Subject
Year Month Date

سرفصل های طراحی و تحلیل الگوریتم

1. مقدمه ای بر تحلیل الگوریتم
2. استقرای ریاضی
3. روش های تقسیم و غلبه
4. روش های حریصانه
5. برنامه نویسی پویا
6. روش های جست و جو و پیمایش
7. روش های پشته
8. روش های انشعاب و تحدید

فصل اول / مقدمه ای بر تحلیل الگوریتم ها:

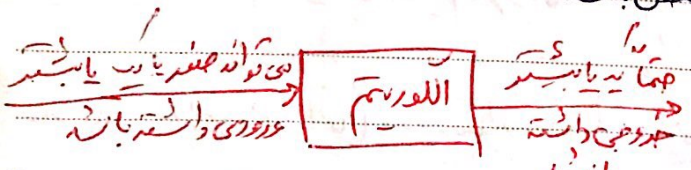
الگوریتم: از ریشه ی الخوارزمی ریاضی دان و منجم قرن ۲ هجری

معنی لغوی: روش حل مسئله

در علم کامپیوتر:

۱. تعریف اول: هر دستور العملی که مراحل مختلف انجام کاری را به زبان دقیق و با جزئیات کافی بیان کند به نوعی که

ترتیب مراحل و شروع و خاتمه عملیات در آن کاملاً مشخص باشد



۲. تعریف دوم: الگوریتم مجموعه ایست متناهی از مراحل که هر مرحله ممکن است مستلزم یک یا چندین عمل باشد

هر عمل دارای محدودیت هایی به صورت زیر است:

Scanned by CamScanner

Subject:
Year. Month. Date.

۵ نمره میان ترم و حضور در کلاس
۱۵ نمره پایان ترم

(۵×۴) or (۷+۹)

۱/ مبهم نباشد یعنی چه؟

۲/ باید قابل اجرا باشد یعنی چه: حساب روی اعداد حقیقی X

۳/ شرط خاتمه داشته باشد یعنی چه: شمردن اعداد اول X

الگوریتم
- مستقیم: در داخل خودش، خودش را صدا می‌زند یعنی چه: فاکتوریل، فیبوناچی، برج هانوی — مثال: fact 5! = fact 4! fact 3!
- غیرمستقیم: دو برنامه A و B؛ A، B را فراخوانی می‌کند و B، A را فراخوانی می‌کند
  - بازگشتی
  - غیربازگشتی

روش‌های مطالعه الگوریتم‌ها:

۱/ اجزای الگوریتم:
۱- استقرای ریاضی
۲- تقسیم و غلبه
۳- روش برنامه‌نویسی پویا
۴- روش پلرد
۵- انشعاب و تحدید
۶- حریصانه

۲/ مقدمه‌سازی: اثبات درستی الگوریتم با ورودی‌های مناسب خروجی درست بدهد.

۳/* تحلیل الگوریتم (تحلیل مقدم، ارزیابی کارآیی): به الگوریتم در زمان اجرا از cpu کامپیوتر برای اجرای عملیات و از حافظه‌ی آن (اصلی یا جانبی) برای ذخیره برنامه و داده‌ها استفاده می‌کند.

تعریف تحلیل الگوریتم: تحلیل یک الگوریتم به فرآیندی اطلاق می‌شود که تعیین می‌کند الگوریتم به چه مدت زمان از cpu برای انجام محاسبات (پیچیدگی زمانی) و به چه مقدار حافظه (پیچیدگی مکانی) برای ذخیره‌سازی داده‌ها و برنامه نیاز دارد

پیچیدگی زمانی: مرتبه‌ی زمانی، درجه‌ی بزرگی یک الگوریتم هم می‌گویند
پیچیدگی مکانی: مرتبه‌ی مکانی

۴/ پیاده‌سازی: پیاده‌سازی الگوریتم با یکی از زبان‌های برنامه‌نویسی متداول

Scanned by CamScanner

۵/ تست برنامه:

۱ اشکال زدایی (Debug): اجرای برنامه روی داده‌های نمونه برای بررسی و یافتن خطاها و برطرف کردن آنها

۲ تحلیل مؤثر (profiling): اندازه‌گیری کارایی

فرآیند اجرای برنامه درست، بر روی مجموعه داده‌های نمونه جهت تعیین زمان و حافظه‌ی لازم برای محاسبه‌ی نتیجه است

تحلیل الگوریتم (پیچیدگی زمانی): فرض کنید یک کامپیوتر با یک RAM داریم که در آن:

الف) دستورالعمل‌هایی بعد از دیگری اجرا می‌شوند بدون هیچ پیش‌آمد غیرمنتظره‌ای

ب) ذخیره و بازیابی هر عنصر در یک مدت زمان ثابت انجام می‌شود

آن چه برای تحلیل یک الگوریتم لازم است، شامل:

۱) تعیین عملیات مورد استفاده در الگوریتم مانند چهار عمل اصلی دستورات مقایسه‌ای خواندن و نوشتن فراخوانی‌ها،

دستور انتساب و غیره و مشخص کردن زمان اجرای آن‌ها.

```
for i=1 to n        → n+1 بار
    if A[i] > 0     → n بار
        A[i] = A[i+1]   → اگر شرط برقرار باشد n بار
    else
        return -1       → 1 بار
```

۲) تعیین تعداد کافی از مجموعه‌ی داده‌ها برای تعیین تمامی حالات ممکن یک الگوریتم در بهترین حالت، بدترین حالت و حالت

Subject :

Year . Month . Date . ( )

میانگین یا متوسط.

مرتبه‌ی زمانی (order):

زمان اجرای الگوریتم: { تحلیل مؤخر: ۱) سخت افزار ۲) زبان برنامه‌نویسی ۳) کامپایلر
تحلیل مقدم: ۴) نوع دستورها و زمان اجرای آن‌ها و تعداد آنها ۵) به پارامتری مانند n (تعداد ورودی‌ها و خروجی‌ها و یا هردو)

زمان اجرای الگوریتم با n ورودی $tp(n) = \sum_{\text{دستورالعمل‌ها}} (\text{زمان اجرا}) * (\text{تعداد دفعات آن})$

(تحلیل مقدم) ← بستگی به سخت افزار

فرض: زمان اجرای همه‌ی دستورالعمل‌هایمان و برابر با یک واحد cpu است.

$tp(n) = \sum_{\text{دستورالعمل‌ها}}$ تعداد دفعات اجرا

مثال: قطعه برنامه‌ی زیر را در نظر بگیرید و تعداد دفعات اجرای دستور $x = x+y$ را محاسبه کنید

الف) $x = x+y$ ← یک بار

ب) for i:=1 to n do $x = x+y$ ← $n+1$ بار
~~$x = x+y$~~ ← n بار

ج) for i:=1 to n do → $n+1$ بار
for j:=1 to n do → $n(n+1)$
$x = x+y$ → n^2 بار

P4PCO

Scanned by CamScanner

Year Month Date (۲)

نکته: به ۱ و n و n^2 درجات بزرگی دستور العمل $x = n + y$ می گویند

* نکته ۱: زمان اجرای یک الگوریتم به چند جمله ای است برحسب n مانند $an^2 + bn + c$ (a, b, c ثوابت هستند)

* نکته ۲: در تحلیل مقدم، هدف مشخص کردن نوع منحنی زمانی است

تمرین: نوع منحنی زمانی را مشخص کنید.

```
for i=1 to n do  → n+1
{
    j = i          → n
    while j > 0 do
    j = [j/2]      → n/2 * log n
}
```

جواب: $\Theta = n \log n$

نمادهای مجانبی O ، Ω ، Θ ، o ، ω

حالت متوسط ← بدترین حالت ← بهترین حالت

اجرای الگوریتم

نوع منحنی زمانی: $an + b$

مرتبه ی اجرائی: $O(n)$ زمان اجرای خطی است

مانند:

```
z = x + y   → 1
for i=1 to n do  → n+1
{
    y = z   → n
    z = y + x  → n
}
```

$1 + 3n + 2$

Scanned by CamScanner

Subject:

Year Month Date

تعریف O (بدترین حالت): $f(n)=O(g(n)) \iff \exists c_1>0 \; \forall n \geq n_0 : f(n) \leq c_1 g(n)$

عدد حقیقی — اعداد طبیعی — طبق تعریف حذف می‌شود

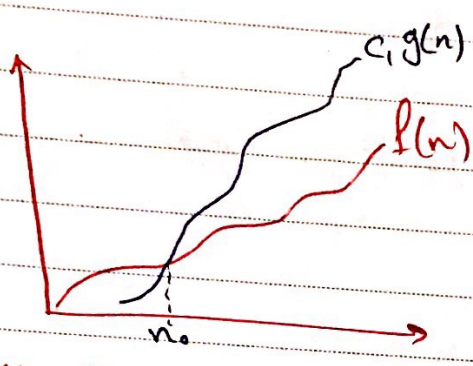
$f(n) \leq c_1 g(n)$

معرف توابع زمان

ماشه: $\lim_{n\to\infty} 2n^2+8n-3 \Rightarrow 2n^2$ در ریاضی

در واقعی الگوریتم $Vn^2+8n-3 \Rightarrow O(n^2)$

تعبیر هندسی O:



Vn^2+4n+9

مثال: فرض کنید مرتبه‌ی اجرای الگوریتمی به صورت زیر است:

مقدار ؟ = O

$O(n^2)$

$c_1 \leq V+\varepsilon$ (اپسیلون دلخواه)

$\Rightarrow c_1 \leq 8$

طبق تعریف $\Rightarrow Vn^2+4n+9 \leq 8n^2 < 8n^3 < 8n^4 < \dots$

$O(n^2)$ $O(n^4)$

$n_0 = 8$

مثال: $2n^2+2n^2\log n+Vn+4$

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

$c_1 = 2+\varepsilon = 3$

با جایی که داری اعداد بالا در تعریف O می‌توانیم n_0 را بدست آوریم

قضیه: اگر $A(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$ و $am > 0$

در این صورت $A(n) = O(n^m)$

P4PCO

Scanned by CamScanner

Subject:

Year: Month: Date: (f)

نتیجه‌ی ۱: اگر برنامه دارای k دستور العمل با درجات بزرگی (تعداد تکرار) $O(n^{m_1}), O(n^{m_2}), \dots, O(n^{m_k})$ باشد

آنگاه درجه‌ی بزرگی برنامه برابر $O(n^m)$ است که در آن $m = \text{Maximum}\{m_1, m_2, \dots, m_k\}$

نتیجه‌ی ۲ (قانون جمع): اگر برنامه‌ی P از دو قطعه برنامه‌ی P_1 و P_2 به ترتیب با زمان‌های اجرای $T_1(n)$ و $T_2(n)$ تشکیل شده باشد که در آن $T_1(n) = O(f(n))$ و $T_2(n) = O(g(n))$ و در این صورت زمان اجرای برنامه‌ی P که آن را با $T(n)$ نمایش می‌دهیم برابر خواهد بود با: $T(n) = O(\text{maximum}(f(n), g(n)))$

