(2) = 4 \leq C \times 2 \times \log 2 = 2C \Rightarrow C \geq 2 \quad \text{II}$$

$$\text{II} \Rightarrow \text{I} \Rightarrow C \geq 2$$

همین‌طور می‌توانیم اثبات کنیم که $C \geq 2$ است.

$$T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1$$

$$T(1) = 1$$

مثال

$$T(n) = O(n)$$

$$T(n) \leq \underbrace{Cn - b}_{O(n)}$$

تست حتمی بوسیله n

برای بررسی درست است یا نه برای مثال حجم مرتب است

$$T(\lfloor \frac{n}{2} \rfloor) \leq C \lfloor \frac{n}{2} \rfloor - b$$

$$T(\lceil \frac{n}{2} \rceil) \leq C \lceil \frac{n}{2} \rceil - b$$

$$T(n) \leq C \lfloor \frac{n}{2} \rfloor + C \lceil \frac{n}{2} \rceil + 1 - 2b$$

C کم نیست و حتی به

$$\leq Cn + 1 - b - b + 1 \Rightarrow \leq Cn - b$$

این رابطه برقرار است

مطم : $\leq Cn$

این برآورد نیست پس به خطای نسبی چون C و n هر دو مثبت هستند

$$b+1 \leq \lfloor \frac{n}{2} \rfloor$$

const جمله اضافی داریم

const جمله اضافی نداریم

اگر جمله اصلی درجه اول باشد و جمله اضافی کمتر از 1/2 باشد

اگر درجه اول از درجه جمله کمتر بود جمله درست است و فقط باید اصلاح شود

اگر درجه اول جمله برابر بود باید یک ثابت در 1/2 به جمله اضافی کنیم

اگر بیشتر باشد جمله باید عوض شود و درجه را زیادتر کنیم

$$T(2) = T(\lfloor \frac{2}{2} \rfloor) + T(\lceil \frac{2}{2} \rceil) + 1 = 3$$

$$T(2) \leq 2C - b$$

$$3 \leq 2C - b \Rightarrow \frac{3+b}{2} \leq C \Rightarrow b=1 \Rightarrow C \geq 2$$

باید راهی بتوانیم که درجه توان 1 درجه کمتر نیست

اگر درجه توان 1 بیشتر از درجه جمله باشد جمله اضافی باید کمتر از 1/2 باشد

در مثال قبلی (مثال اول) اگر order را به نسبت به O(n) حساب کنیم داریم:

$$T(n) = O(n)$$

$$T(n) \leq Cn$$

$$T(\lfloor \frac{n}{2} \rfloor) \leq C \lfloor \frac{n}{2} \rfloor$$

$$T(n) \leq 2C \lfloor \frac{n}{2} \rfloor + n$$

$$\leq 2C \lfloor \frac{n}{2} \rfloor + n$$

$$\leq Cn + n$$

$$\leq (C+1)n$$

$$\leq Cn \Rightarrow X$$

درجه اول جمله اصلی است پس باید کمتر از 1/2 باشد

Subject:

Year. Month. Date. ()

تقسیم توان

$$T(n) = 2T(\sqrt{n}) + \lg n$$

مثال

$$n = 2^m \quad (\text{تغییر متغیر}) \quad \text{و } m = \lg n$$

$$T(2^m) = 2T(2^{m/2}) + m$$

$$S(2^m) = S(m) \quad (\text{تغییر متغیر})$$

$$S(m) = 2S(m/2) + m$$

$$S(m) = O(m \lg m)$$

$$T(2^m) = O(m \lg m)$$

$$T(n) = O(\lg n \lg \lg n)$$

باید تغییر متغیر را برگردانیم

$$T(n) = 2T(\sqrt{n}) + 1$$

تمرین 4.1-6

روش درخت بازگشت:

این روش حساب کنیم جواب دقیق بدست می آید (در صورتی که بتوانیم) و در صورتی که نتوانیم بازگشت کنیم

$$T(n) = 3T(n/4) + cn^2 \quad (\text{تقسیم توان})$$

مثال

$$\text{خوبه: } T(n) = \sum_{i=0}^{\log_4 n} \text{ هزینه در هر سطح}$$

$$i: \text{ تعداد درخت ها}$$

$$i: \text{ هزینه در هر سطح: } c \times \left(\frac{n}{4^i}\right)^2 = \frac{cn^2}{16^i}$$

$$i: \text{ هزینه در هر سطح: } 3i + c \left(\frac{n^2}{16^i}\right) = cn^2 \left(\frac{3}{16}\right)^i$$

$$\frac{n}{4^i} = 1 \quad \text{و } i = \log_4 n$$

تعداد درخت ها (ارتفاع درخت)

$$i: \text{ تعداد درخت ها} = 3^i = 3^{\log_4 n} = n^{\log_4 3}$$

$$\text{خوبه: } T(n) = \sum_{i=0}^{\log_4 n - 1} cn^2 \left(\frac{3}{16}\right)^i + cn^{\log_4 3} = cn^2 \left(\frac{1 - \left(\frac{3}{16}\right)^{\log_4 n}}{1 - \frac{3}{16}}\right) + cn^{\log_4 3}$$

$$\leq \sum_{i=0}^{\infty} cn^2 \left(\frac{3}{16}\right)^i + cn^{\log_4 3} \leq \frac{cn^2}{1 - \frac{3}{16}} + cn^{\log_4 3} < \frac{16}{13} cn^2 + cn^{\log_4 3}$$

$$= O(n^2)$$

Subject:

Year. Month. Date. ()

$$T(n) = O(n \lg^n n)$$

با استفاده از مثال ۱

$$T(n) \leq cn \lg n$$

$$\begin{cases} T(\frac{n}{3}) \leq \frac{cn}{3} \lg \frac{n}{3} \\ T(\frac{2n}{3}) \leq \frac{c \cdot 2n}{3} \lg \frac{2n}{3} \end{cases}$$

$$T(n) \leq \frac{cn}{3} \lg \frac{n}{3} + \frac{2cn}{3} \lg \frac{2n}{3} + dn$$

$$T(n) \leq \frac{cn}{3} \lg n + \frac{2cn}{3} (\lg^2 + \lg n) + dn - \frac{cn}{3} \lg^3 - \frac{2cn}{3} \lg$$

$$T(n) \leq cn \lg n + dn + \frac{2cn}{3} - cn \lg^3$$

۴۰

$$T(n) \leq cn \lg n$$

$$d + \frac{2}{3}c + \lg^3 \times c \leq 0 \Rightarrow c \geq \frac{d}{\lg^3 - \frac{2}{3}}$$

Master Theorem

روش استفاده از قضیه اصلی

$$T(n) = a T(\frac{n}{b}) + f(n)$$

$$b > 1, a \geq 1$$

روش استفاده از قضیه اصلی
روش استفاده از قضیه اصلی
روش استفاده از قضیه اصلی

$$n^{\log_b a} \quad ? \quad f(n)$$

$$f(n) = O(n^{\log_b a - \epsilon})$$

$$T(n) = \Theta(n^{\log_b a})$$

$$f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \times \lg n)$$

$$f(n) = \Omega(n^{\log_b a + \epsilon}), \epsilon > 0, a f(\frac{n}{b}) \leq c f(n), c < 1 \Rightarrow T(n) = \Theta(f(n))$$

Subject: 40

Year.

Month.

Date.

()

$$T(n) = 9T(n/3) + n$$

(مثال 1)

$$a=9, b=3, n^{\log_a b} = n^{\log_3 9} = n^2$$

$$f(n) = n$$

 $f(n)$ به n^2 مقایسه می‌شود

$$f(n) = n = O(n^{2-\epsilon})$$

 ϵ را به قدری انتخاب می‌کنیم

$$2-\epsilon > 1 \Rightarrow \boxed{\epsilon < 1}$$

$$\text{طبق قضیه اصلی حالت اول} \Rightarrow T(n) = \Theta(n^{\log_3 9}) = \Theta(n^2)$$

$$T(n) = T(n/2) + 1$$

(مثال 2)

$$f(n) = 1 \quad ? \quad n^{\log_2 1/2} = n^0 = 1$$

 $f(n)$ به n^0 مقایسه می‌شود

$$f(n) = 1 = \Theta(n^{\log_2 1/2}) = \Theta(n^0) = \Theta(1)$$

$$\text{طبق حالت دوم قضیه اصلی} \quad T(n) = \Theta(n^{\log_2 1/2} \lg n) = \Theta(1 \times \lg n) = \Theta(\lg n)$$

$$T(n) = 3T(n/4) + n \lg n$$

(مثال 3)

$$f(n) = n \lg n \quad ? \quad n^{\log_4 3}$$

 $\lg$ توان 1 و $\lg$ توان 1/2 $\Rightarrow$ توان کمتر از 1 است

$$f(n) = n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

 $\epsilon = 1 - \log_4 3$
 $n^{\log_4 3 + \epsilon}$ از $\log_4 3$ بیشتر است پس بزرگتر است

 ϵ را به قدری انتخاب می‌کنیم

$$\log_4 3 + \epsilon < 1 \Rightarrow \epsilon < 1 - \log_4 3$$

$$a f(n/b) \leq c f(n)$$

 $f(n)$ به $n \lg n$ مقایسه می‌شود

$$3 \frac{n}{4} \lg \frac{n}{4} < c n \lg n$$

$$\frac{3n}{4} \lg n - \frac{3n}{4} \lg 4 < c n \lg n$$

$$c > \frac{3}{4}$$

طبیعت حالت سرعته $T(n) = \theta(f(n)) = \theta(n \lg n)$

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

$$f(n) = n \lg n$$

$$n \log_2 a = n \log_2 2 = n$$

I, II, III, IV, V, VI, VII, VIII, IX, X

$\lg^n$ یعنی $\lg$ چند بار است و حالت سیم

$$f(n) = n \lg n \stackrel{?}{=} n \cdot (n^{\log a} + \epsilon) = n \cdot (n^{1+\epsilon})$$

۴۸ آیا توانستید صبر کنید؟ و یا سبک صبر شدی از "و یا سبک شدی" و در نتیجه هیچ عیبی نداشتی و در نتیجه سبک شدی

$$\lg n = \Omega(n^{1+\epsilon})$$

مناہر میں غمی ترانیم از غنیمہ اصلیہ و شادی و شہادت و غیرہ جہاں فرض و سیرت و سنیت

تمرین 2-4.4

المرتبين Heap Sort

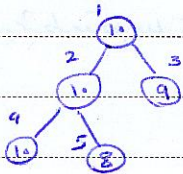
از ساختن داده ای به نام Heap برای sort کردن استفاده می کنند

← Heap → در صورت تغییر نسبت، در هر گامی درخت باید مجدداً تصحیح شود.

Clipping 15 cm long and 1 cm high

" " " " " " main Heap

max



ای ریش Heap در حافظه پویا (dynamic memory) قرار دارد. این حافظه توسط برنامه‌نویس مدیریت می‌شود. در مقابل، متغیرهای global و static در حافظه ای قرار دارند که توسط سیستم مدیریت می‌شود.

از بالا به پایین از چپ به راست جداولی تر برای درک

10	10	9	10	8
----	----	---	----	---

هر گاه i به عنوان $2i$

فرزند است $2i+1$

پدر $\lfloor \frac{i}{2} \rfloor$

$$\text{LEFT}(i) = 2i$$

$$\text{RIGHT}(i) = 2i+1$$

$$\text{PARENT}(i) = \lfloor \frac{i}{2} \rfloor$$

طول آرایه $\text{Length}(A)$ نشان می‌دهد

تعداد عناصر آرایه

$\text{Heap-Size}[A]$ تعداد عناصری که استفاده می‌شود و بقیه اجزای

valid در heap

$$A[\text{PARENT}(i)] \geq A[i] \quad \leftarrow \text{max heap}$$

این ویژگی برای هر یک از گره‌ها برقرار است و برعکس صورت ندارد.

استدلال max-Heapify

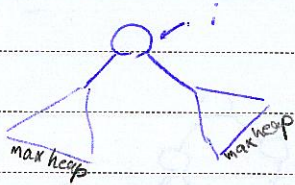
باید فرض کنیم که فرزندان هر گره heap هستند.

فرض کنیم max heap شود.

برای اینکه در هر مرحله این خاصیت برقرار باشد

الگوریتم از فرزندان هر گره به ترتیب بررسی می‌کند تا اگر یکی از آن‌ها بزرگتر از پدر باشد، آن‌ها را با پدر عوض می‌کند.

این کار را تا زمانی که به گرهی برسیم که بزرگتر از پدرش نباشد، انجام می‌دهیم.



