دانشگاه جامع
علمی - کاربردی

عنوان:

طراحی الگوریتم ها

با شبه کدهای C++

نیپولیتان

حل تمرینات چهار فصل

استاد:

جناب آقای مهندس کامبیز فخر

ارائه دهندگان:

الهام برزگر – سمیرا رنجبران اصل

تابستان ۱۳۹۰

فهرست

تمرینات

فصل اول

بخش ۱-۱:

۱)الگوریتمی بنویسید که بزرگترین عدد موجود در یک لیست (آرایه) از n عدد را بیابد.

```
Int Maximum (int max, const S[1..n]Indexi)

Max=S[1];

For(i=1;I<=n;i++)

If(max<S[i])

Max=s[i];

Yeturn   max;
```

۲-الگوریتمی بنویسید که کوچکترین عدد موجود در یک لیست (آرایه) از n عدد را بیابد

```
Int minimum (int  min, const s[1..n] Indexi)

Min=s[1];

For(i=1;1<=n;i++)

If(min>s[i])

Min=s[i];

Veturn min;
```

۳-الگوریتمی بنویسید که همه زیر مجموعه های سه عنصری از یک مجموعه n عنصری را چاپ کند. عناصر این مجموعه در لیستی نگهداری می شوند که ورودی الگوریتم به شمار می رود.

```
For(i=0;i<n-2, i++)

For(j=i+1;j<n-1,j++)

For(k=j+1;k<n;k++)

Cout<<s[i]≤s[j]≤s[k]<<end;
```

٤-یک الگوریتم مرتب سازی جایگذاری بنویسید که برای یافتن موقعیت جایگذاری بعدی از جستجوی دودویی استفاده کند.

الگوریتم Howard برای مرتب سازی عناصر آرایه A[1…n]

Algorithm Howard (low,high)

If A[low]>A[high] exchange A[low]↔A[high]

If low+1<high

$k = \left\lfloor \frac{high - low + 1}{3} \right\rfloor$ $1^{L}\ 2\ 3^{L+k}\ 4\ 5\ 6^{h-k}\ 7\ 8^{h}$

Howard(low,high-k) k=$k = \left\lfloor \frac{8-1+1}{3} \right\rfloor = 2$

Howard(low+k,high)

Howard(low,high-k)

۵-الگوریتمی بنویسید که بزرگترین مقسوم علیه مشترک دو عدد صحیح را بیابد؟

Void Bmm(int m,int n)

تابع استاندارد محاسبه Bmm به روش نردبانی یا خوارزمی:

If(n>m)

return Bmm(n,m);

Else

If(n==0)

Return m;

Else

Return Bmm(n,m%n);

۶-الگوریتمی بنویسید که هم کوچکترین و هم بزرگترین عدد موجود در لیستی از n عدد را بیابد . سعی کنید روشی بیابید که حداکثر حدود 1/5n مقایسه روی عناصر آرایه انجام دهد.

الگوریتم یافتن minو max آرایه A[1..n] با جفت کردن کلیدها (nزوج)

```
If(A[1]<A[2])
{
Min=A[1];
Max=A[2];
{
Else
Min=A[2]
Max=A[1]
For(i=3;i<=n-1;i=i+2)
If(A[i]>A[i+1])
Swap(A[i],A[i+1]);
If(A[i]<min)min=A[i];
If(Ai+1]>max)max=A[i+1];
```

۷-الگوریتمی بنویسید که تعیین کند آیا یک درخت دودویی نسبتاً کامل، یک heap است یا خیر.

این زیر برنامه به صورت heapify(A,i) فراخوانی می شود.

زیر برنامه فرض می کند زیر درخت ها ی با ریشه left(i) و right(i) خود هیپ هستند.

ولی A[i] ممکن است از فرزندانش کوچکتر باشد و خاصیت هیپ را نقض کرده باشد.