نتیجه‌ی ۳ (قانون حاصل ضرب): اگر $T_1(n) = O(f(n))$ و $T_2(n) = O(g(n))$ آنگاه $T_1(n) \cdot T_2(n) = O(f(n) \cdot g(n))$

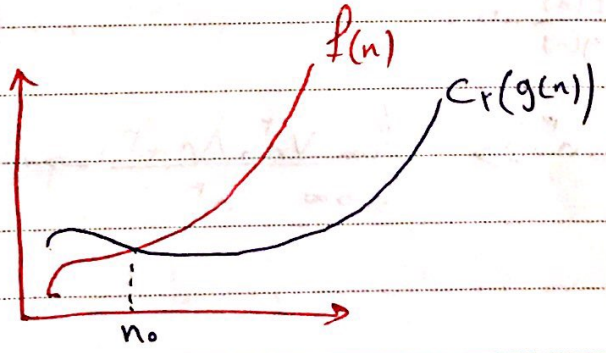
مثلا $T_1 = O(n^2)$ اگر $T_2 = O(\log n)$ باشد آنگاه: $T_1 \cdot T_2 = O(n^2 \log n)$

تعریف Ω (بهترین حالت اجرای الگوریتم):

$$f(n) = \Omega(g(n)) \iff \exists c_2 > 0 \ \forall n \geq n_0 \quad f(n) \geq c_2 g(n)$$

(c_2: عدد حقیقی، n_0: عدد طبیعی)

تعبیر هندسی Ω:



مثال: $\Omega = ?$ $\quad 7n^2 + 6n + 9$

$c_2 = 7 - 6 = 1$

$7n^2 + 6n + 9 \geq 6n^2 \geq 6n \geq 6\log n \geq 6$

$\Omega(n) \geq \Omega(\log n) \geq \Omega(1)$

نکته: Ω عدد ثابت ۱ است

PAPCO $n_0 = 1$

Scanned by CamScanner

Subject:

Year. Month. Date.

تعریف Θ (حالت متوسط): بهترین و دقیق‌ترین معیار

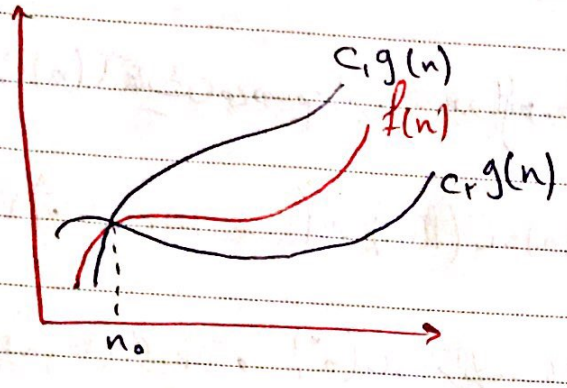
$$f(n)=\Theta(g(n)) \iff \exists c_1,c_2>0 \;\; \forall n\geq n_0 \;\; c_2 g(n)\leq f(n)\leq c_1 g(n)$$

(Ω: $c_2 g(n)\leq f(n)$, O: $f(n)\leq c_1 g(n)$)

قضیه: $f(n)=O(g(n))$ و $f(n)=\Omega(g(n))$ $\Rightarrow f(n)=\Theta(g(n))$

$f(n)=O(g(n))$ و $f(n)=\Omega(g(n))$ $\Rightarrow f(n)=\Theta(g(n))$

تعبیر هندسی Θ:



مثال: ثابت کنید $\Theta(n^3)\ni \Lambda n^3+V n^2\log n+9n+V$ است

$O(n^3)$ و $\Omega(n^3)$ $\Rightarrow \Theta(n^3)$

تعریف o (small o):

$$f(n)=o(g(n)) \iff \lim_{n\to\infty}\frac{f(n)}{g(n)}\to 0$$

مثال: $V n^2+\Lambda n+3$ $\quad g(n)=n^3 \Rightarrow \lim_{n\to\infty}\frac{V n^2+\Lambda n+3}{n^3}\to 0$

$\log n$ $\quad g n=n \Rightarrow \lim_{n\to\infty}\frac{\log n}{n}\to 0$

Scanned by CamScanner

تعریف ω (امگا کوچک): $f(n) = \omega(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} \to \infty$

سعی کنید حتی‌الامکان انتخاب می‌کنیم

مثال:

$\sqrt{n^2+8n+3}$

$g(n) = n$

$\omega(n)$

$\lim_{n\to\infty} \frac{\sqrt{n^2+8n+3}}{n} \to \infty$

$\log n$

$\omega(1)$

$\lim_{n\to\infty} \frac{\log n}{1} \to \infty$

$\lim_{n\to\infty} \frac{\log n}{\log\log n} \to \infty$

شباهت بین نمادهای O، o، Θ، ω و Ω در مقایسه بین زمان‌های اجرای الگوریتم‌ها و نمادهای $<, \leq, =, \geq, >$ در مقایسه بین اعداد طبیعی:

$a \leq b \approx f(n) = O(g(n))$

$a \geq b \approx f(n) = \Omega(g(n))$

$a = b \approx f(n) = \Theta(g(n))$

$a < b \approx f(n) = o(g(n))$

$a > b \approx f(n) = \omega(g(n))$

تحلیل الگوریتم مرتب‌سازی حبابی (Bubble sort): $A[1..n]$

هدف: مرتب‌سازی صعودی آرایه A

```
Algorithm Bubble sort(A,n)
for i=1 to n-1 @                    n بار
for j=n down to i+1 do (b)          n-1 بار
if A[j] < A[j-1] then (c)           n-1 بار
{
```

i=1 : از n تا 2	i=2 : از n تا 3
n-1 بار	n-2 بار
n-1 بار	n-2 بار
...	...

در بهترین حالت هیچ عضوی عوض نمی‌شود | بقیه‌ی حالت‌ها نیز عضوی عوض می‌شود

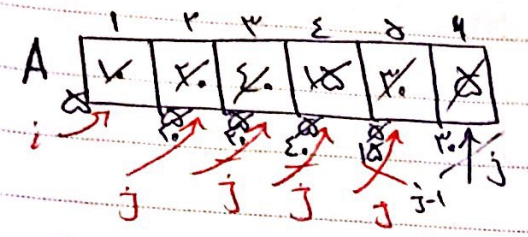
Scanned by CamScanner

Subject:

Year. Month. Date. ( )

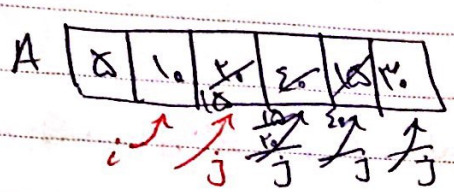
```
temp = A[j-1]   (d)   n-1 بار
A[j-1] = A[j]   (e)   n-1 بار   برای جابجایی بین خانه‌ها استفاده می‌شود
A[j] = temp     (f)   n-1 بار
```

Swap A[j] A[j-1]

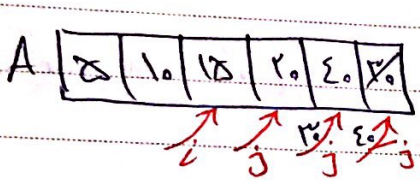


iteration 1

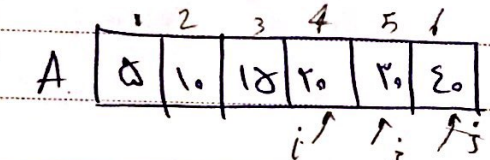
توضیح جدول با یک مثال:



iteration 2



iteration 3

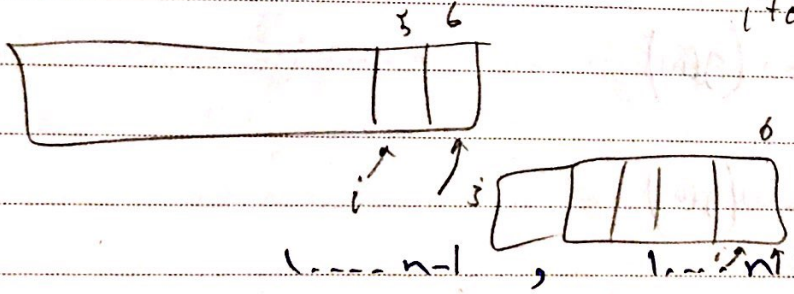


iteration 4

i = 4

iteration 5

i = 5



iteration 6

آرایه صعودی

یا نزولی: بهترین حالت

$1 \dots n-1$, $1 \dots n$

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

Scanned by CamScanner

تحلیل زمانی:

بدترین حالت (O): آرایه به صورت نزولی مرتب شده باشد

	تعداد دفعات اجرا
i ی ۱	1+۵(n-1)
i ی ۲	1+۵(n-۲)
⋮	⋮
i ی n-1	1+۵(1)
	n-1+۵(n(n-1)/۲) تعداد دفعات اجرا

مرتبه‌ی زمانی $O(n^2)$

بهترین حالت Ω: آرایه به صورت صعودی باشد

	تعداد دفعات اجرای
i ی ۱	1+۲(n-1)
i ی ۲	1+۲(n-۲)
⋮	⋮
i ی n-1	1+۲(1)
	1+۲[(n-1)n/۲] => $\Omega(n^2)$

$\Theta(n^2)$

حالت متوسط:

الگوریتم مرتب سازی درج (Insertion Sort):

فرض: آرایه A[1 ... n]

هدف: مرتب سازی

Scanned by CamScanner

Subject :

Year . Month . Date . ( )

Algorithm Insertion sort (A,n)

A[0] = -∞

for j = 2 to n do

{

item = A[j]; i = j-1

(*) while item < A[i] do

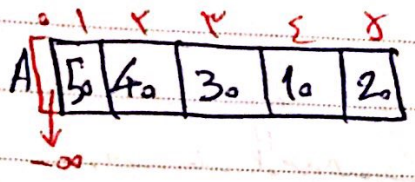
{

A[i+1] = A[i]

i = i-1

}

A[j+1] = item

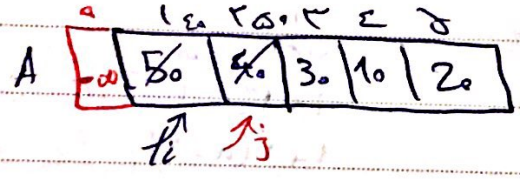


توضيح الدرس من خلال مثال:

interation 1

i = 2

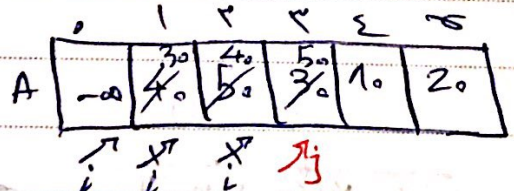
item = 40



interation 2

j = 3

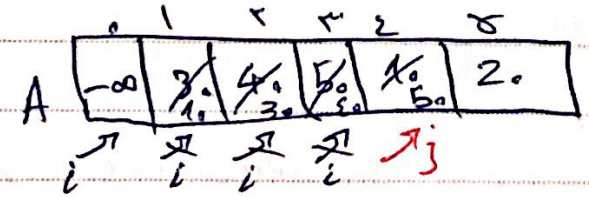
item = 30



interation 3

j = 4

item = 10



Scanned by CamScanner

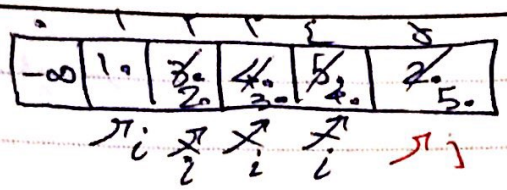
Subject:

Year Month Date

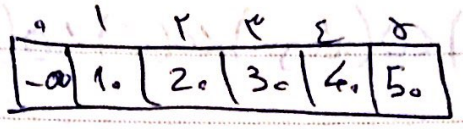
intration 4

j = 5

items = 2.



intration 5



بهترین حالت O: آرایه به صورت نزولی مرتب شده باشد

j	دفعات اجرای (*)
۲	۲
۳	۳
⋮	⋮
n	n بار

$$-1+1+2+3+\cdots+n$$

$$\frac{n(n+1)}{2}-1 \Rightarrow O(n^2)$$

بهترین حالت Ω: آرایه به صورت صعودی مرتب شده باشد

j	تعداد دفعات اجرای (*)
۲	۱
۳	۱
⋮	⋮
n	1

n-1 بار $\Rightarrow \Omega(n)$

Scanned by CamScanner

تمرین: Θ را بدست آورید با ذکر منبع و جلسه.