Subject:

Year. Month. Date. ()

max-Heapify(A, i)

$l \leftarrow \text{LEFT}(i)$

$r \leftarrow \text{RIGHT}(i)$

if ($l < \text{heap-size}[A]$ and $A[l] > A[i]$)

then largest $\leftarrow l$

else largest $\leftarrow i$

max بین i و فرزانش را پیدا می کنیم

محل است که بزرگترین فرزند در آن است و چون بزرگترین است

if ($r < \text{heap-size}[A]$ and $A[r] > A[\text{largest}]$)

then largest $\leftarrow r$

if largest $\neq i$ then

heap $\leftarrow \text{largest} = i$

exchange($A[i], A[\text{largest}]$)

max-Heapify($A, \text{largest}$)

خرید اجاره

$T(n) \leq \Theta(n) + \Theta(n)$

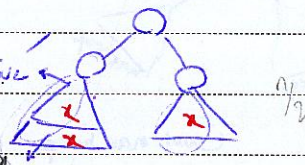
$T(n) = O(\lg n)$

max-Heapify

constant

$T(n) = T(\frac{2n}{3}) + C$

ما بزرگترین فرزند را پیدا می کنیم و با والدین مقایسه می کنیم و اگر بزرگتر باشد جایگزین می کنیم و اگر نه خود والدین را نگه می داریم و به فرزند بزرگتر می رویم و این کار را تکرار می کنیم تا به برگ برسیم



$$3x = n - 1$$

$$x = \frac{n-1}{3}$$



$$n \approx \frac{2n}{3}$$

$$f(n) = c \quad ? \quad \log_b a = n \quad \log_{3/2} n = n^0 = 1$$

$$c = \Theta(1) \Rightarrow T(n) = \Theta(1 \times \lg n)$$

$$= \Theta(\lg n)$$

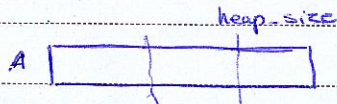
الگوریتم ساختن Heap :

در اینجا max-Heapify را می بینیم

این الگوریتم برای ساختن یک max-Heap است. در هر مرحله max-Heapify را روی یک زیر درخت اجرا می کنیم.

در هر مرحله فقط به یک سمت می رویم.

$$2i, 2i+1 > \text{heap-size}[A]$$



Build-max-Heap(A)

heap-size[A] ← length[A]

for i ← [length[A]/2] down to 1

max-Heapify(A, i)

در اینجا max-Heapify را می بینیم

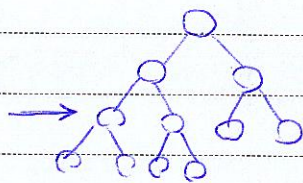
این الگوریتم برای ساختن یک max-Heap است.

$$T(n) = O(n \lg n)$$

ن/ن

این tight است یا نه؟

ارتفاع i × c ≤ هزینه max-Heapify(A, i)



در اینجا از یک max-Heap به یک max-Heap دیگر تبدیل می کنیم.

ناچهارم از زیر

تعداد نود

ارتفاع نود

1

lg n

2

lg n - 1

4

lg n - 2

8

2

16

1

32

$$\text{max-Heapify هزینه } T(n) = \sum_{i=1}^{\lg n} \left(\frac{n}{2^i} \times i \times c \right)$$

Subject:

Year. Month. Date. ()

$$T(n) = \frac{cn}{2} \sum_{i=1}^{\lg n} i \left(\frac{1}{2}\right)^i < \frac{cn}{2} \sum_{i=0}^{\infty} i \left(\frac{1}{2}\right)^i$$

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

$|x| < 1$

$$|x| = \frac{1}{2}$$

$$T(n) < \frac{cn}{2} \times \left(\frac{\frac{1}{2}}{(1-\frac{1}{2})^2}\right) < \frac{cn}{2} \times 2$$

$$T(n) < cn \Rightarrow T(n) = O(n)$$

For

Subject:

Year: Month: Date: ()

Counting Sort



نوع داده

$0 \leq k \leq n$



تعداد اعداد

$O(n+k)$

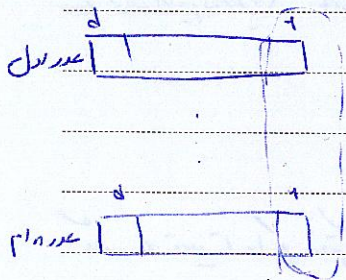
مبنای اعداد

بهره‌مندی: $O(n+k)$ در زمان، $O(1)$ در فضای اضافی

Radix Sort

نوع داده: عددی در مبنای k

اگر داده‌ها به صورت k رقمی باشند، می‌توانیم از این روش استفاده کنیم.



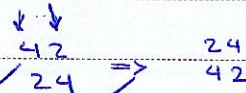
در هر مرحله، مقادیر را بر اساس رقم اول مرتب می‌کنیم.

در هر مرحله، مقادیر را بر اساس رقم دوم مرتب می‌کنیم. این فرآیند را تا زمانی که تمام ارقام مرتب شوند ادامه می‌دهیم.

3 2 9	7 2 0	7 2 0	3 2 9
4 5 7	3 5 5	3 2 9	3 5 5
6 5 7	4 3 6	4 3 6	4 3 6
8 3 9	4 5 7	8 3 9	4 5 7
4 3 6	6 5 7	3 5 5	6 5 7
7 2 0	3 2 9	4 5 7	7 2 0
3 5 5	8 3 9	6 5 7	8 3 9

لیست مرتب شده

اگر در هر مرحله، مقادیر را بر اساس رقم اول مرتب می‌کنیم، در هر مرحله، مقادیر را بر اساس رقم دوم مرتب می‌کنیم. این فرآیند را تا زمانی که تمام ارقام مرتب شوند ادامه می‌دهیم.



Subject:

Year. Month. Date. ()

radix sort را تغییر دهید تا به جای از ارقام با ارزش کمتر به ارقام با ارزش بیشتر

تغییر کنید! اگر مرتبه کم ارزش را از ابتدا شروع کنید.

Radix Sort (A, d)

for $i \leftarrow 1$ to d do

use a stable sort to sort

array A on digit A

counting sort $\theta(n+k)$

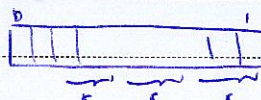
چون d را این که (sort) ای ای ای

$$\theta(d(n+k))$$

اگر d را 2^r بگیریم $\theta(n)$ می شود $\theta(n)$ به جای $\theta(n+k)$

در این sort، ارقام را به هم ترکیب کنیم و به جای آنرا عوض کرد

b bit در 2



در این sort، ارقام را به هم ترکیب کنیم و به جای آنرا عوض کرد

b bit = b bit

$k = 2$ در 2

$$d = \lceil \frac{b}{r} \rceil$$

$k = 2^r$

$$\theta(d(n+k))$$

$$\theta(d(n+k))$$

$$\theta(b(n+2))$$

$$\theta(b_r(n+2))$$

در این sort، ارقام را به هم ترکیب کنیم و به جای آنرا عوض کرد
اگر d را 2^r بگیریم $\theta(n)$ می شود $\theta(n)$ به جای $\theta(n+k)$

8.3-4

تغییر

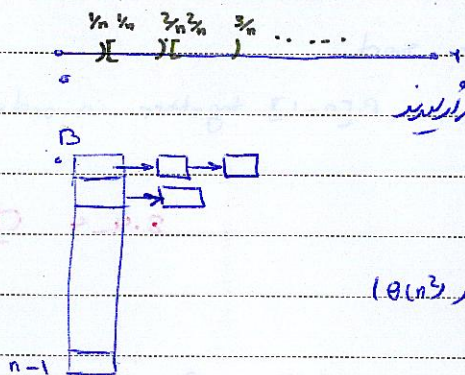
Subject:

Year. Month. Date. ()

دقیق مشخص است. می توان از این استفاده کرد

Bucket Sort

برای اعداد صحیح که در $range [a, b]$ هستند می توانیم Bucket Sort را استفاده کنیم چون توزیع یکنواخت است می توانیم از این استفاده کنیم. $\theta(n)$ است. اما اگر اعداد به صورت تصادفی باشند $\theta(n^2)$ می شود.



برای این یکنواختی انتظار داریم که در بازه ها تعدادی اعداد قرار بگیرد

تعداد زیر بازه ها را داریم B خانه داریم

در هر بازه ای که داریم دو کلمه به آن قرار می دهیم و در آن جا بررسی می کنیم

خودمان از این لیست ها در مسئله insertion sort می توانیم $\theta(n^2)$

چون تعداد اعداد به $\theta(n)$ می شود $\theta(n)$

و چون n بازه داریم که باید بررسی کنیم $\theta(n) \times \theta(n) = \theta(n^2)$

مثال

A	B
0.78	0.12 → 0.17 →
0.17	0.21 → 0.23 → 0.24 →
0.39	0.31 →
0.26	
0.72	
0.94	
0.21	0.68 →
0.12	0.72 → 0.78 →
0.23	
0.68	0.99 →

بازه ها را چطور می توانیم به دست آوریم که بازه های اعداد را [0.1, 0.2] است می توانیم زیر بازه های با طول 0.1 داشته باشیم که این کار را می توانیم به دست آوریم

Subject:

Year. Month. Date. ()

Bucket Sort (A)

$n \leftarrow \text{length}[A]$

for $i \leftarrow 1$ to n do

insert $A[i]$ into list $B[\lfloor n * A[i] \rfloor]$

for $i \leftarrow 0$ to $n-1$ do

sort list $B[i]$ with insertion sort

concatenate the lists $B[0], B[1], \dots, B[n-1]$ together in order

$E(T_n) = \Theta(n)$

8.4.4 Θ

The Selection Problem

min = ارشدترین

max = پایینترین

$\text{ارشدترین و پایینترین}$

n عدد داریم که می‌خواهیم عددی را پیدا کنیم که دقیقاً از آن بزرگتر یا کوچکتر باشد.
این کار را با استفاده از select می‌توانیم انجام دهیم. $\Theta(n \lg n)$

می‌توانیم الگوریتمی بنویسیم که order آن n باشد.

در C++ برای این کار nth_element داریم.

برای min و max می‌توانیم به این روش عمل کنیم.

$$T(n) = 2(T(n/2)) + \Theta(1)$$

$\min \leftarrow i = 1$

$$T(n) = \Theta(n)$$

$\max \leftarrow i = n$

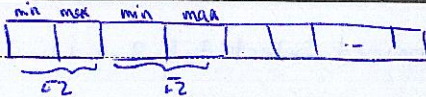
برای min و max می‌توانیم به این روش عمل کنیم. $2n-2$ مقایسه نیاز داریم که بزرگتر یا کوچکتر است.

در این روش برای min و max می‌توانیم به این روش عمل کنیم. max داریم و min داریم. min و max می‌توانیم به این روش عمل کنیم.

$\min$ و max می‌توانیم به این روش عمل کنیم.

Subject:

Year. Month. Date. ()



$\frac{n}{2}$

به دست می آید که برای هر دو حالت $\frac{n}{2}$ و $\frac{n}{2} - 1$ به دست می آید.

$$\frac{n}{2} + (\frac{n}{2} - 1) \times 2 = \frac{3n}{2} - 2$$

$$\frac{1}{2} + \frac{n}{4} + (\frac{n}{4} - 1) \times 2$$

$$\frac{3n}{4} + \frac{2n}{4} - 2 = \frac{5n}{4} - 2$$

9.1.1 $\frac{5n}{4} - 2$

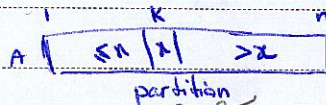
Selection Problem

1- به دست آوردن k امین بزرگترین
عنصر از یک آرایه

$$O(n \lg n)$$

sort

2- انتخاب k امین



sort $\rightarrow$ sort $\rightarrow$ sort

این کار به دست می آید که k امین بزرگترین عنصر از آرایه A است.

Randomized-Select (A, p, r, i)

if $p = r$ then

$i = k$ $\rightarrow$ به دست می آید که k امین بزرگترین عنصر از آرایه A است.

return $A[p]$

$q \leftarrow \text{Randomized-partition}(A, p, r) \rightarrow$ به دست می آید که q امین بزرگترین عنصر از آرایه A است.

این کار به دست می آید که k امین بزرگترین عنصر از آرایه A است.

$k \leftarrow q - p + 1$

if $i = k$ then return $A[q]$

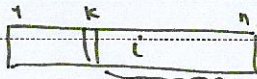
Subject:

Year. Month. Date. ()

else i < k then return randomized Select(A, P, q-1, i)

else return randomize Select(A, q+1, r, i+k)

برای مثال (A) این عضو داریم در زیر یک آرایه ای که از 1 تا n است. اگر i است زیرا
قبل از آن که عضو وجود داشته و وقتی که k + 1 شود از این عضو در آرایه اصلی



2. quick sort کار اجرای آن و سیستم به نوع partition است.

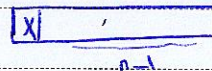
مثال

partition متوازن :

$$T(n) = \theta(n) + T(n/2) = \theta(n \log n) = \theta(n \log n)$$

تقسیم 3 ریشه اصلی در فراخوانی متوازن یک فراخوانی داریم یا چپ یا راست در هر quick sort هم چپ و راستی شود
هم راست

partition نامتوازن :



$$T(n) = \theta(n) + T(n-1) = \theta(n^2)$$

این روشی بود که نامتوازن است این partition که کسی از بدترین است متوازن است و متوازن
زیادتر است که یک طرف constant است این که کسی با بدترین است که یک طرف است که در

سایر partition و pivot و سبب به نام pivot سیستم است که در آن یک طرف است و در آن صورت
توزیع partition نامتوازن خواهد بود.