این زیر برنامه ، A[i} را با فرزندانش مقایسه می کند، و اگر از فرزندانش کوچکتر بود، آن را با بزرگترین فرزندش تعویض می کند و این عمل را آنقدر انجام می دهد، تا خاصیت هیپ برقرار شود.

```
Heapify(A,i)
{
L←left(i)
R←right(i)
If L≤heap-size [A]and A[L]≥A[i]
Then largest←
Else largest←i
If R≤heap-size[A]andA[R]>A[largest]
Then largest←R
If largest≠i
Then{exchange A[i]↔A[largest]
Heapify(A,largest)
}
}
```

بخش ۲-۱:

۸- تحت چه شرایطی ، هنگامی که یک عمل جست و جو مورد نیاز است، جست و جوی ترتیبی (الگوریتم ۱-۱) مناسب نخواهد بود؟

جست و جوی ترتیبی n مقایسه انجام می دهد تا تعیین کند آیا x در آرایه ای به اندازه n وجود دارد یا خیر. این حداکثر تعداد مقایسه است که این جست و جو انجام می دهد. اگر x در آرایه باشد، تعداد مقایسه ها بزرگ تر از n نخواهد بود. وقتی که مقدار nافزایش بیش از حد داشته باشد، segsearch جستجو را در زمان قابل تحمل انجام نداده و تعداد مقایسه ها نیز خیلی زیاد می شود.

۹)یک مثال عملی بزنید که در آ‹ از مرتب سازی تعویضی (الگوریتم ۳-۱)برای انجام عمل مرتب سازی استفاده می شود.

S[1]	s[2]	s[3]	s[4]	s[5]	
53	65	42	58	37	
42	65	53	58	37	
37	65	53	58	42	انتهای گذر اول:
37	53	65	58	42	
37	42	65	58	53	انتهای گذر دوم:
37	42	58	65	53	
37	42	53	65	58	انتهای گذر سوم:
37	42	53	58	65	انتهای گذر چهارم:

بخش۳–۱:

۱۰)عمل اصلی را در الگوریتم های خود برای تمرین های ۷–۱ تعریف کنید و کارایی این الگوریتم ها را مطالعه کنید. اگر یک الگوریتم مفروض دارای پیچیدگی زمانی در هر حالت است. آن را معین کنید. در غیر اینصورت پیچیدگی زمانی در بدترین حالت را بیابید.

Max در s باشد. $A(n)=\sum_{i=1}^{n}(i\times\frac{1}{n})=\frac{1}{n}\times\frac{n(n+1)}{2}=\frac{n+1}{2}$ و w(n)=n و B(n)=1 ⟶ تمرین ۱

Min ممکن است در s نباشد. $A(n)=\sum_{i=1}^{n}(i\times\frac{p}{n})+n(1-p)=n(1-\frac{p}{2})+\frac{p}{2}$ و w(n)=n و

B(n)=1⟶ تمرین ۲

احتمال وجود p⟵min احتمال عدم وجودmin⟵(1-p) چون این دو احتمال متمم یکدیگرند.

به علت وجود ۳ حلقه تو در تو n*n*n $w(n)=n^3$ ⟶ تمرین ۳

$\tau(n)\in\theta(n^{2/7})$⟶تمرین ٤

W(n)=n+1 و اگر N=0 B(n)=1⟶تمرین ٥

عناصر آرایه را جفت کرده و در هر جفت min و max را می یابیم که نیاز به n/2 مقایسه هست ($\frac{n}{2}$ عنصر کوچک و $\frac{n}{2}$ عنصر بزرگ)⟶ تمرین ٦

با $\frac{n}{2}$-1 مقایسه بین عنصرهای کوچک min نهایی و با$\frac{n}{2}$-1 مقایسه بین عنصرهای بزرگ Max نهایی پیدا می شود.

تعداد کل مقایسه ها برابر است با: $\frac{n}{2}+(\frac{n}{2}-1)+(\frac{n}{2}-1)=\frac{3n}{2}-2$

$\theta(1)=$ زمان مقایسه و تعویض نود i با فرزندانش $\tau(n)\le\tau\left[\frac{2n}{3}\right]+\theta(1)$ ⟶تمرین ۷

و $\tau\left[\frac{2n}{3}\right]$ = زمان اجرای heapify روی یکی از درختان نود i . زیرا حداکثر $\frac{2n}{3}$ گره در زیر درخت چپ i وجود دارد.

بدترین حالت زمانی پیش می آید که نصف سطح آخر درخت پر بار باشد. با حل رابطه $\tau(n)$ طبق قضیه Master : $\tau(n)$ =o(logn)

۱۱)پیچیدگی زمانی در بدترین حالت، حالت میانگین و بهترین حالت را برای مرتب سازی جایگذاری پایه و نسخه داده شده در تمرین ٤ (که از جست و جوی دودویی استفاده می کند)تعیین کنید.