جهات بزرگی متداول برای الگوریتم‌ها:

$$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^2 \log n) < \dots < O(n^n)$$

مرتبه‌ی اجرایی: الگوریتم‌های بازگشتی:

تعداد فراخوانی‌های تابع: مرتبه‌ی اجرایی

برای حل الگوریتم بازگشتی:
1. جایگذاری با تکرار
2. معادله‌ی مشخصه
3. سری‌ها و ...

مثال: مرتبه‌ی اجرایی تابع بازگشتی فاکتوریل را بدست آورید.

```
int fact(int n)
{
if (n==1) return 1;
return (n*fact(n-1));
}
```

fact(4)
4 * fact(3)
3 * fact(2)
2 * fact(1)

ترسیم درخت بازگشتی:

$n=4$

فراخوانی‌ها: 4 بار

در کل تعداد فراخوانی‌ها: $n \Rightarrow \Theta(n)$ بار

جایگذاری با تکرار:

$n=1 \Rightarrow T(1)$ return if برای

n ام: $T(n-1)+c$

$n-1$: $T(n-1)$

fact(n-1) = T(n-1)

fact: n * fact(n-1)

$c+\dots$

Scanned by CamScanner

Subject:

Year. Month. Date.

رابطه‌ی بازگشتی

$$\begin{cases} T(n) = T(n-1) + c \\ T(1) = 1 \end{cases}$$

$T(1) = 1$

$T(2) = T(1) + c = 1 + c$

$T(3) = T(2) + c = 1 + 2c$

$T(4) = T(3) + c = 1 + 3c$

$\vdots$

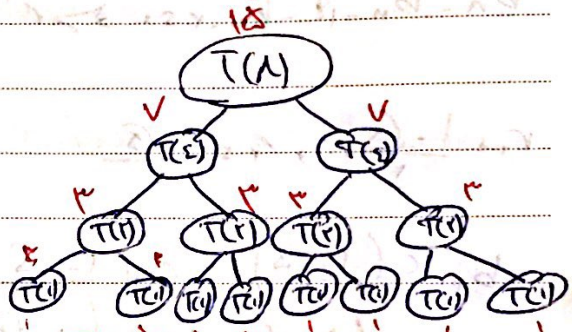
$T(n) = \underbrace{1 + (n-1)c}_{1 + cn - c} \Rightarrow T(n) = O(n)$

مثال: مرتبه‌ی اجرایی تابع زیر را بیابید.

```
int T(int n){
if(n<=1)return 1;
else return T(n/2)+T(n/2)
}
```

$n = 2^m$

$n = 8$



به این دلیل ۲ شده زیرا $T(2)$ خودش را ۲ بار فراخوانی می‌کند

تعداد فراخوانی‌ها:

$T(1) = 1$

$T(2) = 3$

$T(4) = 7$

$T(8) = 15$

$T(16) = 31$

$T(32) = 63$

$2n - 1$

$T(n) = 2n - 1 \Rightarrow O(n)$

PAPCO

Scanned by CamScanner

Subject:

Year Month Date

مثال: مرتبه‌ی زمانی تابع زیر را بدست آورید.

```
Algorithm fib(n)
if n<=1 then return(n)
else
return(fib(n-1)+fib(n-2))
```

تعداد فراخوانی‌های a_n

دنباله‌ی فیبوناچی

n	0	1	2	3	4	5	6	...
fib	0	1	1	2	3	5	8	...

$$\begin{cases} a_0^{+1} = 1^{+1} \\ a_1^{+1} = 1^{+1} \\ a_n^{+1} = a_{n-1}^{+1} + a_{n-2} + 1 \end{cases} \Rightarrow \text{رابطه‌ی بازگشتی} \quad \begin{cases} b_n = b_{n-1} + b_{n-2} \\ b_0 = 2, b_1 = 2 \end{cases}$$

(b_0, b_1, b_n, b_{n-1}, b_{n-2})

معادله‌ی مشخصه

$$c_n r^2 + c_{n-1} r + c_{n-2}$$

حل با کمک معادله‌ی مشخصه

$$b_n - b_{n-1} - b_{n-2} = 0 \Rightarrow r^2 - r - 1 = 0 \Rightarrow r_1, r_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{1 \pm \sqrt{5}}{2}$$

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}$$

دوریشه‌ی حقیقی

$$b_n = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$\begin{cases} b_0 = c_1\left(\frac{1+\sqrt{5}}{2}\right)^0 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^0 = 2 \\ b_1 = c_1\left(\frac{1+\sqrt{5}}{2}\right)^1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^1 = 2 \end{cases} \Rightarrow \text{جواب کلی:} \quad \begin{cases} c_1 + c_2 = 2 \quad (1) \times -\left(\frac{1+\sqrt{5}}{2}\right) \\ c_1\left(\frac{1+\sqrt{5}}{2}\right) + c_2\left(\frac{1-\sqrt{5}}{2}\right) = 2 \end{cases}$$

$$\Rightarrow \begin{cases} -c_1\left(\frac{1+\sqrt{5}}{2}\right) - c_2\left(\frac{1+\sqrt{5}}{2}\right) = -1-\sqrt{5} \\ c_1\left(\frac{1+\sqrt{5}}{2}\right) + c_2\left(\frac{1-\sqrt{5}}{2}\right) = 2 \end{cases} \Rightarrow -c_2\left(\frac{1+\sqrt{5}}{2}\right) + c_2\left(\frac{1-\sqrt{5}}{2}\right) = 1-\sqrt{5}$$

$$\Rightarrow \frac{-c_2 - \sqrt{5}c_2 + c_2 - \sqrt{5}c_2}{2} = 1-\sqrt{5} \Rightarrow c_2 = \frac{-1+\sqrt{5}}{\sqrt{5}} \quad (1)$$

Scanned by CamScanner

Subject:

Year. Month. Date. ۹۱

① $\Rightarrow c_1 = \frac{1+\sqrt{5}}{\sqrt{5}}$

$c_1 = \frac{1+\sqrt{5}\times 2}{\sqrt{5}\times 2}$ و $c_2 = \frac{-1+\sqrt{5}\times 2}{\sqrt{5}\times 2}$

نکته: $\frac{1+\sqrt{5}}{2} \simeq 1,41$ عدد طلایی است

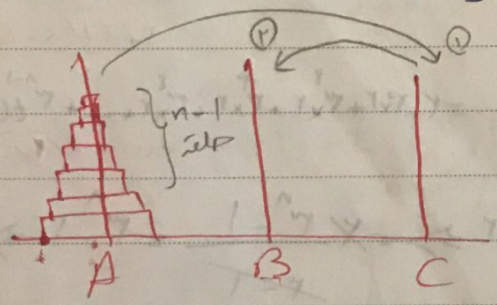
$b_n = c_1 r_1^n + c_2 r_2^n$

$b_n = \frac{1+\sqrt{5}\times 2}{\sqrt{5}\times 2}\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{-1+\sqrt{5}\times 2}{\sqrt{5}\times 2}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n$

$b_n = \frac{2}{\sqrt{5}}\left\{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right\} \Rightarrow$

$\Rightarrow \boxed{a_n = \frac{2}{\sqrt{5}}\left\{\underbrace{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}}_{(1,41)^{n+1}} - \underbrace{\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}_{(-0,41)^{n+1}}\right\} = 1} \Rightarrow O\left((1,41)^n\right)$

مسئله‌ی برج هانوی: مسئله‌ی برج هانوی را با سه میله‌ی A و B و C در نظر بگیرید. در این جا هیچ حلقه‌ای را نمی‌توان مستقیماً از A به B یا از B به A برد و جهت انتقال حلقه فقط به کمک میله‌ی C انجام می‌شود. در شروع کار حلقه در میله‌ی A داشته باشیم و $t(n)$ حداقل تعداد جابجایی حلقه‌ها برای انتقال حلقه از A به B باشد:



الف) یک رابطه‌ی بازگشتی برای $t(n)$ ارائه دهید.

ب) این رابطه‌ی بازگشتی را حل کنید.

$t(1) = 2$ ردوبدلت به دلیل قانون‌های نوشته شده است

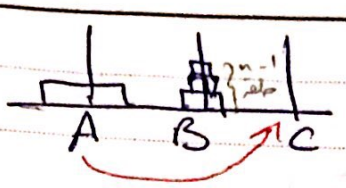
$t(n) =$

Scanned by CamScanner

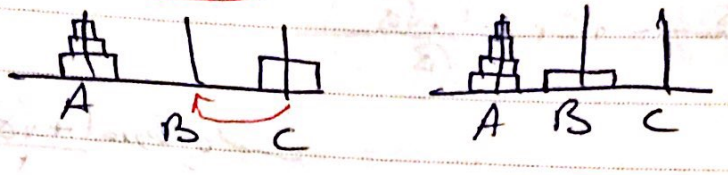
Subject:

Year. Month. Date.