Subject:

Year. Month. Date. ()

الگوریتمی که همیشه حالت متوازن را تضمین می‌کند و order آن همیشه n است

Selection

الگوریتم

1. اعداد را به $\lceil \frac{n}{5} \rceil$ گروه 5 تایی با اضافه شدن صفر به گروه‌های کمتر از 5 است تقسیم می‌کنیم

2. میانه هر یک از $\lceil \frac{n}{5} \rceil$ گروه را می‌گیریم (مثلاً با استفاده از Insertion sort)

3. الگوریتم select را به صورت بازگشتی برای یستمه میانه میانه‌های فوق فرضی می‌کنیم

چون کار select همین است

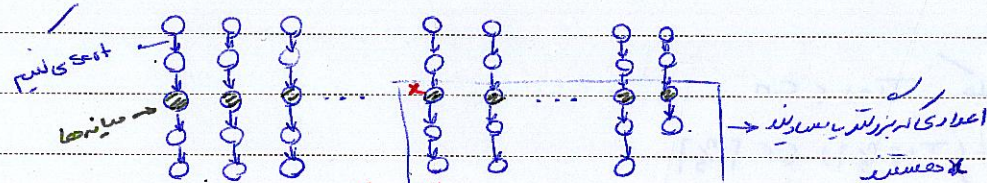
4. برای دوری از بازگشت به از جواب $\lceil \frac{n}{5} \rceil$ (2 میانه میانه‌ها) partition می‌کنیم فرض کنیم از پس

آن کار می‌شود

5. اگر کار n باشد و ما برای دوری از بازگشت الگوریتم select را چسب می‌زنیم برای یستمه اول

و برای فرضی میانه‌ها اگر کار n باشد الگوریتم select را دوری می‌زنیم و برای یستمه دوم کار n است این

عملیات فرضی می‌کنیم



الگوریتمی که همیشه میانه‌ها را میانه میانه‌هاست

تمام گروه‌های که میانه‌ها را از n که چقدر است در نهایت جب و آنهایی که بزرگترند و کوچکتر است الگوریتم

میانه است ← نصف گروه‌ها بزرگتر از n

کوچکتر از n

$$\left(\frac{\frac{n}{5}}{2} \right) * 3 = \frac{3n}{10} \rightarrow \text{صاف}$$

$$\frac{3n}{10} - 6 \leq \leq \frac{7n}{10} + 6$$

Subject:

Year. Month. Date. ()

Select

نشان اجرای الگوریتم را بکلیل بنویسید

$$T(n) = O(n) + T(\lfloor \frac{n}{5} \rfloor) + T(\lfloor \frac{7n}{10} \rfloor + b)$$

$$1 \text{ f.b. } O(n)$$

$$2 \text{ f.b. } O(n)$$

$$3 \text{ f.b. } T(\lfloor \frac{n}{5} \rfloor)$$

$$\text{partition } 4 \text{ f.b. } O(n)$$

$$5 \text{ f.b. } T(\lfloor \frac{7n}{10} \rfloor + b)$$

$$T(\lfloor \frac{7n}{10} \rfloor + b) \text{ (اینجا به جای } T(\lfloor \frac{n}{5} \rfloor) \text{ می‌نویسیم)}$$

max قیمت حساب می‌کنیم

$$\frac{3n}{10} - b \leq \text{اندازه هر قسمت} \leq \frac{7n}{10} + b$$

اینجا به جای $\frac{3n}{10}$ می‌نویسیم (اینجا به جای $\frac{7n}{10}$ می‌نویسیم)

این کار حساب می‌کند تا به الگوریتم با balance شود و $O(n)$ شود

$$T(n) \leq cn$$

$$T(\lfloor \frac{n}{5} \rfloor) \leq c \lfloor \frac{n}{5} \rfloor$$

$$T(\lfloor \frac{7n}{10} \rfloor + b) \leq c(\lfloor \frac{7n}{10} \rfloor + b)$$

$$\leq \frac{n}{5} + 1$$

$$T(n) \leq an + c \lfloor \frac{n}{5} \rfloor + c(\lfloor \frac{7n}{10} \rfloor + b)$$

$$\leq an + c(\frac{n}{5} + 1) + c \frac{7n}{10} + bc$$

$$\leq an + \frac{9cn}{10} + 7c$$

$$\leq cn + (\frac{-cn}{10} + 7c + an)$$

برای اینکه این کلمه به این الگوریتم ≤ 0 باشد

$$\frac{-cn}{10} + 7c + an \leq 0$$

$$c(\frac{n}{10} - 7) \geq an$$

Subject:

Year. Month. Date. ()

CP

$$n > 70$$

$$C \geq 1.0 \frac{n}{n-70}$$

$$n = 140 \Rightarrow C = 2.10a$$

$$\forall n \geq 140 : \frac{n}{n-70} \leq 2$$

$$C \geq 1.0 \cdot a \cdot 2 > 1.0 \cdot a \cdot \frac{n}{n-70}$$

برای هر constant a می توانیم C را انتخاب کنیم که این رابطه برقرار باشد

ما می خواهیم یک رابطه بسازیم که از n مستقل باشد

و برای چنین n این کار می کنیم

برای $n = 140$ است که این رابطه برقرار است

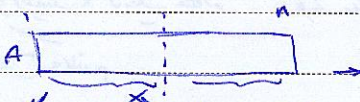
تمرین 9.3-8 و 9.3-9

روش های حل مساله 9

1. روش تقسیم و حل Divide & conquer
2. برنامه ریزی پویا dynamic programming
3. روش حریص Greedy
4. جستجوی فضای حالات back tracking

طرح های اصلی روش تقسیم و حل: (حال در دست بارش می آید)

1. تقسیم به زیر مسایل به شیوه ای که حجم کاری تقسیم کنیم بهتر است
2. حل بازگشتی زیر مسایل و حل اینده زیر مسایل حل شدن نقطه به نقطه به شرط بازگشت به توضیح کنیم
3. ترکیب جواب زیر مسایل و درست کردن جواب اصلی



برای merge sort تقسیم و حل Partition می کنیم که همان تقسیم به دو بخش است و به عنوان pivot داریم

برای حل مساله ای تقسیم و حل به طوری که تقسیم کنیم

شماره کار در این روش 1 و 2 است.

Subject:

Year.

Month.

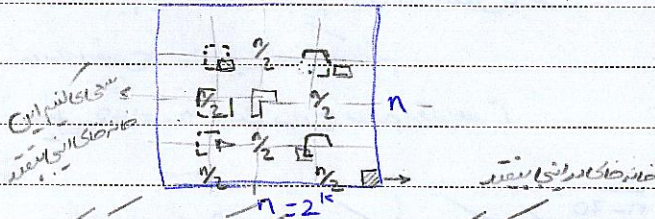
Date.

()

پرسش در خصوص تست خودرسانه ای

مثال ۱. به سبب طرح داریم فرض کنیم $n = 2^k$

با فرض اینکه برای شکل ۱ از این تست



با فرض اینکه برای شکل ۱ از این تست داریم فرض کنیم $n = 2^k$

با فرض اینکه برای شکل ۱ از این تست داریم فرض کنیم $n = 2^k$

با فرض اینکه برای شکل ۱ از این تست داریم فرض کنیم $n = 2^k$

$$2^k \times 2^k = 4^k$$

$$2^k \times 2^k = 4^k$$

$$4^k \bmod 3 = 1$$

فرض است که 4^k به ۳ بخش پذیر است

توجه: اگر 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در این تست، هرگاه 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

در تست ۳، فرض است که 4^k به ۳ بخش پذیر است، پس $4^k \bmod 3 = 0$ است.

Subject:

Year. Month. Date. ()

Fill (A, n, Flag)

if $n \leq 2$

it flag = BR

$AL[7:0] = AL[3:0] = AL[17:0] = \text{counter}++$

else if

}

A_1, A_2, A_3, A_4 از روی A می سازیم

Fill (A, $\frac{n}{2}$, BR)

Fill (A, $\frac{n}{2}$, BL)

Fill (A, $\frac{n}{2}$, TR)

Fill (A, $\frac{n}{2}$, TL)

Flag برای انجام عملیات خازن سازی کاربرد دارد.

در هر مرحله از روی آرایشی که داریم باید اصل

حفظه است که در هر مرحله از روی اصل

در غیر این صورت باید در هر مرحله از روی اصل

مسئله: حساب مسابقات: n تیم در هر مرحله از روی اصل

رتبه	1	2	3
تیم	1	2	3
1	1	2	3
2	2	3	1
3	3	1	2

A B

$\frac{n}{2}$ $\frac{n}{2}$

همه A ها با هم بازی می کنند و نتایج با هم بازی می کنند در B هستند

رتبه	1	2	3	...	$\frac{n}{2}$
تیم	1	2	3	...	$\frac{n}{2}$
1	1	2	3	...	$\frac{n}{2}$
2	2	3	4	...	$\frac{n}{2}$
3	3	4	5	...	$\frac{n}{2}$

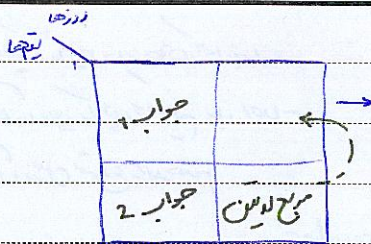
$\frac{n}{2}$ 1 2 3 ... $\frac{n}{2}$

هر یک هم با هم بازی می کنند و نتایج با هم بازی می کنند

به صورت چرخشی بازی می کنند و در هر مرحله از روی اصل

Subject:

Year. Month. Date. ()



برای به دست آوردن این نتیجه
از باز بررسی کنیم

نتیجه اول با $n-1$ تیم بازی کنند

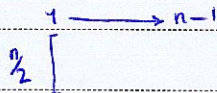
" دوم " $n-2$ " "

" n " " "

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

تعداد کل بازی‌ها (مسابقات)

اگر n زوج باشد در هر روز $\frac{n}{2}$ بازی می‌شود. می‌توان در $n-1$ روز تیم‌ها را زمان بندی کرد



در روز اول بازی
 $n+1-1=n$
ی‌شود

اگر n فرد باشد به تیم‌های اضافی می‌شود

Subject:

Year. Month. Date. ()

مثال: فرض کنیم 6 تیمی خواهند با هم مسابقه بدهند

تیم‌ها

	1	2	3	4	5
1		2	3	4	5
2	1		5	3	6
3	6	1	2	5	4
4	5	6	1	2	3
5	4	3	6	5	1
6	3	4	5	1	2

+3

دوره تیم‌ها

1	2	3	4
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

1 اضافه راست جدول می شود

جدول تیم 7 می توان 7 را حذف کرد

سپار جدول 7

1	2	3	4
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

جدول زمان بندی 3 تیم در جدول اصلی

قراری داریم

یک جدول بازی به ترتیب اضافه می کنیم تا به جدول اصلی جدول بازی را به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم. جدول بازی را به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم. جدول بازی را به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم.

اول جدول بازی از جدول اصلی را بازی ها به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم. جدول بازی را به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم. جدول بازی را به دست آوریم. روزهای بازی را در جدول بازی قرار می دهیم.

Subject:

Year. Month. Date. ()

$n=8$

1	2	3	4	5	6	7	8
2	1	4	3	8	5	6	7
3	4	1	2	7	8	5	6
4	3	2	1	6	7	8	5
5	6	7	8	1	2	3	4
6	5	8	7	4	1	2	3
7	8	5	6	3	4	1	2
8	7	6	5	2	3	4	1

$+4$

۱- زیر مسئله اول
۲- زیر مسئله دوم

$$T(n) = 2T(n/2) + O(n^2) = \Theta(n^2)$$

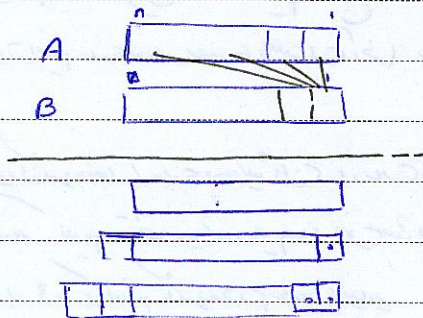
$T(n)$ هزینه کل است.

$$T(n) = O(n^2) + 4T(n/2) + O(1) = \Theta(n^2 \lg n)$$

اگر ما هزینه های زیر مسئله را در نظر بگیریم، هزینه کل را می توانیم به این صورت بیان کنیم:

$$T(n) = O(1) + 4T(n/2) = \Theta(n^2)$$

هزینه کل در هر مرحله:



$$\text{تعداد کارهای کل} = \Theta(n^2)$$

$$\text{تعداد بارها} = \Theta(n^2)$$

در هر مرحله، ما هزینه جمع می کنیم و در هر مرحله، هزینه کل را به دست می آوریم.

ما در این تقسیم و حل، زیر مسئله را به صورتی می بینیم که هزینه کل در هر مرحله، هزینه کل از مرحله قبل است.

هزینه کل در هر مرحله، هزینه کل از مرحله قبل است.

$$A = [A_1 \mid A_2]$$

$$B = [B_1 \mid B_2]$$

Subject:

Year. Month. Date. ()

حالا باید این کار را برای ضرب در 2^{n/2} تکرار کنیم

$$A \times B = (A_2 + A_1 \times 2^{n/2})(B_2 + B_1 \times 2^{n/2})$$

$$= A_2 \times B_2 + (A_2 \times B_1 + A_1 \times B_2) \times 2^{n/2} + A_1 \times B_1 \times 2^n$$

تعداد ضرب های لازم: $M(n) = 4M(n/2)$

له زیر مسئله های به حجم کم ساینده شدن 1/2 است

$$M(n) = \Theta(n^2)$$

تعداد ضرب های لازم: $T(n) = 4T(n/2) + \Theta(n)$

تعداد ضرب های لازم: $= \Theta(n^2)$

(جمع ضرب های لازم)

استفاده از روش تقسیم و فتح در حل مسائل محلی است. order را به این می آوریم تا بتوانیم به راحتی ضرب های لازم را در آن قرار دهیم. (نسبت به مسائل محلی)

به 4 ضرب داریم که می توانیم به 3 ضرب کاهش دهیم.

$$A_1 \times B_2 + A_2 \times B_1 = (A_1 - A_2)(B_2 - B_1) + A_2 \times B_2 + A_1 \times B_1$$

در این ضرب 2 و 3 حساب شده و در آخر تعداد ضرب ها 3 تا می شود

$$M(n) = 3M(n/2)$$

$$= \Theta(n^{\lg 3})$$

Subject:

Year. Month. Date. ()

: strassen

استراسن

$$C = A_{n \times n} \times B_{n \times n}$$

کامپیوتر سیستم این کار را به صورت ضرب انجام می‌دهد

$$\begin{bmatrix} \text{---} \\ \text{---} \end{bmatrix} \times \begin{bmatrix} \text{---} \\ \text{---} \end{bmatrix} = \begin{bmatrix} \text{---} \\ \text{---} \end{bmatrix}$$

A B C

هر دو این C از ضرب یک سیستم در یک سیستم است

این سیستم C، n ضرب و n جمع

نیاز دارد

$$n^2 \times n = n^3$$

$$\text{مثال } n^2(n-1) = n^3$$

با روش تقسیم ضرایب می‌توان این هزینه‌ها را کاهش داد.