$t(n-1)+1$

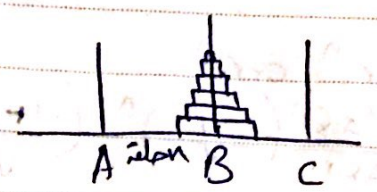


$2t(n-1)+1$



$2t(n-1)+2$

$\boxed{3t(n-1)+2}$ فرمول بازگشتی



$$t(n)=\begin{cases}3t(n-1)+2 & n\geq 2\\ t(1)=2 & n=1\end{cases}$$



جایگذاری پیشرو:

$t(n)=3t(n-1)+2=3(2+3t(n-2))+2=$ ($3t(n-1)+2$: $t(n-1)$)

$=2+3\times 2+3^2[2+3t(n-2)]=2+3\times 2+3^2\times 2+3^2t(n-2)=$ ($2+3t(n-3)$: $t(n-2)$)

$2+3\times 2+3^2\times 2+3^3\left(2+3t(n-4)\right)=2+3\times 2+3^2\times 2+3^3\times 2+3^4t(n-4)=$ ($2+3t(n-4)$: $t(n-3)$)

$2+3\times 2+3^2\times 2+3^3\times 2+3^4\times 2+3^5t(n-5)=\cdots=2+3\times 2+3^2\times 2+3^3\times 2+3^4\times 2+\cdots+3^{n-1}t(1)=$

($t(1)=2$ ← $n-(n-1)$)

$2+3\times 2+3^2\times 2+3^3\times 2+3^4\times 2+\cdots+3^{n-1}\times 2=2\times\frac{3^n-1}{3-1}=3^n-1$ ($q=3$, $\frac{q^n-1}{q-1}$: فرمول مجموع هندسی)

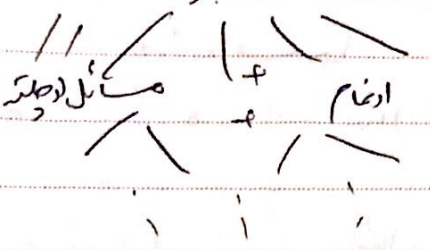
$\Rightarrow O(3^n)$

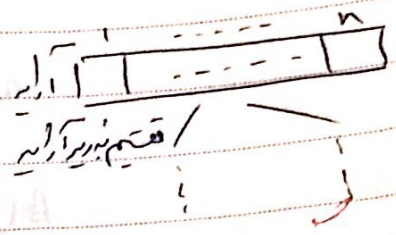
Scanned by CamScanner

Subject :

Year . Month . Date . ۱۴۰۱

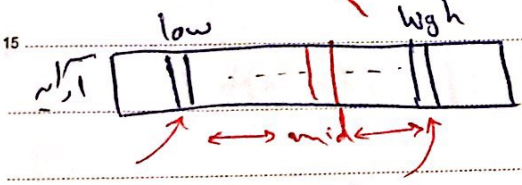
روش های تقسیم و غلبه (Divide & conquer):





صورت کلی الگوریتم تقسیم و غلبه:

```
Algorithm D&C(low&high)
if small(low,high) return G(low&high)
else{                                  تابع پولی
mid=Divide(low&high)
return combine(D&C(low,mid),D&C(mid+1,high))}
```



صورت زمان اجرای الگوریتم تقسیم و غلبه:

$$t(n)=\begin{cases} g(n) & \text{برای ورودی های کوچک} \\ t(\lceil \frac{n}{2} \rceil)+t(\lfloor \frac{n}{2} \rfloor)+f(n) & \text{برای ورودی های بزرگ} \end{cases}$$

$\frac{n}{2}$ تقریب ← $\lceil \frac{n}{2} \rceil$ ، $\lfloor \frac{n}{2} \rfloor$ تقریب →

اگر $n=2^m$ باشد زمان را به صورت زیر داریم:

$$t(n)=\begin{cases} g(n) & \text{برای ورودی های کوچک} \\ 2t(\frac{n}{2})+f(n) & \text{برای ورودی های بزرگ} \end{cases}$$

PAPCO

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

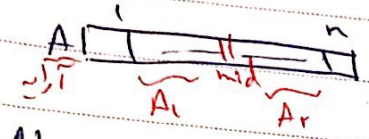
الگوریتم تقسیم و غلبه برای پیدا کردن عنصر ماکزیمم یک آرایه:

تعداد مقایسه‌ها

الف) $n=1$: مثال: A[۱۳] ۰

ب) $n=2$: مثال A[۲۳|۱۰] ۱

$n>1$ $\left\lceil \frac{n}{2} \right\rceil$ max , $\left\lfloor \frac{n}{2} \right\rfloor$ max



الگوریتم: Findmax(A,1,n,fmax) 10

Algorithm Findmax(A,low,high,fmax)

آرایه ۱ n عنصر ماکزیمم

fmax عنصر ماکزیمم

```
if (low=high) then fmax=A[low]
else if(high=low+1) then
if A[low]>A[high] then
fmax=A[low]
else fmax=A[high]
else{
mid⌊(low+high)/2⌋
Findmax(A,low,mid,lmax)
Findmax(A,mid+1,high,rmax)
if(max>rmax then fmax=lmax)
else fmax=rmax
}
```

P4PCO

Scanned by CamScanner

Subject :

Year Month Date ۱۱

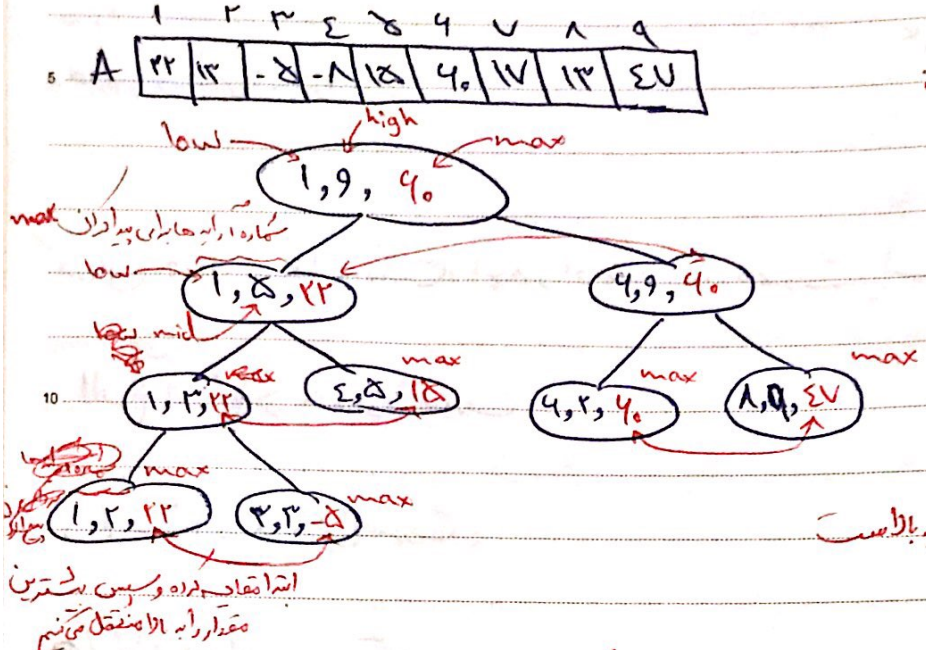
Find max (A, 1, 9, fmax)

مراحل اجرای الگوریتم به شکل:

بررسی و آرایه:

A[1 ... 9] = [۲۲, ۱۳, -۷, -۸, ۱۵, ۶۰, ۱۷, ۱۳, ۴۷]

حل با توجه به الگوریتم بصری قبل:



شماره آرایه‌های پیمایش

ابتدا مقایسه کرده و سپس بیشترین مقدار را به بالا منتقل می‌کنیم

* مراحل پیمایش max از پایین به بالاست

مرتبه‌ی زمانی الگوریتم Find max

مقایسه با خود برای پیمایش max

$$t(n) = \begin{cases} t(n) = t\left(\lceil \frac{n}{2} \rceil\right) + t\left(\lfloor \frac{n}{2} \rfloor\right) + 1 & n > 2 \\ t(1) = 0 \;,\; t(2) = 1 & \end{cases}$$

برای سهولت کار، $n = 2^m$

$$\Downarrow$$

$$t(n) = \begin{cases} t(n) = 2t\left(\frac{n}{2}\right) + 1 \\ t(1) = 0, t(2) = 1 \end{cases}$$

حل بازگشتی با تکرار:

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

$$1 + 2\underbrace{\left[1 + 2t\left(\frac{n}{2^2}\right)\right]}_{t\left(\frac{n}{2}\right)} = 1 + 2 + 2^2 t\left(\frac{n}{2^2}\right)$$

$$1 + 2 + 2^2 \underbrace{\left[1 + 2t\left(\frac{n}{2^3}\right)\right]}_{t\left(\frac{n}{4}\right)} = 1 + 2 + 2^2 + 2^3 t\left(\frac{n}{2^3}\right)$$

$$\Rightarrow = 1 + 2 + 2^2 + 2^3 + \dots + 2^{m-1} + 2^m \underbrace{t(1)}_{t(1)=0}$$

$\frac{n}{2^m} \Rightarrow n = 2^m \Rightarrow \frac{2^m}{2^m} = 1$

$$\Rightarrow = 2^m - 1 = n - 1$$

$n = 2^m$

قضاعدهش مرتبه‌ی زمانی

$$\boxed{\Theta(n)}$$

PAPCO

Scanned by CamScanner

Subject:
Year: Month: Date: ( )

تمرین: الگوریتم Findmax را طوری تغییر دهید و علاوه بر عنصر ماکزیمم، عنصر مینیمم آرایه را نیز بیابد. به صورت بازگشتی و خطی؟

قضیه‌ی اصلی برای حل روابط تقسیم و غلبه:

$$\begin{cases} t(n) = at\left(\frac{n}{b}\right) + cn^k \\ t(1) = c \end{cases}$$

جواب رابطه‌ی تقسیم و غلبه‌ی

که در آن a و b و c و k اعداد مثبت، ایده و اعداد ثابت هستند، به صورت زیر است:

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

الف) اگر $a > b^k$، در این صورت:

$$t(n) = \Theta(n^k \log n)$$

ب) اگر $a = b^k$ در این صورت:

$$t(n) = \Theta(n^k)$$

ج) $a < b^k$ در این صورت:

مثال: جواب روابط تقسیم و غلبه‌ی زیر را بدست آورید:

الف) $t(n) = \overset{a}{2}t\left(\frac{n}{2}\right) + \overset{k=0}{1}$ $\quad a > b^k \Rightarrow 2 > 2^0 \Rightarrow \Theta\left(n^{\log_2^2}\right) \Rightarrow \Theta(n)$

ب) $t(n) = \overset{a}{2}t\left(\frac{n}{2}\right) + \overset{k=1}{n}$ $\quad c=1 \Rightarrow a = b^k \Rightarrow 2 = 2^1 \Rightarrow \Theta(n\log n)$

پ) $t(n) = t\left(\frac{2}{3}n\right) + 1 \Rightarrow a=1, b=\frac{3}{2}, k=0 \Rightarrow a = b^k \Rightarrow 1 = \left(\frac{3}{2}\right)^0 \Rightarrow 1=1 \Rightarrow \Theta(n^0 \log n) \Rightarrow \Theta(\log n)$

ت) $t(n) = 2t\left(\frac{n}{2}\right) + n^2 \Rightarrow a=2, b=2, k=2 \Rightarrow a < b^k \Rightarrow 2 < (2)^2 \Rightarrow \Theta(n^k) \Rightarrow \Theta(n^2)$

~~ث) $t(n) = 2t(\sqrt{n}) + \log n, \quad n \geq 2$~~

P4PCO

Scanned by CamScanner

Subject:

Year Month Date

ب) $t(n) = 2t(\sqrt{n}) + \log n$, $n = 2^m$ $\quad t(a^b) = S(b)$ *

$\sqrt{n} = \sqrt{2^m} = 2^{\frac{m}{2}}$, $t(n) = t(2^m) = S(m)$

$n = 2^m \Rightarrow m = \log_2^n$ $\quad S(m) = 2S(\frac{m}{2}) + m \Rightarrow a=2, b=2, k=1$

$a = b^k$

$\Rightarrow$ حالت (ب) $S(m) = \Theta(m \log m)$

$\Rightarrow t(n) = \Theta(\log n \log\log n)$

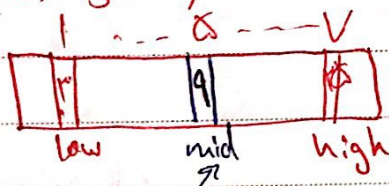
الگوریتم جست و جوی دودویی (Binarysearch):

آرایه مرتب شده $A[1 \dots n]$ به صورت صعودی و کلید k مفروض است.

هدف: آیا کلید k در آرایه A هست یا خیر؟

اگر k در آرایه باشد اندیس آن در آرایه را برگرداند (جست و جوی موفق) و اگر k در آرایه نباشد مقدار 0 را برگرداند (جست و جوی ناموفق)

P(low, high, x)



$\Theta(1)$ اگر $x = A[low]$ مقدار low را برگرداند

$\Theta(1)$ اگر $x \neq A[low]$ مقدار صفر را برگرداند

حالت پایه: اگر آرایه یک عضو داشته باشد low = high

حالت بازگشتی: اگر آرایه بیش از یک عضو داشته باشد low < high آنگاه mid مشخص می‌شود: $mid = \lfloor \frac{low+high}{2} \rfloor$

آنگاه mid مقایسه می‌شود با سه حالت زیر:

P4PCO

Scanned by CamScanner

Subject:

Year. Month. Date.

اگر $x = A[mid]$: این حالت مسئله بصورت $p(low, mid-1, x)$ حل می‌شود.

اگر $x < A[mid]$: در این حالت مسئله بصورت $p(mid+1, high, x)$ حل می‌شود.

Binary search$(A, 1, n, x)$

صورت الگوریتم

Algorithm Binary search(low, mid, x, j)

$mid \leftarrow \left\lfloor \frac{low + high}{2} \right\rfloor$

if $x = A[mid]$ then $j \leftarrow mid$ return j

else

if $x < A[mid]$ then Binary search$(low, mid-1, x, j)$

else

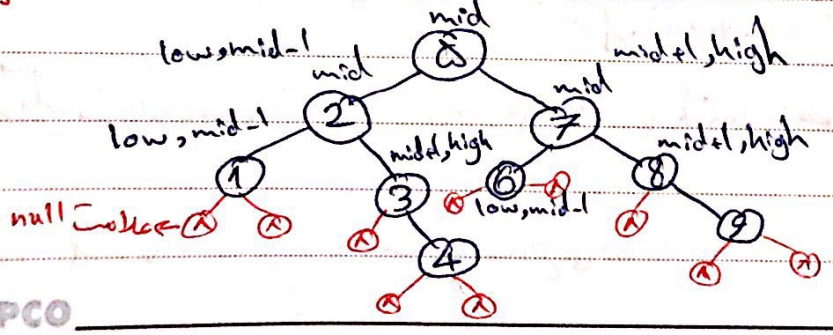
Binary search$(mid+1, high, x, j)$

توضیح الگوریتم با مثال:

$n = 9$

i	1	2	3	4	5	6	7	8	9
A[i]	-20	-10	20	30	35	40	70	90	110

درخت دودویی تصمیم



PAPCO

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9$

$A_s[i] = -20 \quad 10 \quad 20 \quad 30 \quad 25 \quad 40 \quad 70 \quad 90 \quad 180$

$3 \quad 2 \quad 3 \quad 4 \quad 1 \quad 3 \quad 2 \quad 3 \quad 4$

$3 \quad 3 \quad 3 \quad 4 \quad 4 \quad 3 \quad 3 \quad 3 \quad 4 \quad 4$

تعداد مقایسه در جست‌وجوی موفق:

تعداد مقایسه در جست‌وجوی ناموفق: $x<20 / -20<x<-10 / -1<x<20 / 20<x<25 / 25<x<40 / 40<x<70 / 70<x<90 / 90<x<180 / x>180$

حالات $n+1$ حالت جست‌وجوی ناموفق

موفق

میانگین تعداد مقایسه برای جست‌وجوی موفق $= \frac{25}{9} \approx 2,77 \rightarrow S_n = \frac{\sum \cdots}{n}$

ناموفق $= \frac{34}{10} = 3,4 \rightarrow U_n = \frac{\sum \cdots}{n+1}$

تحلیل زمانی الگوریتم جست‌وجوی دودویی:

حالت / مرتبه زمانی	بهترین حالت	بدترین حالت	حالت میانگین
جست‌وجوی موفق	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$
جست‌وجوی ناموفق	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$

مثال ضرب دو چند جمله‌ای:

$1 + x + x^2 + 2x^3$

$2 + 2x^2 + x^3$

$\Rightarrow$ ضرب $\rightarrow 2 + 2x^2 + x^3 + 2x + 2x^3 + x^4 + 2x^2 + 2x^4 + x^5 + 4x^3 + 4x^5 + 2x^6$

$n \Rightarrow$ حاصل ضرب $2n$

مثال:

ورودی:

$A(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

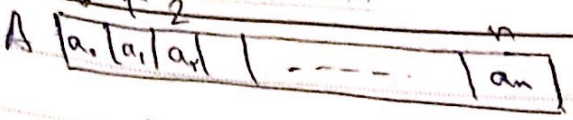
$B(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$

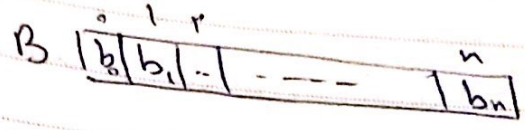
PAPCO

Scanned by CamScanner

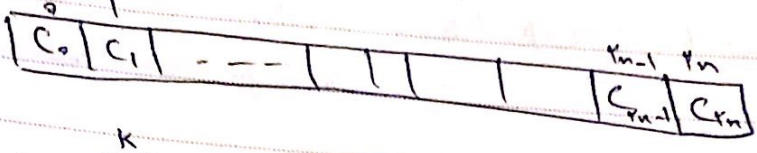
Subject:

Year Month Date





$C(x) = A(x)B(x) = c_{2n}x^{2n} + c_{2n-1}x^{2n-1} + \cdots + c_1x + c_0$ هدف: بدست آوردن



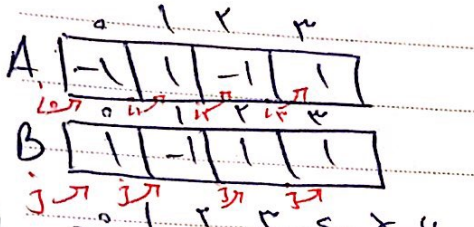
$$c_k = \sum_{i=0}^{k} a_i b_{k-i} \quad , k = 0,1,\dots,2n$$

$c_0 = a_0b_0$, $c_1 = a_0b_1 + a_1b_0$, $c_2 = a_0b_2 + a_1b_1 + a_2b_0$

$A(x) = x^3 - x^2 + x - 1$

$B(x) = x^3 + x^2 - x + 1$

توضیح همراه با مثال:



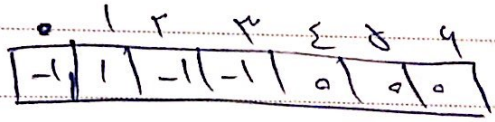
۱ روش معمولی

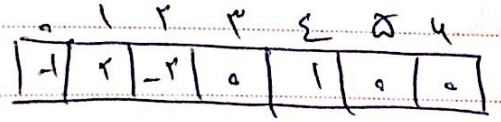
۲ روش تقسیم و غلبه معمولی

۳ روش تقسیم و غلبه سریع

روش ۱: روش معمولی

```
Algorithm mulpoly (A,B,C)
for i:=0 to 2n do c[i]:=0. Repeat
for i:=0 to n do                    n      n+n^2
    c[i+j]:=c[i+j]+A[i]*B[j]        <  for j:=0 to n do  n^2  Θ(n^2)
```





Scanned by CamScanner

i و ۲

-۱	۲	-۳	۱	۰	-۱	۰

i و ۳

۰	۱	۲	۳	۴	۵	۶
-۱	۲	-۳	۲	-۱	۰	۱

5 $C(x) = x^6 - x^5 + 2x^3 - 2x^2 + 2x - 1$

ضرب دو چند جمله ای:

۱) به صورتی ساده از $O(n^2)$ شروع می‌شود ۲) روش تقسیم و غلبه سریع

تقسیم

۱) روش ساده و معمولی ۲) روش تقسیم و غلبه ای معمولی

۲- ضرب دو چند جمله ای با روش تقسیم و غلبه ای معمولی:

یادآوری:

توضیح همراه با مثال:

$P(x)$ n از درجه

$Q(x)$ n از درجه

$R(x) = P(x)Q(x)$ $2n$ از درجه

$P_L(x)$ $P_R(x)$

مثال: $P(x) = x^3 - x^2 + x - 1 = (-1 + x) + x^2(-1 + x)$

$Q(x) = x^3 + x^2 - x + 1 = (1 - x) + x^2(1 + x)$

$Q_L(x)$ $P_R(x)$

$$R(x) = [P_L(x) + x^2(P_R(x))][Q_L(x) + x^2(Q_R(x))] \Rightarrow$$

$$\Rightarrow = P_L(x)Q_L(x) + [P_L(x)Q(x) + P_R(x)Q_L(x)]x^2 + (P_R(x)Q_R(x))x^4$$

$$P_L(x)Q_L(x) = (-1 + x)(1 - x) = -1 + 2x - x^2$$

Scanned by CamScanner

Subject:
Year. Month. Date. ( )

$P_L(x)Q_r(x) = (-1+x)(1+x) = x^2-1$

$P_r(x)Q_L(x) = (-1+x)(1-x) = -1+2x-x^2$

$P_r(x)Q(x) = (-1+x)(1+x) = x^2-1$

$R(x) = -1+2x-x^2 + \left[\overbrace{x^2-1-1+2x-x^2}^{-2+2x}\right]x^2 + (x^2-1)x^4 \Rightarrow$

$\Rightarrow = -1+2x-2x^2+2x^3-x^4+x^4$

	0	1	2	3	4	5	6
C	-1	2	-2	2	-1	0	1

حل تمرین ۷

$P(x) = 1-x+2x^2+2x^3+4x^4-5x^5+6x^6+7x^7$

$\rightarrow P(x) = \underbrace{(1-x+2x^2+2x^3)}_{P_L(x)} + x^4\underbrace{(4-5x+6x-7x^2)}_{P_r(x)}$

تحلیل زمانی:

فرض: $n = 2^m$; تعداد مراحل الگوریتم n

$P(x), Q(x)$ درجه $n \leftarrow 2^m-1$

$R(x) = P(x)Q(x)$.