ماتریس را به دو بخش تقسیم می‌کنیم و هر یک را با جابجایی می‌کنیم

$$\begin{bmatrix} [A_{11}] & [A_{12}] \\ [A_{21}] & [A_{22}] \end{bmatrix} \times \begin{bmatrix} [B_{11}] & [B_{12}] \\ [B_{21}] & [B_{22}] \end{bmatrix} = \begin{bmatrix} [C_{11}] & [C_{12}] \\ [C_{21}] & [C_{22}] \end{bmatrix}$$

برای حل این مسئله از یک سری تغییراتی که می‌کنیم (استراسن) استفاده می‌کنیم

$$Q_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$

$$Q_2 = (A_{21} + A_{22}) \times B_{11}$$

$$Q_3 = (A_{11}) \times (B_{12} - B_{22})$$

$$Q_4 = A_{22} \times (B_{21} - B_{11})$$

$$Q_5 = (A_{11} + A_{12}) \times B_{22}$$

$$Q_6 = (-A_{11} + A_{21}) \times (B_{11} + B_{12})$$

$$Q_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$

$$C_{11} = Q_1 + Q_4 - Q_5 - Q_7$$

$$C_{12} = Q_3 + Q_5$$

$$C_{21} = Q_2 + Q_4$$

$$C_{22} = Q_1 + Q_3 - Q_2 + Q_6$$

Subject:

Year: Month: Date: ()

$$M(n) = 7M(n/2)$$

لرینه‌ها در ضربها

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$$

$$C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$$

$$\Rightarrow M(n) = 8M(n/2)$$

$$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21}$$

$$C_{22} = A_{21} \times B_{12} + A_{22} \times B_{22}$$

Strassen: 8 ضرب راه 7 ضرب محسوب می‌شود

روش برنامه‌ریزی پویا: Dynamic programming.

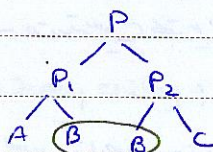
2 ویژگی مسائلی که برای روش حل می‌شود:

1. مقادیر 1 و 2 مسئله کینه‌ساز می‌باشد. (min, max)

2. جواب کینه‌ساز از روی جواب کینه‌ساز دیگر به دست می‌آید. (برای اینکه مسئله کینه‌ساز حل شود)

optimal substructure (بهترین مسئله هم کینه‌ساز حل می‌شود)

3. اجزای مسئله 3 زیرمسئله تکراری زیاد داشته باشد



این زیرمسئله دوبار حل می‌شود (تکرار می‌شود)

آنگاه روش تقسیم و حل به روش 3 تبدیل می‌شود

و در این روش هر زیرمسئله بار حل می‌شود

در بهترین مسئله ما روشی داریم و جواب آنجا را پیدا می‌کنیم و در یک حافظه می‌نویسیم و بعد از آن وقت که دوباره به آن نیاز داریم، آن را از روی جوابی که قبلاً به دست آورده‌ایم می‌خوانیم تا به مسئله اصلی برسیم

برای هر مسئله که ما این روش حل می‌شود و جوابی به دست می‌آید و در یک حافظه می‌نویسیم و بعد از آن وقت که دوباره به آن نیاز داریم، آن را از روی جوابی که قبلاً به دست آورده‌ایم می‌خوانیم تا به مسئله اصلی برسیم

بهترین زیرمسئله که به دست می‌آید را در یک حافظه می‌نویسیم و بعد از آن وقت که دوباره به آن نیاز داریم، آن را از روی جوابی که قبلاً به دست آورده‌ایم می‌خوانیم تا به مسئله اصلی برسیم

Year. Month. Date. ()

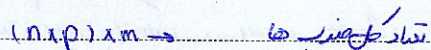
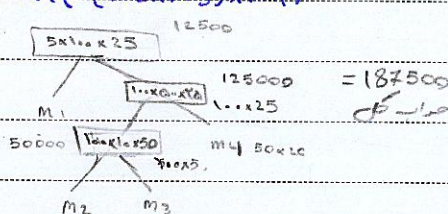
مسئله ۱) شرکت انبساطی را با توجه به اطلاعات زیر در نظر بگیرید

مخبرہ پرائیڈنڈاری درجہ اولیٰ صوبہ سندھ

$$M_1 \times M_2 \times M_3 \times M_4$$

$$5 \times 100 \quad 100 \times 1 \quad 1 \times 50 \quad 50 \times 25$$

$$m_1((m_2 \times m_3) \times m_4)$$



برای ضرب متناهیات کثیرین برانند لذری کدام است؟

برای تشخیص این که آیا یک رابطه ی خطی است یا نه می توانیم از روش (1) و (2) استفاده کنیم

① 62 ← 2 min 0.25

حضرت ابو ذر غفاری رضی اللہ عنہما

برائے نذرین کا حصہ

میدان برای اندازه گیری P_1 و P_2 هم یکسان باشد

ضرب آخر + ضرب P_2 + ضرب P_1 = ضرب

الم $P_1 P_2$ كهيئة مثلثية مستقيمة يسند في التوازي ، ندر كهيئة صائبة لئلا نسلم له من زاوية كل واحد للمثلث

Subject:

Year. Month. Date. ()

$$M_1 \times (M_2 \times M_3 \times M_4 \times M_5)$$

$$M_2 \times (M_3 \times M_4 \times M_5)$$

$$(M_1 \times M_2) \times (M_3 \times M_4 \times M_5)$$

این عملیات را می‌توان به روش دیگری نوشت
به صورت زیر

$$M_{ij} = M_{i+1} \times M_{i+2} \times \dots \times M_j$$

$$d_{i-1} \times d_i \times d_{i+1} \times d_{i+2}$$

این ضرب را می‌توانیم به روش دیگری بنویسیم

مسئله را حل می‌کنیم

این ضرب

$$(M_{i+1} \times M_{i+2} \times \dots \times M_k) \times (M_{k+1} \times \dots \times M_j)$$

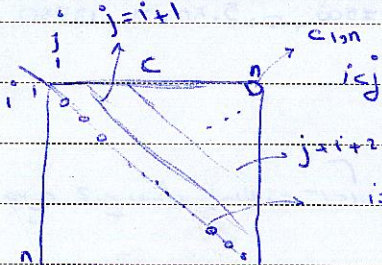
$$C_{ij} \Rightarrow M_{ij} \text{ مقدارهای مختلف در } M_{ij}$$

$$C_{ij} = C_{ik} + C_{kj} + d_{i-1} \times d_k \times d_j \quad i \leq k \leq j$$

$$C_{ij} = \min_{i \leq k \leq j} \{ C_{ik} + C_{kj} + d_{i-1} \times d_k \times d_j \}$$

چون C دایره‌ای است باید به این ترتیب بنویسیم

$$i=j \rightarrow C_{ii} = 0$$



$$i=j \rightarrow C_{ij} = 0$$

این را برای هر مقدار i می‌نویسیم
تا زمانی که $i=j$

مسئله اصلی $M_{1,n}$ است و می‌توانیم به این روش بنویسیم

Subject:

Year.

Month.

Date.

()

$$M_1 \times (M_2 \times M_3 \times M_4 \times M_5)$$

$$M_2 \times (M_3 \times M_4 \times M_5)$$

$$(M_1 \times M_2) \times (M_3 \times M_4 \times M_5)$$

این روش برای ماتریسهای کوچک است
ماتریسهای بزرگ را نمیتوان به این روش حل کرد

$$M_{ij} = M_{i1} \times M_{1i+1} \times \dots \times M_{j-1j}$$

در اینجا d_i و d_{i+1} و d_{j-1} و d_j

این ضرایب برای ضرایب ماتریسهای کوچک است

این ضرایب

ماتریسهای کوچک را میتوان به این روش حل کرد

$$(M_{i1} \times M_{1i+1} \times \dots \times M_{k-1k}) \times (M_{k1} \times \dots \times M_{j-1j})$$

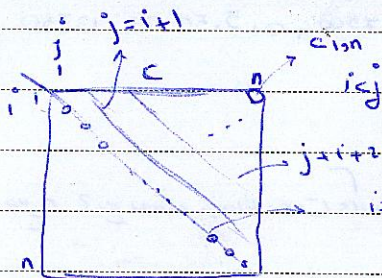
$$C_{ij} \rightarrow M_{ij} \text{ در این ضرایب جایگزین می شود}$$

$$C_{ij} = C_{ik} + C_{k+j} + d_i + d_k + d_j \quad i < k < j$$

$$C_{ij} = \min \{ C_{ik} + C_{k+j} + d_i + d_k + d_j \mid i < k < j \}$$

چون C دایره ای است باید به این ترتیب برای آن محاسبه کنیم

$$i=j \rightarrow C_{ii} = 0$$



این برای مرتبه n است و این ضرایب را میتوان به این روش حل کرد

ماتریس $M_{1,n}$ است و این ضرایب را میتوان به این روش حل کرد

Subject:

Year.

Month.

Date.

()

$$M_1 \times (M_2 \times M_3 \times M_4 \times M_5)$$

$$M_2 \times (M_3 \times M_4 \times M_5)$$

$$(M_1 \times M_2) \times (M_3 \times M_4 \times M_5)$$

$$M_{ij} = M_1 \times M_{i+1} \times \dots \times M_j$$

$$d_{i-1} \times d_i \times d_{i+1} \times d_{i+2}$$

$$d_j - 1 \times d_j$$

این ضرب را می توانیم به این روش بنویسیم

این ضرب را می توانیم به این روش بنویسیم

مسئله اصلی این است که چگونه می توانیم

$$(M_i \times M_{i+1} \times \dots \times M_k) \times (M_{k+1} \times \dots \times M_j)$$

$$d_{i-1} \times d_i$$

$$d_{k-1} \times d_k \times d_{k+1}$$

$$d_j - 1 \times d_j$$

$$C_{ij}$$

این ضرب را می توانیم به این روش بنویسیم

تفاوت های بین این دو روش

تفاوت های بین این دو روش

$$C_{ij} = C_{ik} + C_{k+1j} + d_{i-1} \times d_k \times d_j$$

$$i \leq k \leq j$$

$$C_{ij} = \min \{ C_{ik} + C_{k+1j} + d_{i-1} \times d_k \times d_j \}$$

$$i \leq k \leq j$$

فرض کنیم

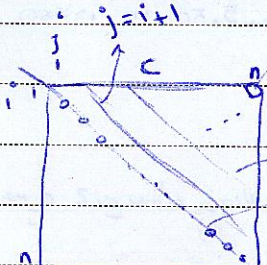
چون که در این روش داریم یک رابطه بین این دو روش

$$i=j \rightarrow C_{ii} = 0$$

$$C_{ii} = 0$$

برای هر

این رابطه را می توانیم به این روش بنویسیم



$$C_{ij}$$

$$C_{ij}$$

$$C_{ij} = 0$$

این رابطه را می توانیم به این روش بنویسیم

این رابطه را می توانیم به این روش بنویسیم

مسئله اصلی این است که چگونه می توانیم

Subject:

Year. Month. Date. ()

i \ j	1	2	3	4
1	0	5000 k=1	7500 k=2	13750 k=2
2		0	50000 k=2	137500 k=2
3			0	12500 k=3
4				0

1.02

$$C_{1,2} = \min \{ C_{1,k} + C_{k+1,2} + d_1 \cdot d_k \cdot d_2 \}$$

$$k=1 \leq k \leq 2$$

$$C_{1,2} = C_{1,1} + C_{2,2} + d_1 \cdot d_1 \cdot d_2 = 5 \times 100 \times 10$$

$$C_{2,3} = C_{2,2} + C_{3,3} + d_1 \cdot d_2 \cdot d_3 = 100 \times 10 \times 50 = 50000$$

$$C_{3,4} = C_{3,3} + C_{4,4} + d_1 \cdot d_3 \cdot d_4 = 10 \times 50 \times 25 = 12500$$

$$M_1 \times M_2 \times M_3 \times M_4$$

$$5 \times 100$$

$$100 \times 10$$

$$10 \times 50$$

$$50 \times 25$$

$$d_1 \cdot d_1$$

$$d_1 \cdot d_2$$

$$d_2 \cdot d_3$$

$$d_3 \cdot d_4$$

حسابات زیر را در جدول زیر وارد کنید

$$C_{1,3} = \min \begin{cases} k=1 \rightarrow C_{1,1} + C_{2,3} + d_1 \cdot d_1 \cdot d_3 & 1 \leq k < j \\ k=2 \rightarrow C_{1,2} + C_{3,3} + d_1 \cdot d_2 \cdot d_3 & 1 \leq k < 3 \end{cases}$$

$$= 0 + 50000 + 5 \times 100 \times 50 = 75000$$

$$= \min(75000, 75000)$$

$$k=2 \rightarrow C_{1,2} + C_{3,3} + d_1 \cdot d_2 \cdot d_3$$

$$= 5000 + 0 + 5 \times 10 \times 50 = 7500$$

$$C_{2,4} = \min \begin{cases} k=2 \rightarrow C_{2,2} + C_{3,4} + d_1 \cdot d_2 \cdot d_4 \\ k=3 \rightarrow C_{2,3} + C_{4,4} + d_1 \cdot d_3 \cdot d_4 \end{cases}$$

$$= 0 + 12500 + 100 \times 10 \times 25 = 37500$$

$$2 \leq k < 4$$

$$k=3 \rightarrow C_{2,3} + C_{4,4} + d_1 \cdot d_3 \cdot d_4 = 50000 + 0 + 10 \times 50 \times 25 = 77500$$

$$C_{1,4} = \min$$

$$1 \leq k < 4$$

$$k=1 \rightarrow C_{1,1} + C_{2,4} + d_1 \cdot d_1 \cdot d_2$$

$$= 0 + 37500 + 5 \times 100 \times 25 = 50000$$

$$k=2 \rightarrow C_{1,2} + C_{3,4} + d_1 \cdot d_2 \cdot d_4 = 5000 + 12500 + 5 \times 10 \times 25 = 18750$$

$$k=3 \rightarrow C_{1,3} + C_{4,4} + d_1 \cdot d_3 \cdot d_4 = 7500 + 0 + 5 \times 50 \times 25 = 13750$$

$$((M_1 \times M_2) \times M_3) \times M_4$$

آخرین 2 زیر مندرج را حاصل به دست می آید به همین ترتیب

Subject:

Year. Month. Date.

$$M_1 \times (M_2 \times M_3 \times M_4 \times M_5)$$

$$M_2 \times (M_3 \times M_4 \times M_5)$$

$$(M_1 + M_2) \times (M_3 \times M_4 \times M_5)$$

$$M_{ij} = M_i \times M_{i+1} \times \dots \times M_j$$

این ضرب را می توانیم به این ترتیب بنویسیم

$$(M_i \times M_{i+1} \times \dots \times M_k) \times (M_{k+1} \times \dots \times M_j)$$

$$C_{ij} \rightarrow$$

تفاضل های مجانبه در M_{ij}

فرمول ضرب

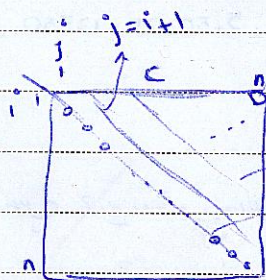
$$C_{ij} = C_{ik} + C_{k+1,j} + d_i \cdot d_k \cdot d_j \quad i \leq k \leq j$$

$$C_{ij} = \min \{ C_{ik} + C_{k+1,j} + d_i \cdot d_k \cdot d_j \mid i \leq k \leq j \}$$

فرمول ضای

جدول C دو تا ایندیس دارد که یک باترینس برای آن تعیین می کنیم

$$i=j \rightarrow C_{ii} = 0$$



$$i=j \rightarrow C_{ij} = 0$$

این را برای مرتبه n از طریق اختلاف باترینس می کنیم

که آنقدر می یابیم

مسئله اصلی $M_{1,n}$ است جدول C را می یابیم

Subject:

Year. Month. Date. ()

بنیاد ریاضیاتی

مسئله پیدا کردن طولانیترین زیرساخت مشترک:

$\vec{a} : a_1, a_2, \dots, a_n$
 $\vec{b} : b_1, b_2, \dots, b_m$

فرض کنیم $\vec{a}$ و $\vec{b}$ از object ها در دنباله اصلی به ترتیب

زیرساخت هر object خود دنباله اصلی است و ترتیب object ها به ترتیب زیرساخت اصلی باشد

$a : 1, 5, 2, 3, 1$

$b : 4, 1, 6, 2, 1$

زیرساخت $\vec{a}$ و $\vec{b}$ به ترتیب $1, 2, 1$ → (طولانیترین ساختار object مشترک)

اگر $\vec{a}$ و $\vec{b}$ object ها در دنباله اصلی باشند و $\vec{a}$ و $\vec{b}$ به ترتیب زیرساخت اصلی باشند و طولانیترین ساختار مشترک $\vec{a}$ و $\vec{b}$ object $\vec{a}$ و $\vec{b}$ باشد

longest Common Subsequence

اگر $a_n = b_m$ → وجود دارد $LCS(\vec{a}, \vec{b})$
 که حوت آخرین $(b_m = a_n)$ باشد

$a_n = b_m \rightarrow LCS(\vec{a}, \vec{b}) = LCS(\vec{a}_{n-1}, \vec{b}_{m-1}) + a_n$

اگر $a_n \neq b_m$ → $LCS(\vec{a}, \vec{b}) = \max \{ LCS(\vec{a}_{n-1}, \vec{b}), LCS(\vec{a}, \vec{b}_{m-1}) \}$
 ② " " " " $b_m \rightarrow LCS(\vec{a}, \vec{b}_{m-1}) = LCS(\vec{a}_{n-1}, \vec{b}_{m-1})$
 ③ " " " " $a_n \rightarrow LCS(\vec{a}_{n-1}, \vec{b}_{m-1}) = LCS(\vec{a}_{n-1}, \vec{b}_{m-1})$

P_{ij}

$\vec{a} : a_1, \dots, a_i$
 $\vec{b} : b_1, \dots, b_j$

طولانیترین زیرساخت مشترک L_{ij}

$L_{ij} = \begin{cases} L_{ij-1} + 1 & \text{اگر } a_i = b_j \\ \max \{ L_{i-1, j}, L_{i, j-1} \} & \text{اگر } a_i \neq b_j \\ 0 & \text{اگر } i=0 \text{ یا } j=0 \end{cases}$

زیرساخت $L_{i-1, j}$ و $L_{i, j-1}$ → $L_{ij} = \max \{ L_{i-1, j}, L_{i, j-1} \}$

طُفْلٌ وَنُفُوسٌ

از ناله به پاس ارجح به راست متبرعین اسپی لندم

خود زبر رشت نه

خانه: زیرا با استفاده از اصل برتری نسبی، اگر $\frac{a}{b} \leq \frac{c}{d}$ باشد، آنجا که a, b, c, d اعداد صحیح و $b, d \neq 0$ باشند، می‌توانیم بنویسیم $ad \leq bc$.
 زیرا برای هر دو عدد صحیح a, b می‌توانیم بنویسیم $a \leq b$ یا $b < a$ و از آنجا که $a \leq b$ یا $b < a$ می‌توانیم بنویسیم $a \leq b$ یا $b < a$ و از آنجا که $a \leq b$ یا $b < a$ می‌توانیم بنویسیم $a \leq b$ یا $b < a$.

در یک برد $n \times n$ می‌توانیم $\alpha \neq \beta$ باشد برای قرار دادن n شش به این که $\max$ است از صریح نیست
 اگر α و β از n به $\max$ بود به جهت صریح و اگر $\alpha = \beta$ $\max$ بود به ملا قرار می‌دهیم

a: a b a a c d ba

سُئِلَ

mod 11

	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	9	7	5	3	1
3	0	3	6	9	12 → 1	15 → 4	18 → 7	21 → 10	24 → 2	27 → 5	30 → 8
4	0	4	8	12 → 1	16 → 5	20 → 9	24 → 2	28 → 6	32 → 10	36 → 3	40 → 7
5	0	5	10	15 → 4	20 → 9	25 → 4	30 → 8	35 → 3	40 → 7	45 → 10	50 → 5
6	0	6	12 → 1	18 → 7	24 → 2	30 → 8	36 → 3	42 → 6	48 → 9	54 → 1	60 → 5
7	0	7	14 → 2	21 → 10	28 → 6	35 → 3	42 → 6	49 → 5	56 → 1	63 → 9	70 → 4
8	0	8	16 → 5	24 → 2	32 → 10	40 → 7	48 → 9	56 → 1	64 → 4	72 → 6	80 → 8
9	0	9	18 → 7	27 → 5	36 → 3	45 → 10	54 → 1	63 → 9	72 → 6	81 → 0	90 → 0
10	0	10	20 → 9	30 → 8	40 → 7	50 → 5	60 → 5	70 → 4	80 → 8	90 → 0	100 → 1

۱- این خانه ای بیس خانه 3 و 2 ، از طریق max می سیدیم

هو و لي بويزند فوجي دارو دارو را در نظر بداريم

b1, 5

16

22

.....

= 95



-1-1

2

2

1

1

۲۲۲

14

19

100

یادداشتی کنیم

$a_6 = b_5$ به سادگی بداد

چون از آخر اولی، دوم، زیر دانه از آخری استوار

اندازیم چون آثارش خانه آخر قتل خود نیز رشانه طولانی هستند.

برای پیدا کردن خود زیرین به نام عشق جان عالم و حیات ی نسیم

[illegible]

Subject:

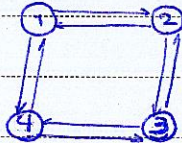
Year. Month. Date. ()

یک مسأله بزرگ را به زیرمسئله‌های کوچک‌تر تقسیم می‌کنیم.

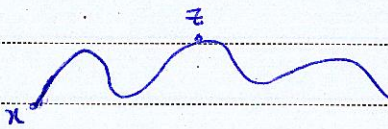
optimal substructure

مسئله اصلی می‌تواند از مسأله‌های بزرگ‌تر برای حل آن استفاده کند.

Longest Path



مسأله بزرگ را به مسأله‌های کوچک‌تر تقسیم می‌کنیم. max



طولانی‌ترین مسیر بین x و y را برای اینکه بتوانیم به راحتی

به دست آوریم، باید به کمک طولانی‌ترین مسیر بین x و z و طولانی‌ترین

مسیر بین z و y که اینطور نیست.

③ : طولانی‌ترین مسیر بین ① و ③
این یعنی طولانی‌ترین بین ① و ② و ② و ③

② : طولانی‌ترین مسیر بین ① و ②
① → ④ → ③ → ②

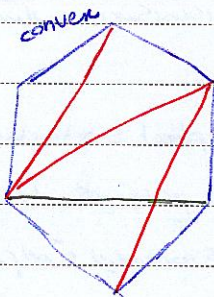
③ : طولانی‌ترین مسیر بین ② و ③
② → ① → ④ → ③

Subject:

Year. Month. Date. ()

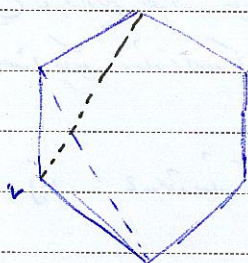
مسئله: مثلث بندی یک چند ضلعی کج است:

تقریر: $convex$ ۱: یعنی از زاویه بیرون کمتر از ۱۸۰



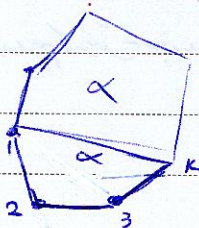
تقریر: هر ضلعی را رسم کنیم که بتوانیم مساحت را به چندین مثلث تقسیم کنیم. تقریر باید به چند ضلع تقسیم شود. مثلث ها در مثلث چند ضلعی باشند. به تعداد مثلث ها ثابت است $(n-2)$ حسب ترتیب که مجموع طول تقریرها $n-3$ شود.

یک چند ضلعی را به چند مثلثی تقسیم می توان مثلث بندی کرد؟



پایان ۷ تقریر درونی شود. باید تقریر از n رسم می شود. به دو این حالت می توان فرض کرد تقریر به واسطه یک دور می کشد که همان ضلع است.

رأس n گانه ۷ را در تقریر داریم و یک تقریر از آن رسم می کنیم که به دو قسمت تقسیم می کند. ضرب تعداد حالات مثلث بندی این دو قسمت جواب می شود.



$$\sum_{k=2}^n T(k) \cdot T(n-k+2)$$

این فرمول تعداد حالات تقریری را می دهد. اگر بخواهیم به سادگی (تقسیم به دو زیر مسئله) حل کنیم باید از همین روش برویم.

برای مثال: $(m_1 + m_2 + \dots + m_k) \cdot (m_{k+1} + \dots + m_n)$

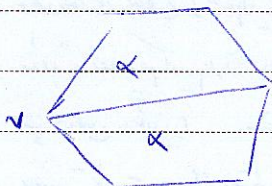
$$\sum_{k=1}^n T(k) \cdot T(n-k) = \frac{1}{n+1} \binom{2n}{n}$$

$$= \frac{1}{n+1} \binom{2n}{n} \frac{2n!}{n!n!} = \frac{(n+1)(n+2) \dots (2n)}{1 \cdot 2 \cdot 3 \dots n} \cdot \frac{1}{n+1}$$

Subject:

Year. Month. Date. ()

نکته لنیم (تعداد حالت‌های ممکن برای چند تا است؟)



ببینیم optional substructure برقرار است یا نه؟

یک جواب کلیه داریم برای اینکه جواب صحیح یا کلیه باشد
لازم است زیرمساله‌ها کلیه باشند $\leftarrow$ برنامه سازی پویا

راهنما: ۱) در تقریب داریم هر تقریبی که از v روی شوری تا آن کلیه باشد در زیرمساله تقسیم‌ی لنیم جوابکار
باید در بهترین حالت داریم جواب صحیح برای شوری جواب زیرمساله‌ها + تقریب
بهترین زیرمساله باید حالت است که جواب حاصل است درجه برای ۴ صلیبی درجه برای ۵ صلیبی چهار... حساب
ی لنیم



دقیقاً اختلاف از آن $\rightarrow$ به مقدار صحت است (از یک مجموعه‌ی شوری تا یک
ادامه پیدا می‌کند)

مساله درخت جستجوی دودویی (جمله) optimal Binary Search Tree

یک سری جمله داریم که درخت جستجو را می‌سازد و طریقه‌ی خرابی داریم در dictionary که دریم طوطی مثلج است
invalid

a_i جمله

P_i احتمال رخداد جمله درخت

b_i

$a_1 < a_2 < \dots < a_n$ n جمله معتبر

set S مجموعه جبراساس جدول انبساطی

$a_i < b_i < a_{i+1}$

لحظه‌ای که نتوانیم به درخت هشتمی و invalid هشتمی

آنچه اینجا search کنیم unsuccessful ضمیمه

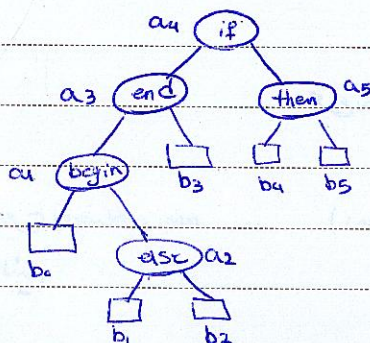
حالت‌های a_1 که به a_n می‌رسد یا به a_{n+1} می‌رسد و به a_n می‌رسد به a_{n+1} می‌رسد و به a_n می‌رسد

$b_0 \quad b_1 \quad b_{n-1} \quad b_n$

✓

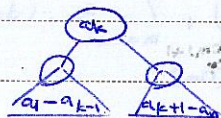
$\langle a_1 \quad a_2 \quad a_3 \quad \dots \quad a_n \quad \dots \quad a_n \quad \dots \quad a_n \rangle$

۹۔ جنتا کلمہ دارم کہ نہ عقیدہ است و نہ ایمان



صريح حاشیه صحیحی
مستوفح حسینه بیواها صوفی

کتاب آمده این است آن حکم را اقتضای نیست است در ابتدا باشد و همین طوری باین برسم
باید اول در نظر بگیریم کدام حکم ریشه باشد و subtree و subtree ریشه ها می باشند



صغیری then : if حق سیدی لکنم برکت دارا if صحت رستای ریم حق سیدی لکنم بهر است به ضیق

$$\text{متوسط مقایسه‌ای صلب برای صلبی صفت} = \sum_{i=1}^n p_i [\text{depth}(a_i) + 1]$$

برای b_2 ، 4 تا سقا سیم داریم . مقدار سقا سیم ها برای سقا سیم های b_1 و b_2 است

$$\sum_{i=1}^n q_i \cdot \text{depth}(b_i) = \text{متوسط عمق تعداد حقایق}$$

Subject:

Year. Month. Date. ()

یضا صیم جمع تعداد سیم های سرفوق و سرفوق min سرفوق

یضا صیم یکتا BST سیم از a_i تا a_j

$P_{i+1}, P_{i+2}, \dots, P_j$

$b_i, b_{j+1}, b_{j-1}, b_j$

$q_i, q_{i+1}, q_{j-1}, q_j$

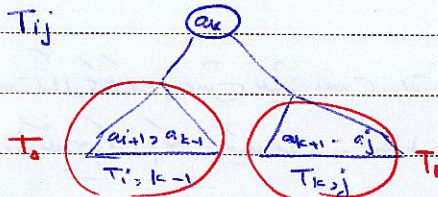
سطح سرفوق را دوست

جمع تعداد سیم های ایجا شده بصورت سرفوق یا سرفوق برای $T_{ij} = C_{ij}$ جواب مساله

T_{ij}

$$C_{ij} = \sum_{m=i+1}^j P_m (\text{depth}(a_m) + 1) + \sum_{m=i}^j q_m (\text{depth}(b_m))$$

این یضا صیم min کنیم



$$C_{ij} = \sum_{m=i+1}^{k-1} P_m (\text{depth}(a_m) + 1) + P_k (\text{depth}(a_k) + 1) + \sum_{m=k+1}^j P_m (\text{depth}(a_m) + 1) + \sum_{m=i}^{k-1} q_m (\text{depth}(b_m)) + \sum_{m=k}^j q_m (\text{depth}(b_m))$$

اینجا $C_{k,j}$ داریم

ارتباط میان درختان را می توانیم به این شکل بنویسیم: در درخت اصلی یکتا a_i کلی سیم های سرفوق آن در درختان است

$$C_{ij} = C_{i,k-1} + C_{k,j} + P_k + \sum_{m=i+1}^{k-1} P_m + \sum_{m=k+1}^j P_m + \sum_{m=i}^{k-1} q_m + \sum_{m=k}^j q_m$$

$$= C_{i,k-1} + C_{k,j} + \sum_{m=i+1}^j (P_m + q_m) + q_i$$

به آنز قبلی داریم که $1 = w_{ij}$

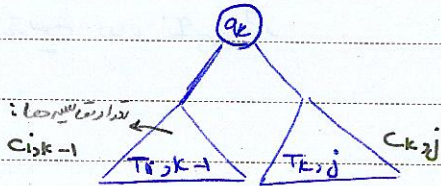
$$= C_{i,k-1} + C_{k,j} + w_{ij}$$

Subject:

Year. Month. Date. ()

اداره درخت ردی صحتجوی بهینه : OBST

بعد از ساخت گره ها، طمان تعداد زیادی درختی توان از روی آن ساخت و همین طور برای زیر درخت ها هم می توان هر node را به عنوان ریشه در نظر گرفت.



$$\begin{aligned} \text{حقیقت} \quad a_{i+k-1} &\leq a_{k-1} & a_{k+1} &\leq a_j \\ \text{حقیقت} \quad b_{i+k-1} &\leq b_{k-1} & b_k &\leq b_j \end{aligned}$$

$$C_{ij} = \sum_{m=i}^j p_m \cdot (\text{depth}(a_m) + 1) + \sum_{m=i}^j q_m \cdot \text{depth}(b_m)$$

این $\sum$ به سبب می باشد این $\sum$ به سبب می باشد

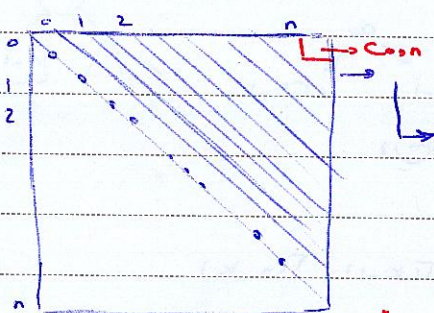
هر node در زیر درخت h است و درخت اصلی h است

ما فرض کنیم جواب بهینه aq است
باید برای اینکه هر node بهینه باشد حساب کنیم رنج $\min$ کنیم

$$C_{ij} = \min(C_{i,k-1} + C_{k,j}) + w_{i,j}$$

$i \leq k \leq j$

برای k مقدار k حساب کنیم و برای هر k که $\min$ شد آن k را same می کنیم



برای نگهداری این ها

حساب طریقه ها در این حالت است

به جواران تقریباً به یک شکل این می کنیم

نویسید به زیر می باشد

چون اصله حساب می ای ایام $\rightarrow a_i = 0$ از a_1 تا a_n درخت aq $\rightarrow$ درخت aq
 که در این aq به سبب می باشد k هیچ مقدار می توان از ریشه باشد
 در این k به حساب می بریم از ریشه aq به سبب می باشد

Subject:

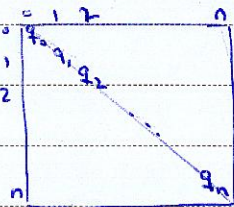
Year. Month. Date. ()

ی توانیم جمله آخرین را جدا کنیم

$$w_{ij} = \sum_{m=i+1}^j (p_m + q_m) + q_i = w_{i,j-1} + p_j + q_j$$

در حالتی که $i=j$ را بتوانیم از رابطه حساب کنیم

$$w_{ii} = q_i$$



قطر اصلی در سطرین و q_i سری شود

مثال (0.05)

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

$$a_1 < a_2 < a_3 < a_4 < a_5$$

برای p_i و q_i در سطرین و q_i در سطرین

در سطح 5 تا 5 داریم و q_i در سطرین و q_i در سطرین

$$N \sum_{i=1}^n T(i) \cdot T(n-i+1) = n! \cdot \frac{1}{n+1} \binom{2n}{n}$$



این برای دقیق است که در وقت با سطرین و q_i در سطرین

$$a_1 < a_2 < \dots < a_n$$

$$T(n) = \sum_{k=1}^n T(k-1) \cdot T(n-k)$$

Subject:

Year. Month. Date. ()

جدول را در هر مرحله شش
انتخاب کنیم از یک است

زیرین شش را

	0.30	0.45			
1	0.25	0.25			
2		0.15			
3			0.2		
4				0.35	
5					0.4

شش 1
شش 2
شش 3
شش 4
شش 5

$$C_{0,1} = \min \{ C_{0,1}^1 + C_{0,1}^2 \} + W_{0,1} = W_{0,1} + P_1 + q_1$$

$0 < k \leq 1$
شش 1

$$C_{1,2} = \min \{ C_{1,2}^1 + C_{1,2}^2 \} + W_{1,2} = W_{1,2} + P_2 + q_2$$

$1 < k \leq 2$
شش 2

$$W_{2,3} = W_{2,2} + P_3 + q_3$$

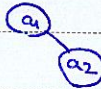
	0.30	0.03	0.45		
1		0.10	0.15		
2			0.05	0.15	
3				0.05	0.2
4					0.05
5					0.10

$$C_{0,2} = \min \left\{ \begin{array}{l} k=1 \Rightarrow C_{0,2}^1 + C_{0,2}^2 + W_{0,2} = 0.30 \\ 0 < k \leq 2 \\ k=2 \Rightarrow C_{0,2}^1 + C_{0,2}^2 + W_{0,2} = 0.35 \end{array} \right\}$$

$V_{\min}$

$$W_{0,2} = W_{0,1} + P_2 + q_2 = 0.30 + 0.15 + 0.45$$

$$a_{0,1}, a_{0,2} =$$



روش حریصانه Greedy

برای حل مسائل گزینش سازی زیر مسائل به صورت گزینش حل می شوند
شبه برنامه ریزی دینامیک است. بین چند انتخاب باید $\max$ و $\min$ را بگیریم و یکی از حریصانه یا تجربه و گزینش سازی کنیم
گزینه است یا نه. اگر انتخاب کنیم و جوابی بدیم

انتخاب حریصانه (greedy choice) ← سعی می کنیم گزینه بهترین را با راه حل گزینش سازیم

activity Selection

مسئله ای که در آن زمان برای هر فعالیت یک بازه مشخص است و در روش حریصانه هر فعالیت را می توانیم

Subject:

Year. Month. Date. ()

مسئله خودکشی اسکناس با سکه ها

ایضا حاکم اسکناس ها را با تعدادی تعداد سکه خود کنیم بهترین تعداد سکه معادل اسکناس را پیدا کنیم
اسکناس 28

سکه های 10 و 5 و 2 و 1

هر چه سکه بزرگتر و تعداد کمتری سکه خواهیم داشت

انتخاب حرفه ایانه: سکه های بزرگتر اول انتخاب شوند

اول سکه ای بزرگتر است 10 را برقی داریم در 28 حتما 10 است پس 8 تومان باقی می ماند بعد از 5 تومانی داریم
3 تومان 5 سکه با سکه جدیدی برای داریم که داریم

$$28 = 2 \times 10 + 1 \times 5 + 1 \times 2 + 1 \times 1$$

5 سکه

با تعداد سکه های کمتر می توان این اسکناس را خود کرد پس جواب را بکنیم است

اسکناس 15

سکه 10 و 5 و 2 و 1

$$15 = 1 \times 10 + 4 \times 1 = 5$$

5 سکه

انتخاب حرفه ایانه

$$15 = 3 \times 5 = 3$$

3 سکه

انتخاب بکنیم

پس بهترین است انتخاب حرفه ایانه بجز به انتخاب بکنیم نشود

توجه: نشان دهنده این سکه ها توان درستی از یک جتهادی با سکه

برای این مسئله انتخاب حرفه ایانه جواب بکنیم است

Subject:

Year. Month. Date. ()

مسئله زمان بندی کارها:

یک پردازنده داریم. تعدادی برنامه که قرار است در پردازنده اجرا شوند.

$\{T_i\}$ مجموعه کارها
 $\{t_i\}$ زمان پردازش

CPU

Task را به چه ترتیبی به CPU بدهیم که اجرا شوند تا زمان بایک $\min$ شود.
پیدا کردن ترتیب اجرا کارها که زمان پردازش به قدری کم شود که زمان بایک $\min$ شود.
زمانی که پردازنده دارد کارها را می پردازد و هیچ جوابی ندارد.

فاصله بین تحویل کارها در تمام آن = زمان بایک کار

T_1, T_2

مجموع زمان پردازش $= t_1 + t_2$

t_1, t_2

بایک $= t_1 + t_1 + t_2 = 2t_1 + t_2$

اگر T_1 اول اجرا شود به T_2 زمان بایک T_1 برابر است و t_1 و زمان بایک T_2 $t_1 + t_2$ چون باید کار اول تمام شود.

$T_1, T_2, \dots, T_n$

آیا این ترتیب اجرا است که زمان بایک $\min$ شود؟

T_1 زمان بایک t_1

T_2 " " $t_1 + t_2$

برای $\min$ شدن واریان ضریب دسترسی دارد باید از همه کمتر باشد.

T_n " " $t_1 + t_2 + \dots + t_n$

مجموع زمان بایک ها $\sum_{i=1}^n (n-i+1) \cdot t_i$

انتخاب حرفیه:

کارها بر اساس ترتیب صعودی زمان پردازش اجرا شوند.

یعنی که کمترین اول اجرا شود و بیشترین آخر.