$P_L(x)Q_L(x) + \left[P_L(x)Q_r(x) + P_r(x)Q_L(x)\right]x^{\frac{n}{2}} + P_r(x)Q_r(x)x^n$

$T(n) = 3T\left(\frac{n}{2}\right) + Cn$

۳ تا تعداد عملیات الگوریتم $\frac{n}{2}$ — عملیات الگوریتم

۸-۱

Scanned by CamScanner

Subject
Year Month Day

$a > b^k \Leftarrow \begin{matrix} a=2 \\ b=2 \\ k=1 \end{matrix}$ الگوریتم استراسن

$2 > 1$

$t(n) = \Theta(n^{\log_2^2}) = \Theta(n^2)$

$P(x)$
$Q(x)$ $\quad R(x) = P(x)Q(x)$

$R(x) = \underbrace{P_L(x)Q_L(x)}_{R_L(x)} + \underbrace{\left[P_L(x)Q_R(x) + P_R(x)Q_L(x)\right]}_{R_m(x)} x^{\frac{n}{2}} + P_R(x)Q_R(x)x^n$

$R_m = \left[P_L(x) + P_R(x)\right]\left[Q_L(x) + Q_R(x)\right] \Rightarrow$

$\Rightarrow = \underbrace{P_L(x)Q_L(x)}_{R_L(x)} + \underbrace{\left[P_L(x)Q_R(x) + P_R(x)Q_L(x)\right]}_{(*)} + \underbrace{P_R(x)Q_R(x)}_{R_R(x)}$

$(*) = R_m(x) - R_L(x) - R_R(x) \Rightarrow R(x) = R_L(x) + \left[R_m(x) - R_L(x) - R_R(x)\right]x^{\frac{n}{2}} + R_R(x)x^n$

$t(n) = 3t\left(\frac{n}{2}\right) + Cn$

$\begin{cases} \Theta(n^{\log_2^3}) \\ = \Theta(n^{1.58}) \end{cases} \Leftarrow \begin{matrix} a>b^k \\ 3>2 \end{matrix} \Leftarrow \begin{matrix} a=3 \\ b=2 \\ a=1 \end{matrix}$ الگوریتم

$R_L(x) = P_L(x)Q_L(x) = -1 + 2x - x^2$

$R_R(x) = P_R(x)Q_R(x) = x^2 - 1$

$R_m(x) = \left[P_L(x) + P_R(x)\right]\left[Q_L(x) + Q_R(x)\right] = \left[-2 + 2x\right]\left[2\right] = -4 + 4x$

Scanned by CamScanner

Subject:
Year. Month. Date.

$(*) = R_m(x) - R_L(x) - R_r(x) = -4 + 4x + 1 - 2x + x^2 - x^2 + 1 = -2 + 2x$

$R(x) = R_L(x) + [(*)]x^{\frac{n}{2}} + R_r(x)\,x^n$

* نکته: از روش تقسیم و غلبه‌ی سیو برای ضرب اعداد صحیح خیلی بزرگ استفاده می‌شود

مثال: $1234 * 2343$

$R_m(x) = \underbrace{(4 + 2x + 2x^2 + x^3)}_{P(x)} \underbrace{(3 + 4x + 3x^2 + 2x^3)}_{Q(x)} \quad x = 10$

فصل ۴:

روش‌های حریصانه (Greedy):

۱. روش‌های هسته تکرار شونده (iterative)

۲. برای بهینه سازی (ماکزیمم یا مینیمم) کردن یک تابع به کار می‌رود.

۳. این روش‌ها معمولاً دارای n ورودی بوده و مسئله از ما می‌خواهد تا زیر مجموعه‌ای از این n ورودی بهینه آوریم به گونه‌ای که در شرایط معینی صدق کند

آرایه‌ای که در شرایط معینی صدق کند

فرض کنید ورودی یک آرایه n عنصری مثل $[n \dots 1]$ A است در هر مرحله یکی از عناصر آرایه بر اساس

Scanned by CamScanner

Subject:
Year Month Date

(select)

یک تابع از بین انتخاب ها را (با فرض) آن عنصر به مجموعه جواب های انتخاب شده تا حال اضافه کردن

بودن (feasible) به این مجموعه جواب اضافه می شود (union)

صورت عمومی یک الگوریتم حریصانه:

```
Algorithm Greedy(A,n)
Solution = ∅          (بردار جواب) مجموعه جواب
for i:=1 to n do
{
x:=Select(A)          انتخاب یک عنصر از آرایه
if Feasible(x,Solution) then     یعنی آیا به درستی مسئله صدق می کند یا خیر.
Solution:=Union(x,Solution)      x را به مجموعه جواب اضافه می کند
}
```

مسئله کوله پشتی (knapsack problem)

فرض این که کوله پشتی به ظرفیت M و n جسم داریم که از یک تا n شماره گذاری شده اند.

w_i وزن جسم i ام

P_i ارزش جسم i ام

هدف: یافتن x_i هایی که $\sum_{i=1}^{n} x_i P_i$ ماکزیمم باشد

x_i: درصد برداشته شده از جسم i ام مشروط به آن که $\sum_{i=1}^{n} x_i w_i \le M$

$$* \quad \sum_{i=1}^{n} w_i > M$$

توضیح الگوریتم حداکثر امثال

Scanned by CamScanner

Subject:

Year: Month Date:

فرض $n=3$ و $M=20$ و $(w_1, w_2, w_3)=(18, 15, 10)$ و $(p_1, p_2, p_3)=(25, 24, 15)$ ارزش

i	(x_1, x_2, x_3)	$\sum_{i=1}^{3} x_i w_i$	$\sum_{i=1}^{3} x_i p_i$
۱)	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{4})$	$\frac{18}{3}; \frac{15}{3}, \frac{10}{4}$ $9+5+2.5=16.5$	$\frac{25}{3}, \frac{24}{3}, \frac{15}{4}=23.5$
۲)	$(1, \frac{2}{15}, 0)$	$18+2=20$	$25+25\times\frac{2}{15}=28.13$
۳)	$(0, \frac{10}{15}, 1)$	$10+10=20$	$15+\frac{10}{15}\times 24=31$
۴)	$\frac{p_i}{w_i}=\frac{(p_1, p_2, p_3)}{(w_1, w_2, w_3)}=(\frac{25}{18}, \frac{24}{15}, \frac{15}{10})$ $(1.38, 1.60, 1.50)$		
	$(0, 1, \frac{1}{2})$	$15+5=20$	$24+7.5=31.5$

* نکته: الزاماً باید ترتیب نزولی ارزش بر واحد وزنشان مرتب کنیم به جواب بهینه خواهیم رسید.

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$$

الگوریتم:

۱) مرتب سازی $\frac{p_i}{w_i}$ ها با پیچیدگی زمانی $O(n\log n)$

۲) الگوریتم حریصانه کوله‌پشتی

Algorithm Greedy_knapsack(n, W, P, M, X, profit)

تعداد اشیاء — وزن‌ها — ارزش‌ها — وزن کوله — جواب بهینه — سود نهایی

```
x=0, profit=0, cu=M{
for i=1 to n=do{
if w[i]≤cu then{
 x[i]=1, profit+p[i]}
 cu=cu-w[i]}
 else exit}
 x[i]=cu/w[i];
 profit=profit+X[i]*P[i]
```

$O(n) < O(n\log n)$

و مرتب سازی آن $O(n\log n)$ است

Subject:
Year. Month. Date.

صورت مسئله

مثال: مسئله کوله پشتی زیر را حل کنید.

$n=4$, $M=14$

$p=(p_1,p_2,p_3,p_4)=(10,7,12,9)$

$w=(w_1,w_2,w_3,w_4)=(2,1,2,8)$

$\frac{p_i}{w_i}=(5,7,6,1.5)$ ④ ② ③ ①

$C=14$

profit = 0

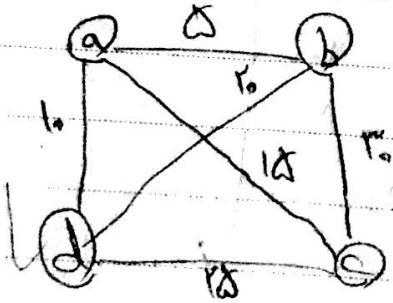
i	w_i	p_i	$\frac{p_i}{w_i}$ (ارزش نهایی)	مقدار داشتنی x_i (جزء پذیر است)	وزن نهایی $C-w$	profit (ارزش نهایی)
1	2	10	5	1	9	$10+20=30$ ③
2	1	7	7	1	13	$0+7$ ①
3	3	12	4	1	6	$12+30=42$ ②
4	2	13	6.5	1	11	$7+13=20$ ④
5	12	9	0.5	0	—	
6	8	20	2.5	$\frac{6}{8}$	0	$42+\frac{6}{8}\times 20=57$ ⑤

$\begin{cases} X(1,1,1,1,0,\frac{6}{8}) \\ profit=57 \end{cases}$

درخت پوشا: فرض کنید $G(V,E)$ گراف همبند بدون جهت باشد زیرگراف $T=(V',E')$ از G یک درخت پوشا برای G نامیم هرگاه:

1) یک درخت باشد و 2) $V'=V$ باشد

مثال:



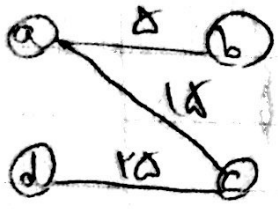
تعداد درخت‌های پوشا: $4^{4-2}=16$

Scanned by CamScanner

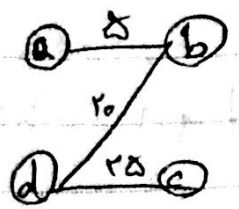
Subject
Year Month Date

قضیه: تعداد درخت های پوشای گراف کامل K_n برابر n^{n-2} است.