می توانیم اینها را کنیم که کمتر از انتخاب هم است.

در ترتیب در صورتی داریم

هر دو یکی است

① ترتیب

$T_1, \dots, T_a, T_b, \dots, T_d$

② ترتیب

$T_1, \dots, T_b, T_a, \dots, T_d$

OS taskb

Subject:

Year. Month. Date. ()

این دو ترتیب عین هم هستند فقط T_a و T_b عوض شده. اگر نشان دهیم زمان پردازش ① کمتر از ② است می‌توانیم برای هر دو task دخواه نشان دهیم و تخمین دهیم بزرگ ترین صدوری است.

زمان بایگ‌ها را با هم جمع می‌کنیم

زمان بایگ T_a تا قبل از $T_a = \alpha$ و مجموع زمان بایگ‌های از ابتدا تا T_a که برای هر درختی است چون ترتیب آن کلی است.

T_1 مجموع زمان پردازش کارها از ابتدا تا T_a

مجموع زمان بایگ‌های آنها از ابتدا تا $T_b = \alpha$

T_1 مجموع زمان پردازش کارها از ابتدا تا T_b

T_2 مجموع زمان پردازش کارها بعد از T_a تا آخر
 { برای درختی T_2 باشد برای درختی هم T_2 خواهد بود
 $T_a = \dots$

β زمان بایگ کارها برای هر دو

$$\textcircled{1} \text{ مجموع زمان بایگ ترتیب } = \alpha + \beta + (T_1 + t_a) + (T_1 + t_a + t_b) = \alpha + \beta + 2T_1 + t_a + t_b + t_a$$

زمان بایگ T_b زمان بایگ T_a

$$\textcircled{2} \text{ مجموع زمان بایگ ترتیب } = \alpha + \beta + (T_1 + t_b) + (T_1 + t_b + t_a) = \alpha + \beta + 2T_1 + t_b + t_a + t_b$$

این عین هم هستند فقط درجه آخر تفاوت دارند

زمان بایگ ترتیب ② $\leftarrow$ زمان بایگ ترتیب ① $\Rightarrow$ چون $t_a \leq t_b$

activity Selection

مسئله انتخاب فعالیت ها:

$$\left\{ (s_i, f_i) \mid \text{تجویز فعالیت} \right\}$$

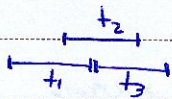
تجویز حالت

هدف انتخاب بهترین تعداد فعالیت که تضاد ندارند

محور زمان

انتخاب حرفه‌ایانه ①: فعالیت‌های کوتاه‌تر اول انتخاب شوند

مثال مشخص:



② انتخاب حرفه‌ایانه بهینه: انتخاب فعالیت‌ها که از هم زودتر تمام می‌شوند و تضاد ندارند

برای انتخاب دیگر آنگاه که به این تضاد دارند از دور خارج می‌شوند و به همین ترتیب (رادانه انتخاب می‌کنیم)

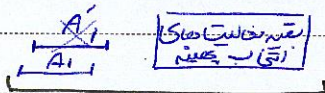
به انتخاب فعالیت‌ها که بهترین زمان خاتمه را دارند و تضاد ندارند به همین صورت

③ انتخاب فعالیت‌ها که بهترین تضاد را دارند

حالت نشان می‌دهد که در این انتخاب ما بهینه است، برای این کار:

تعیین ← با فرض اینکه فعالیت A_1 دارای بهترین زمان خاتمه است - فعالیت‌هایی که با A_1 بهینه از فعالیت‌ها وجود دارند که A_1 در این فعالیت آن باشد

برای اینکه با انتخاب بهینه از فعالیت‌ها داریم که A_1 در این فعالیت آن است:



انتخاب بهینه

A_1 را حذف می‌کنیم و A_2 را به آن اضافه می‌کنیم

حالت‌های A_1 و A_2

پس با هیچ تضاد نخواهد داشت. می‌توانیم اضافه کنیم و چون بهینه انتخاب‌ها هستند پس این

انتخاب هم بهینه است پس تعیین می‌کند فعالیت‌ها انتخاب می‌کنیم داریم که A_1 شروع می‌شود

انتخاب A_1 تجربه از دست دادن جواب‌های بهینه بهینه فعالیت‌ها خواهد بود

الموردینم هزینه انتخاب فعالیت ها :

1- صورتی که در آن فعالیت ها به واسطه زمان خاتمه به صورت تصوری

2- انتخاب فعالیت با زمان خاتمه کمینه

3- صرف فعالیت های مقدمات

4- ادغام در 2

به این طرز رفتار اداری به هم می پیوندد

نمونه فعالیت (مثال)

S_i E_i

1

1 4 ✓

بر اساس زمان ختم می شود

2

3 5 ①

start به کار می آید (است) را به هم می پیوندد

3

5 6 ①

min (است) انتخاب می کنیم

4

5 7 ✓

در هر مرحله فعالیت را به هم می پیوندد

5

3 8 ①

1 4 5 7 8 11 12 13

6

5 9 ②

7

6 10 ②

8

8 11 ✓

9

8 12 ③

10

2 13 ①

11

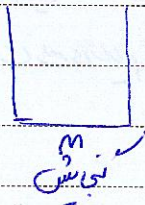
12 13 ✓

Subject:

Year. Month. Date. ()

یک نسخه از دایره مسطحی:

یک دایره مسطحی داریم که حواله‌های آن m است. یک تعداد می‌داریم که هر کدام دارای خواصیم و دایره مسطحی ترا داریم که این خواص باید در شکل حفظ شود و این خواص اصلی داخل دایره مسطحی max شود.



$\{v_i, w_i\}$ | مجموعه اشیاء
انرژی | وزن

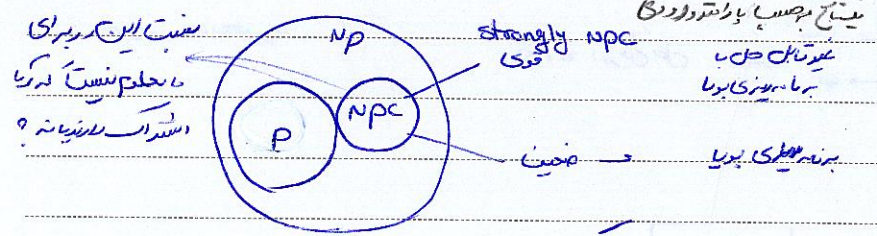
$$\sum_{i \in S} w_i \leq M$$

مجموع وزن اشیاء داخل کمد

$$\text{Max } \sum v_i$$

np-complete

بزرگترین فرموله حل می‌شود. یا حل می‌شود یا نداریم یا اشیاء می‌داریم. انتخاب کل می‌باشد یا هیچ. برنامه‌ریزی دینامیک (زمان و فضای کم).
اجازه داریم که کسری از یک می‌انتخاب کنیم. (fractional). کسری.
می‌تواند که بدون هزینه حل کرد. زمان و فضای کم. P - با چند جابجایی می‌توان اینها را حل کرد. P class.



چگونه است می‌تواند که مشکل سخت‌تر از این مسئله را در انتخاب کنیم. این باید مسئله باشد.

$$q_i = \frac{v_i}{w_i}$$

به انتخاب هزینه

انتخاب هزینه: انتخاب اشیاء بر حسب ترتیب نزولی $\frac{v_i}{w_i}$ تا اینکه دایره مسطحی پر شود.

برای نمونه Fractional

6	60	10	1
5	100	20	2
4	120	30	3

نمونه Fractional رابطی داریم

اگر چه ارزش بیشتر است را برای داریم و ...

2	100
4	60

M=30

مجموع ارزش ها = 160

3	75
4	60

M=25

مجموع ارزش ها = 135

برای نمونه 0-1

یضا هم بیشتر از آن درجه اول

20	100
3	60

M=50

ارزش کل = 160

جواب کار می باشد

جواب کنیم

3	120
2	100

= ارزش کل = 220

پس با همین روش می توان نمونه 0-1 را به سبب اصل کرد

Fractional :

توجه داشته باشید تعداد واحد در آن برای کنیم چون Fractional است می توانیم هر کدام را به تعداد در آن برابر و آن تقسیم کنیم اینده که را با واحد در آن ارزش بیشتری دارد برای کنیم به هم می آید انتخاب کنیم و است

Subject:

Year. Month. Date. ()

جوابی 1-2 جواب نمی دهد؟ یا چرا برای 1-2 جوابی جوابی دهد؟

نمی آید که M با وزن W باشد و لازم نیست برای W تقسیم کنیم جواب بقیه ای دهد.
به این علت که می توانیم بارها تقسیم کنیم قدری از فضای که در دسترس داریم را تقسیم کرده و در یک بخش است این
قدری در دسترس است که در این انتخاب های قبلی در دسترس هم نیستند پس برای 1-2 جواب
می دهد.

مساله زمان بندی کارها با جریمه تأخیر

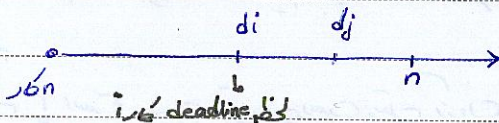
اگر از Deadline بگذرد جریمه می شود و مجموع جریمه ها $\min$ شود.

نشان بدهیم
 $\{T_i, t_i, d_i, w_i\}$ مجموعه کارها
Deadline

و $t_i = 1$ فرض

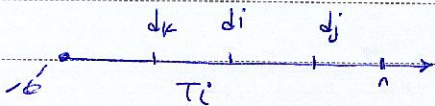
w_i

هر کارها باید اندر زمان نیاز دارند
میزان جریمه به نسبت زمان که از Deadline می گذرد



زمان بندی کارها در n و زمان انجام می شود

دقیق کاری بخش جریمه تعلق می گیرد کمتر است. آخر تر از هر کار که جریمه تعلق می گیرد هم این کار است.
Deadline آن بلند بر میزان جریمه تأخیر می گذرد. آخر تر از هر کار که جریمه تعلق می گیرد هم این کار است.
بلند جری می باشد قبل از Deadline می توانیم از این کارها



کمتر است کار را به زمان قبل از Deadline انجام می دهیم تا برای کارهای که Deadline آنجا است
وقت داشته باشیم

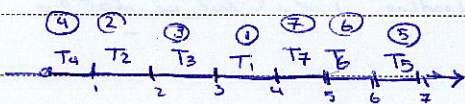
منطق است که کارها را به ترتیب نزولی جریمه ها انجام دهیم.

1. کارها را به ترتیب نزدیکی جریه ها بر روی میز می نشاند
 2. هر کاره حقیقتاً قبل از آنکه برسد زمان بندی می شود
 3. اگر هیچ کاره ای در صف انتظار نباشد یعنی به وقت بعد از آنکه کارها تمام شده و در اولین جای خالی زمان بندی می شود
 4. اگر هیچ کاره ای در صف انتظار نباشد آن کاره را در اولویت قرار می دهیم و به ترتیب کارها را انجام می دهیم و به ترتیب زمان بندی می شود.
- سوال: اگر هر کاره deadline داشته باشد چطور می توانیم آن را در اولویت قرار دهیم؟

پاسخ: اگر هر کاره deadline داشته باشد به صورت کلی می توانیم به ترتیب از deadline

(مثال)

کارها	1	2	3	4	5	6	7
w_i	70	60	50	40	30	20	10
d_i	4	2	4	3	1	4	6



اول کارها را به ترتیب از نزدیکترین زمان بندی می کنیم. 1 - قبل از 4
برای 7 چون از 4 به است یعنی عقب تر است و قبل از 4 نمی نشاندیم

برای 5، 6، 7، 2، 3، 4، 1، 7، 6، 5 است چون از 4 به است و قبل از 4 نمی نشاندیم. 5، 6، 7، 2، 3، 4، 1، 7، 6، 5
جریه = 30 + 20 = 50

پس $O(n \log n)$

$$O(n^2 + n \log n) = O(n^2)$$

Subject:

Year. Month. Date. ()

تعداد بیت مورد نیاز برای ذخیره کلمات

$$\min \sum f(c_i) \times p(i)$$

تعداد بیتی که در متن تکرار شده

مسئله دنداری بجهینه:

ی ضرایب این مجموع را min کنیم

در ASCII همیشه زبانی که در برابر 8 ی باشد (ادی که زبانش بیشتر دارد) کوتاه تر است (آن اختصاصی خارجی)

الگوریتم هانچن:

برای این که حرفی همیشه در ی ضرایب

برای این که تکرار کنیم

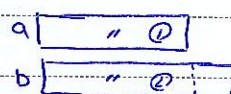
جایگاه های این حرف به حسب ی درجیم و چپ

به سمت راست!

که حرف ی شود به حسب یال ها از مسیر به تکرار

۱۰۰: ۱۰۰

که حرف ی به حسب یال ها مسیر به تکرار (بزرگ)



①, ③, ⑤ به هم برابرند

حرفی که طولش کمینه است باید بیشتر در حرفی که طولش

بیشتر است

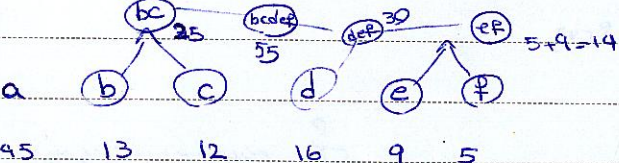
درست این خاصیت: خاصیت بهشوندی

این که با طول تغییر نیابد خاصیت بهشوندی داشته باشد

این که خاصیت چون حرف در هر کجاست پس خاصیت بهشوندی نداریم

طول کم = طول مسیر از ریشه تا برگ پس هر چه به ریشه نزدیکتر باشد که کوتاه تر است خواهد داشت

(شکل)

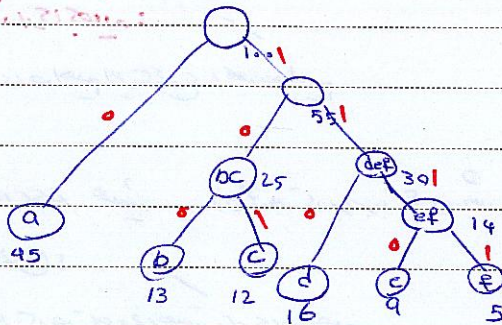


اول دقت کنیم که تکرار حرفش را طوری که merge ی شود حرف جدید با طولش مجموع آنها به صورت

recessive اول در تکرار به هم min هستند به هم merge ی کنیم

Subject:

Year. Month. Date. ()



a : 1000

d : 110

b : 100

e : 1110

c : 101

f : 1111

طول کد برای a = $1000 \times 3 = 300$ بیت

طول کد برای b = $45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4 = 224$ بیت

الگوریتم:

Huffman (C)

{
 $n \leftarrow |S|$

$Q \leftarrow C$ (مجموعه اولویت)

 for $i = 1$ to $n-1$ do

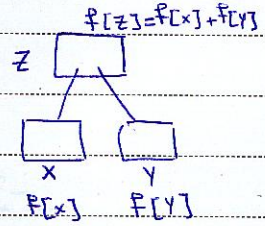
$Z \leftarrow \text{allocate_node}()$

$X \leftarrow \text{left}[Z] \leftarrow \text{Extractmin}(Q)$

$Y \leftarrow \text{right}[Z] \leftarrow \text{Extractmin}(Q)$

$f[Z] \leftarrow f[X] + f[Y]$

 Insert(Q, Z)



در هر مرحله از ساخت درخت

Z را انتخاب می‌کنیم

Subject:

Year. Month. Date. ()

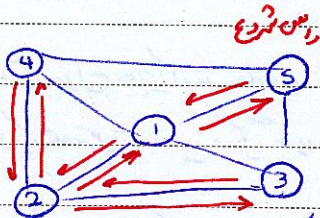
الگوریتم های گراف:

الگوریتم های گراف: DFS و BFS

(Depth First Search) DFS

(Breadth First Search) BFS

: DFS



الگوریتم DFS: در این الگوریتم تمام گره ها را visit می کنند. اگر گره node را visit می کنند و تمام گره های مجاور آن گره را visit می کنند. اگر گره های مجاور آن گره را visit می کنند و تمام گره های مجاور آن گره را visit می کنند. اگر گره های مجاور آن گره را visit می کنند و تمام گره های مجاور آن گره را visit می کنند.

DFS(w)

برای گراف هم بند

Begin

mark[v] = visited

node v

for each w in L(v) do

if mark[w] = unvisited then

DFS(w)

end

Depth First Search

for v = 1 to n do

if mark[v] = unvisited then

DFS(v)

end

در edge گره های مجاور برای اینکه ببینیم آیا گره های مجاور آن گره را visit می کنند یا نه. اگر گره های مجاور آن گره را visit می کنند و تمام گره های مجاور آن گره را visit می کنند.



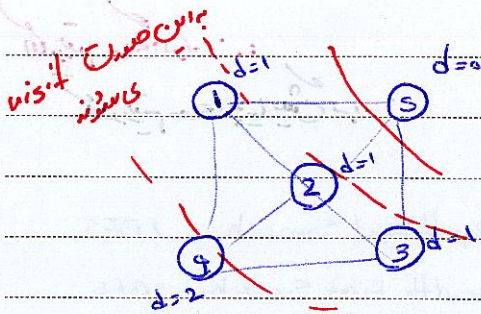
DFS زمان: $O(|V| + |E|)$

PAPCO

Subject:

Year. Month. Date. ()

BFS:



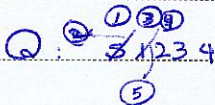
منظور از این توضیح اینست که
ابتدا در node شروع هستیم در مرحله دوم تمام همسایه‌های
current node را visit می‌کنیم

الگوریتم $d=i$ به شدت محدود می‌کند که تا node start فاصله دارد. این است در واقع ضمیمه کوتاه‌ترین مسیر از ریشه
نتیجه این است که تا به ریشه (کوتاه‌ترین مسیر) نرسیم هیچ کاری نمی‌کنیم (در واقع به عقب برمی‌گردیم)

هر node را به یک بزرگی می‌کنیم و همسایه‌های آن را هم به یک بزرگی می‌کنیم و هر node را به یک بزرگی می‌کنیم و به یک بزرگی می‌کنیم
چک می‌کنیم

$$O(|V| + |E|)$$

Queue در هر مرحله node را در صف می‌گذارد و node بعدی را در صف می‌گذارد
نتیجه این است که تا به ریشه (کوتاه‌ترین مسیر) نرسیم هیچ کاری نمی‌کنیم (در واقع به عقب برمی‌گردیم)



Subject:

Year. Month. Date. ()

(Shortest Path)

الگوریتم های کوتاه ترین مسیر

$G(V, E)$ و u, v

$$\min \sum w(e_i)$$

$$e_i \in \text{path}(u, v)$$

دлина кратчайшего пути
← min

(خوارزمية)

Dijkstra

(برنامه سازی پویا)

Floyd-warshall

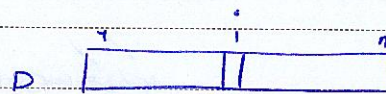
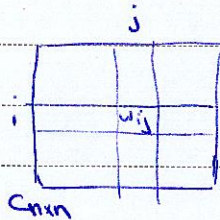
الگوریتم

Dijkstra

الگوریتم

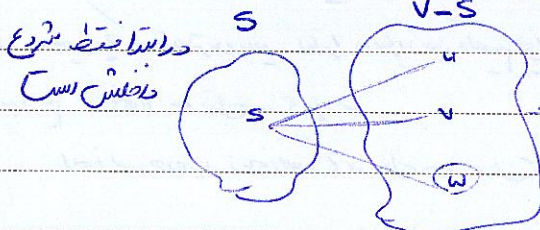
فرض کنید ما می خواهیم در یک گراف به صورت یکتا به یک نقطه برسیم.

Single source یک node مبدأ دارد (source) و مسیرهای به مقصد



یک آرایه یک بعدی در نظر می گیریم

این الگوریتم D را بر روی یک گراف به صورت یکتا اجرا می کند



از اینجا به بعد $\min$ است پس w را از s به w می بریم و به s منتقل می کنیم

نکته: اگر D را از روی C می گیریم، می توانیم به روش دیگر به جواب برسیم

$$D[i] = C[i, a] \quad \text{و} \quad \text{من می توانم به این راه به وصل می کنم}$$

این یک الگوریتم است که در یک بار استفاده می کنیم (سیستم) یا جواب دارد یا ندارد (نه)

در هر بار از اینجا کمتر می شود، زیرا به عنوان جواب انتخاب می کنیم

$$\left. \begin{array}{l} \text{پس این رو} \\ \text{min می گیریم} \end{array} \right\} \begin{array}{l} D[v] \\ D[w] + C[w, v] \end{array}$$

Subject:

Year. Month. Date. ()

Dijkstra

begin

$S = \{1\};$

for $i=2$ to n do

$D[i] = C[1, i];$

for $i=1$ to $n-1$ do $O(n)$

begin

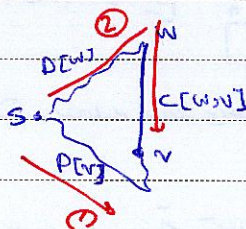
choose a vertex w in $V-S$

such that $D[w]$ is minimum

$S = S \cup \{w\}$

for each vertex v in $V-S$ do $O(n)$

$D[v] = \min \{D[v], D[w] + C[w, v]\}$



سین یاد دے با ترجمہ یہ مثال پال جائے گا کہ

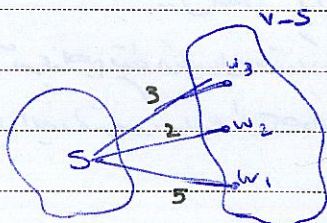
min ہونے پر مثال میں اصل میں کتاب کی سیر

از مسیر کے لئے کہ ہر بار یہ لہذا اس کے لئے کہ ہر بار یہ

ہر بار یہ node کے لئے کہ ہر بار یہ

انہی کے لئے کہ ہر بار یہ node کے لئے کہ ہر بار یہ

در هر مرحله (با راههای کمترین) می توانیم دیتا تر مسیر پیدا کنیم که با ترجمہ node های کمترین مسیر که با ترجمہ



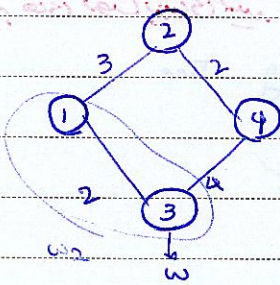
دیا کہ ہر بار یہ (فصلہ) از کی دارد و چون مسیر دیتی

پیدا می کنیم کہ در هر مرحله کمترین مسیر باشد چون با ترجمہ

پال های کمترین را هم به آن اضافه کنیم

Subject: _____

Year. _____ Month. _____ Date. ()



① Step $\rightarrow$ D

	1	2	3	4
	-	3	2	∞

② Step $\rightarrow$

		3	-	6

w

③ Step $\rightarrow$

		-	-	5

این مسیر که پیدا کردیم از قبل کوتاهتر بوده ← update کنیم

All pairs shortest paths

Floyd-warshall الگوریتم

Floyd-warshall

begin

for $i=1$ to n do

$A[i,j] = C[i,j]$

for $k=1$ to n do

for $i=1$ to n do

for $j=1$ to n do

if $A[i,k] + A[k,j] < A[i,j]$ then

$A[i,j] = A[i,k] + A[k,j]$

end

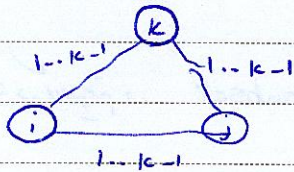
Complexity $\rightarrow$

$A[i,j] \rightarrow A[i,j]$

طول مسیرهای کوتاه

(update می شود) کوتاه تر

$\Rightarrow \Theta(n^3)$



Subject:

Year. Month. Date. ()

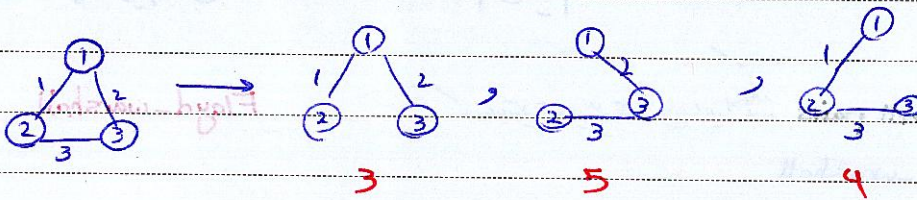
الگوریتم های دخت پوشای کمینه

(MST) minimum spanning Tree

$G(V, E)$

رشته گراف ← دخت

پوشش
رشته گراف پوشش
رشته گراف پوشش



دخت پوشای کمینه

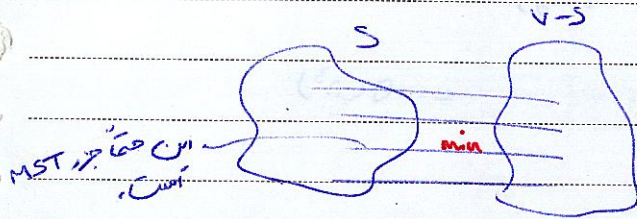
به عنوان مثال

prime

kruskal

هر دو روش به دست می آید و انتخاب greedy آن بر اساس یک تعریف است.

تعریف: اگر $G(V, E)$ را به دو قسمت S و $V-S$ تقسیم کنیم به دو بخش S و $V-S$ می آید. MST است.



این هم به دست می آید و انتخاب یک به یک

تفاوت هم این دو الگوریتم در subset است

الگوریتم Prime

یک سر و خواره را انتخاب می کنیم و در مجموع S قرار می دهیم
 سپس S و $V-S$ آن را یکی به دیگری پیوند می دهیم اگر دو وکتور متناظر
 آن را در $V-S$ یعنی W را به S اضافه می کنیم و این کار را
 تا زمانی که S تمام V را پوشش دهد (این کار را می توان به تدریج کرد)

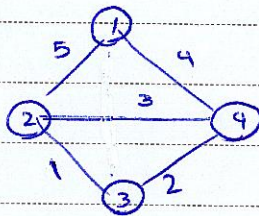


پس از آنکه انتخاب می شود پل جدا می کنند MST هستند و اگر $n-1$ بار این کار را تکرار کنیم پل جدا می
 پیدا می شود و در نهایت $n-1$ پل می شود.

Kruskal

الگوریتم

پس از آن که تمام پل ها را بر اساس وزن آن ها به صورتی مرتب می کنیم و به همین ترتیب پل ها را به
 MST اضافه می کنیم. اگر اضافه کردن یک پل یک حلقه ایجاد کند آن پل اضافه نمی شود.
 پل min حلقه MST است یک پل $(node)$ را S می کنیم و بقیه $node$ ها را $V-S$ و در نظر
 می گیریم



پس از آنکه 3 را اضافه می کنیم اگر 3 را اضافه
 کنیم حلقه می شود پس اضافه نمی کنیم 4 را انتخاب می کنیم
 پل 1 و 2 را اضافه می کنیم و 5 را اضافه نمی کنیم زیرا اگر 5 را
 اضافه کنیم حلقه تشکیل می شود و در نتیجه اضافه نمی کنیم