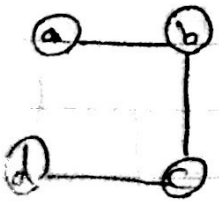
وسع درخت گراف صفحه قبل:

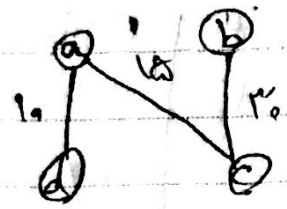


هزینه (cost): ۴۵



هزینه (cost): ۵۰





هزینه (cost): ۵۵

هدف: بدست آوردن کمینه درخت پوشا برای یک گراف وزن دار

۱) پریم (prim)

۲) کراسکال (Kruscal)

در هر کدام از این الگوریتم ها ورودی برای است گراف وزن دار و همبند است $G=(V, E, cost)$

که در آن $V=\{v_1, v_2, \dots, v_n\}$ مجموعه رئوس و $cost[i,j]$ هزینه یال (v_i, v_j) بنابراین ماتریس $cost[i,j]$

یک ماتریس $n \times n$ خواهد بود و یک هزینه‌ی یال هاست. خروجی هر کدام از این الگوریتم ها درختی مثل T از

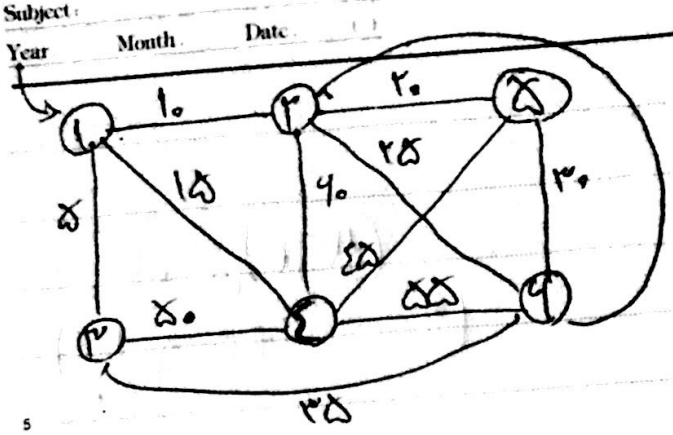
$n-1$ یال برای درخت پوشای مینیمم است

$$cost = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} \infty & ۵ & ۱۵ & ۱۰ \\ ۵ & ۰ & ۳۰ & ۲۰ \\ ۱۵ & ۳۰ & ۰ & ۲۵ \\ ۱۰ & ۲۰ & ۲۵ & ۰ \end{bmatrix} \end{array}_{۴\times۴}$$

ماتریس هزینه

Scanned by CamScanner

الگوریتم پریم حمداد باسال:



ابتدا از یک رأس دلخواه شروع می‌کنیم (رأس ۱)، این رأس را برداشته شده فرض می‌کنیم و بقیه‌ی رئوس هنوز برداشته نشده اند. در هر مرحله رأسی از بین رئوس برداشته نشده به گونه‌ای انتخاب خواهیم کرد که با حداقل هزینه به یکی از رئوس قبلاً برداشته شده وصل شده باشد. در هر مرحله مجموعه رئوس به دو زیر مجموعه‌ی برداشته شده با نام P و برداشته نشده با نام N تقسیم می‌شوند. در هر مرحله یک رأس از N (مثلاً j) و یک رأس از P (مثلاً i) به گونه‌ای انتخاب می‌کنیم که cost[i][j] مینیمم باشد
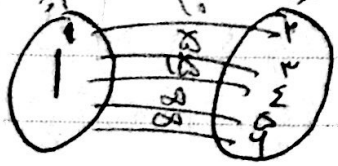
$P = P \cup \{j\}$

$N = N - \{j\}$

cost $\{i, j\}$ مینیمم باشد

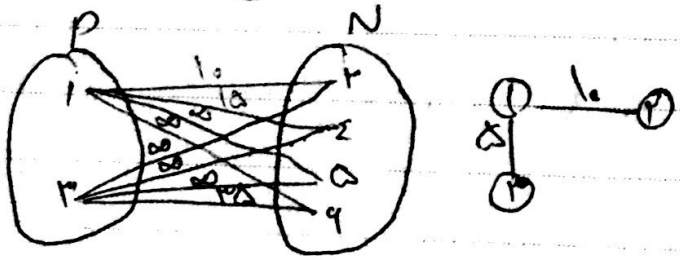
P N

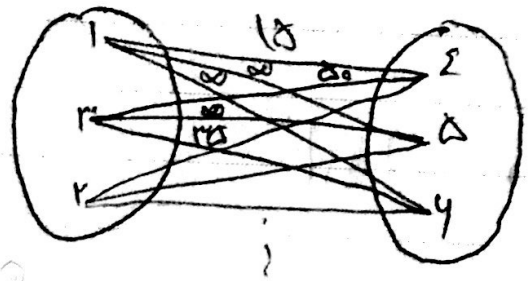
P برداشته شده — N برداشته نشده





۵۵: به دلیل نداشتن مسیر نوشته می‌شود





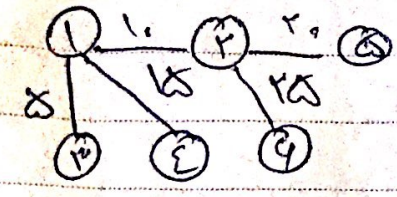
این کار را تا جایی که تمامی اعداد برداشته شده و شود ادامه می‌دهیم

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

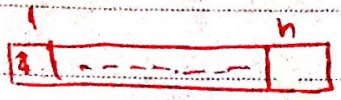
جواب آخر



درخت پوشای مینیمم (MST)

(Minimum spanning Tree)

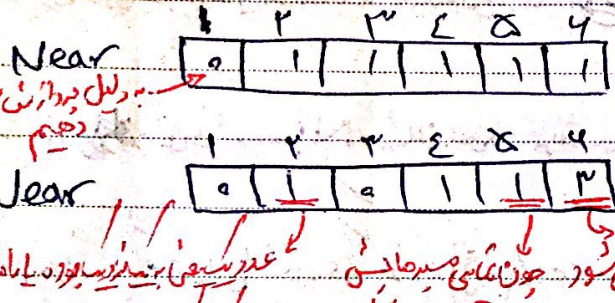
Klear



روند الگوریتم:

رأس برداشته نشده Near[j] ≠ 0

Near[j] ≠ i ≠ 0 الوسیار [j, i] cost مینیمم باشد



الگوریتم پریم:

ورودی — خروجی

Algorithm prim (n, cost, T, mincost)

Near[1] = 0 , min cost = 0 O(1)

for i = 2 to n do Near[i] = 1 Repeat O(n)

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

رأس j را به گونه‌ای انتخاب کن که Near[j] ≠ 0 و

cost [j, Near[j]] هم مینیمم باشد.

(T[i,1], T[i,2]) = (j, Near[j])

Min cost = Mincost + cost [j, Near[j]]

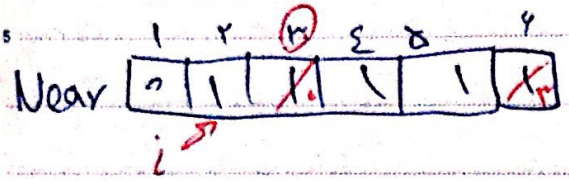
Near[j] = 0

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

```
for k=2 to n do
if (Near[k]≠0) and
(cost[k,j] < cost[k, Near k]) then
  Near[k]=j
```



مرتبه‌ی زمانی الگوریتم $O(n)$ است.

مثال: مراحل مختلف اجرای الگوریتم پریم بروی گراف مثال قبل را بنویسید.

j	Near[j]	cost[j, Near[j]]
۱	۰	—
۲	۱	۱۰
۳	۱	۵
۴	۱	۱۵
۵	۱	∞
۶	۱	∞

مرحله‌ی اول

min cost: ۰+۵=۵

j	Near[j]	cost[j, Near[j]]
۱	۰	—
۲	۱	۱۰
۳	۰	—
۴	۱	۱۵
۵	۱	∞
۶	۳	۳۵

مرحله‌ی دوم

min cost: ۵+۱۰=۱۵

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

j	Near[j]	cost[j, Near[j]]
۱	۰	—
۲	۰	—
۳	۰	—
۴	۱	۱۵ ←
۵	۲	۲۰
۶	۲	۲۵

mincost: ۱۵ + ۱۵ = ۳۰

j	Near[j]	cost[j, Near[j]]
۱	۰	—
۲	۰	—
۳	۰	—
۴	۰	—
۵	۲	۲۰ ←
۶	۲	۲۵

mincost = ۲۰ + ۲۰ = ۵۰

j	Near[j]	cost[j, Near[j]]
۱	۰	—
۲	۰	—
۳	۰	—
۴	۰	—
۵	۰	—
۶	۲	۲۵ ←

mincost: ۵ + ۲۵ = ۷۵

Scanned by CamScanner

Subject:
Year. Month. Date.

j	Near[j]	cost[j, Near[j]]
۱	۰	
۲	۰	
۳	۰	
۴	۰	
۵	۰	
۶	۰	

خاتمه الگوریتم

نکته: درخت پوشای یک گراف با n رأس دارای n-۱ یال خواهد بود

الگوریتم کراسکال:

۱/ مرتب سازی یال ها به صورت صعودی براساس هزینه ی آنها

۲/ با شروع از ابتدای لیست مرتب شده ی یال ها، یالی را انتخاب می کنیم به اضافه ی آن به مجموعه ی یال هایی که قبلاً انتخاب شده اند دور تشکیل ندهد. این عمل را آنقدر ادامه می دهیم تا (n-۱) امین یال انتخاب شود

تعداد یال ها

ن	هزینه	یال
۱	۵	۱و۳
۲	۱۰	۱و۲
۳	۱۵	۱و۴
۴	۲۰	۲و۵
۵	۲۵	۲و۶
۶	۳۰	۵و۶
۷	۳۵	۳و۶
۸	۴۰	۲و۶
۹	۴۵	۴و۵
۱۰	۵۰	۳و۴
۱۱	۵۵	۴و۶
۱۲	۶۰	۲ ۴

با توجه به یال اول شان زده شده داریم:

PAPCO

Scanned by CamScanner

Subject:

Year. Month. Date. ( )

مرحله‌ی اول: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$



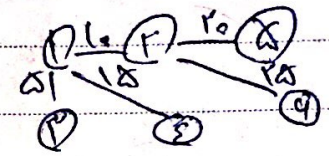
مرحله‌ی دوم: $\{1,3\}\ \{2\}\ \{4\}\ \{5\}\ \{6\}$

مرحله‌ی سوم: $\{1,2,3\}\ \{4\}\ \{5\}\ \{6\}$

مرحله‌ی چهارم: $\{1,2,3,4\}\ \{5\}\ \{6\}$

مرحله‌ی پنجم: $\{1,2,3,4,5\}\ \{6\}$

مرحله‌ی ششم و مرحله‌ی آخر: $\{1,2,3,4,5,6\}$



کوتاه‌ترین مسیرهای هم مبدأ (الگوریتم دایکسترا): یا Dijkstra shortest paths می‌باشد

فرض: $G=(V,E,cost)$

V: رئوس

E: یال‌ها

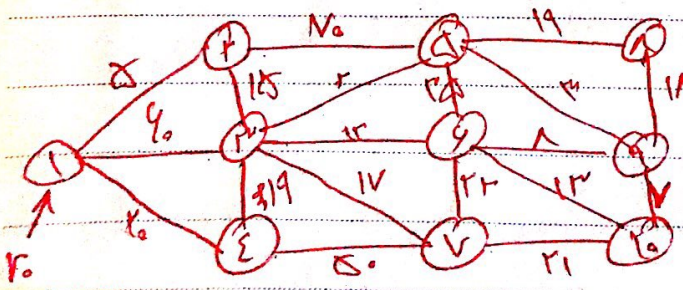
$cost$: هزینه‌ی یال‌ها

$\{1,2,\dots,n\}$ ← رأس مبدأ s

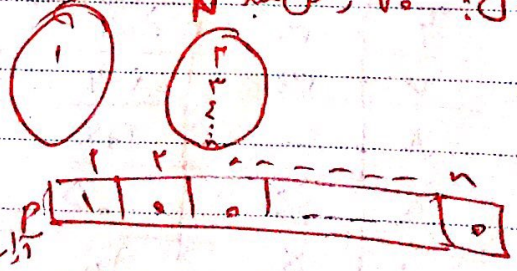
کوتاه‌ترین از رأس v_0 (رأس مبدأ) به سایر رئوس

در این الگوریتم در هر مرحله رئوس V به دو زیرمجموعه‌ی پردازش شده و پردازش نشده تقسیم می‌شوند

ایده: در ابتدا v_0 (رأس مبدأ) را جزو رئوس پردازش شده و بقیه‌ی رئوس را پردازش نشده در نظر می‌گیریم.



مثال: v_0 رأس مبدأ



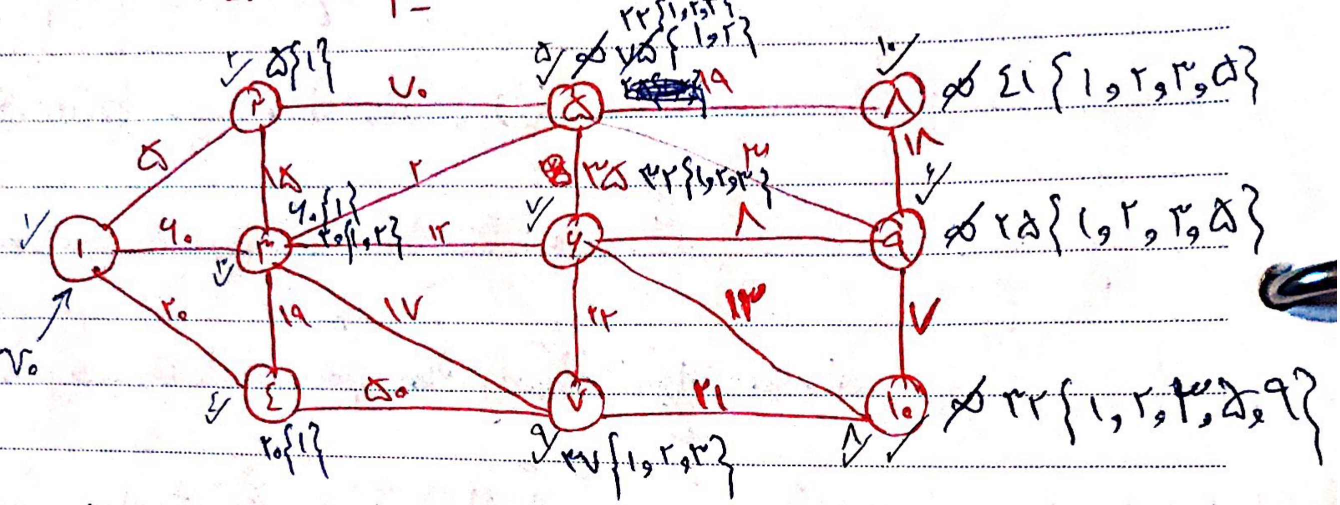
p: پردازش شده، $p[i]=0$: پردازش نشده و مرتبه‌ی اجرایی: $O(n^2)$

Scanned by CamScanner

Subject:
Year. Month. Date. ( )

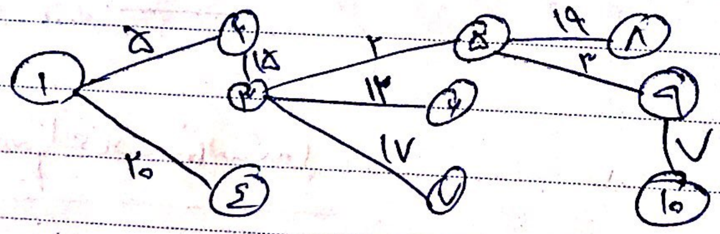
تعریف: برای هر $t \in N$ پیدا نشده ($p[t] = ۰$) مسافت $dist[t]$ برابر است با طولانی‌ترین مسیری که از رأس v_0 شروع شده و به رأس t ختم شود به‌طوری که تمامی رئوس داخلی آن پیدا شده باشند.

مثال: کوتاه‌ترین مسیرها از رأس ۱ به سایر رئوس را با کمک الگوریتم دایکسترا بیابید.



	۱	۲	۳	٤	۵	۶	۷	۸	۹	۱۰
P	۱	۱	۱	۱	۱	۱	۱	۱	۱	۱

مراحل: ① ② ③ ④ ⑤ ⑦ ⑨ ⑩ ⑥ ⑧



یک درخت پوشا

تمام رئوس — $n-۱$ یال

نکته: درخت SPST با MST یکی نیست.

تمرین: با کمک الگوریتم کراسکال و پریم، MST این گراف را بنویسید.

PAPCO

Scanned by CamScanner

Subject:
Year. Month. Date. ( )

مسئله انتخاب فعالیت‌ها (زمانبندی فعالیت‌ها) (Activity selection):

فرض: مجموعه‌ای از n فعالیت $\{a_1, a_2, \dots, a_n\} = S$ که منبع واحد مشترک است. فاصله‌ی زمانی برای استفاده‌ی فعالیت a_i از منبع به صورت $[s_i, f_i)$

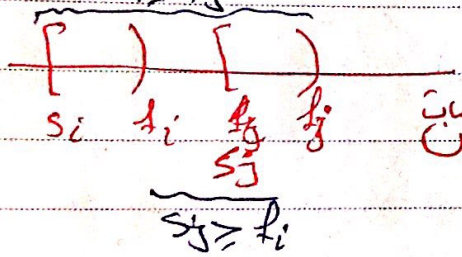
s_i: زمان شروع ، f_i: زمان پایان

هدف: انتخاب بیشترین تعداد فعالیت‌های سازگار

نکته: حتی زمان داده شده هم بین فعالیت‌ها می‌توانند از منبع به اشتراک استفاده کنند

تعریف دو فعالیت سازگار: هر فعالیت را با یک بازه در نظر می‌گیریم هر دو هم پوشانی زمان جهت استفاده از منبع را

$s_i \geq f_j$ یا $s_j \geq f_i$ یعنی



فعالیت سازگار: ——

فعالیت ناسازگار: ——

الگوریتم انتخاب فعالیت‌ها:

① مرتب سازی فعالیت‌ها به صورت صعودی زمان خاتمه‌ی آن‌ها: (f_i)

② Algorithm Activity_selection(s, f)
A = {1}, j = 1
for i = 2 to n do
if s[i] ≥ f[j] then {
A = A ∪ {i}
j = i }
return A;

پیچیدگی زمانی $O(n \log n)$

Scanned by CamScanner

Subject:

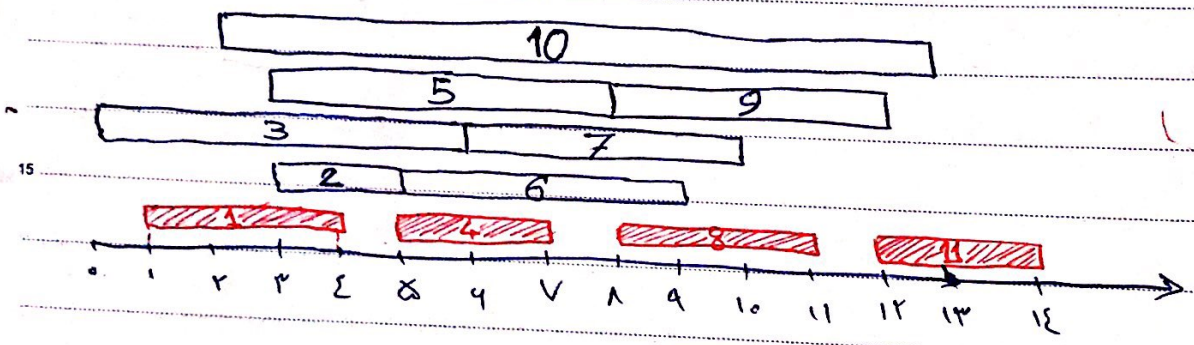
Year. Month. Date. ( )

مثال: مراحل مختلف اجرای الگوریتم انتخاب فعالیت ها را برای فعالیت های زیر بنویسید

i	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱
S_i	۱	۳	۰	۱۲	۱۴	۵	۳	۸	۸	۵	۶
f_i	۴	۵	۶	۱۴	۱۳	۹	۸	۱۲	۱۱	۷	۱۰

مرحله ۱: مرتب سازی بر اساس f_i:

i	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱
S_i	۱	۳	۰ (از لحظه شروع)	۵	۳	۵	۶	۸	۸	۲	۱۲
f_i	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴ (لحظه خاتمه)



برنامه نویسی:

اصل بهینه سازی:

مسئله فروشنده دوره گرد (Tsp) (Traveling sales person problem)

هدف یافتن کوتاه ترین مسیر برای فروشنده به طوری که از تمام شهرها یک بار عبور کند

دور هامیلتونی: در یک گراف جهت دار به دور هامیلتونی عبارت است از مسیری که از یک رأس به خودش برگردد و از تمام رأس ها دقیقا یک بار عبور کند

PqPco

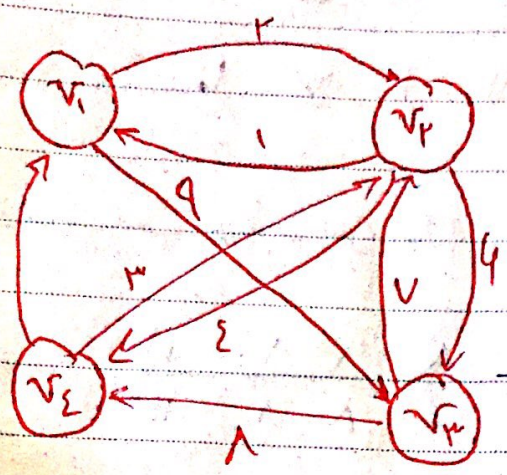
Scanned by CamScanner

Year. Month. Date. ( )

طول آن حداقل (Minimum) باشد. بنابراین مسئله‌ی فروشنده‌ی دوره‌گرد دقیقاً یافتن همین دور

است

مثال: دور همیلتونی مشکل زیر را پیدا کنید.



طول: $[v_1 و v_4 و v_3 و v_2 و v_1] = 22$

طول: $[v_1 و v_2 و v_4 و v_3 و v_1] = 21$ ←

طول: $[v_1 و v_4 و v_2 و v_3 و v_1] = 26$

Scanned by CamScanner