$\tau(3^k) = 6\tau(3^{k-1})$ $\longrightarrow$ با فرض n توانی از ۳ $\quad t_k = \tau(3)^k \qquad \begin{cases} \tau(n) = 6\tau(\frac{n}{3}) \\ \tau(1) = 0 \end{cases} \qquad \theta(n^{2/7})$

$\tau(n) = c_1 6^{\log n} = c_1 6^{\log 6} \qquad \tau(3^k) = c_1 6^k \qquad t_k = c_1 6^k \qquad t_k = 6_{k-1} \longrightarrow$ بنا به قضیه $\quad t_k = 6t_{k-1}$

با استفاده از شرط $\tau(1) = 0$ شرط اولیه دوم را بدست آورده و مقادیر ثابت را محاسبه می کنیم.

$\tau(n) = c_1 6^{\log 6} \in \theta(n^{2/7})$

۱۲)یک الگوریتم خطی زمانی بنویسید که n عدد متمایز را مرتب کند. این n عدد از ۱ تا ۵۰۰ تعیین می کنند . (راهنمایی: یک آرایه ۵۰۰ عنصری به کار ببرید)

```
Void Msort (int  n, int s[1..500])
{
Int  p=[n/2]  ,  m=n-p;
Int  A=[1..p], B[1..m];
If(n>1){
Copy s[1] through s[p] to A[1] through A[p];
Copy s[p+1] through s[n] to B[1] through B[m];
Msort (p,A), Msort (m,B);
```

```
Msort(p,m,A,B,s);
}
}
```

تعداد مقایسه در بهترین حالت:

تعداد مقایسه در بدترین حالت:

الگوریتم همواره از مرتبه خطی nlong خواهد بود.

۱۳)الگوریتم A و $10n^2$ عمل اصلی و الگوریتم B 300Ln n عمل اصلی اجرا می کند به ازای چه مقداری از n کارایی الگوریتم B شروع به بهتر شدن می کند.

$300Ln\ n<10n^2$ 10n طرفین تقسیم بر $30\frac{Lnn}{n}<n$

اگر عمل اصلی در الگوریتم B بیشتر از A به طول انجامد، فقط یک مقدار بزرگتر از n وجود دارد که در آن الگوریتم B کارایی بیشتری دارد. برای $30\frac{Lnn}{n}<n$ الگوریتم B شروع به بهتر شدن می کند.

$$\lim\frac{300\ln x}{10x^2}=\lim\frac{300\ {}^{1}\!/_{x}}{10\times 2x}=\lim\frac{15}{x^2}=0 \quad \longrightarrow \text{مرتبه}$$

$x\to\infty$

۱٤)دو الگوریتم با نام های Alg1 و Alg2 برای مسئله به اندازه n وجود دارد. Alg1 در مدت n^2 میکرو ثانیه وAlg2 در nlongn ۱۰۰ میکروثانیه اجرا می شود .Alg1 را می توان با ٤ ساعت صرف وقت برنامه نویس و ۲ دقیقه زمان cpu پیاده سازی کرد. Alg2 به ۱۵ ساعت وقت برنامه نویس ٦ دقیقه زمان cpu نیاز دارد. اگر به هر برنامه نویس ساعتی ۲۰ دلار پرداخته شود و زمان cpu ، دقیقه ای ۲۰دلار ارزش داشته باشد. مسئله نمونه ای به اندازه ۵۰۰ را چند بار باید با Alg2 حل کرد، تا هزینه توجیه داشته باشد؟

هزینه Alg2	هزینه Alg1	زمان اجرا Alg2 (µs)	زمان اجرا Alg1 (µs)	دفعات اجرا
$15\times20=300$ $6\times20=120$ $300+120=420$	$4\times20=80$ $2\times20=40$ $80+40=120$	$100\times210g2=200\times10^{-6}s$	$4\mu s=4\times10^{-6}s$	2
240	120	$204800\mu s=204800\times10^{-6}s$	$65536\mu s=65536\times10^{-6}s$	256
240	120	$460800\mu s$	$262144\mu s$	512
240	120	$1024000\mu s$	$1048576\mu s$	1024

برای $n\geq1024$ هزینه Alg2 قابل توجیه خواهد بود، زیرا از نظر صرف زمان بهینه تر می شود . ولی برای $n\leq500$ هم از نظر هزینه و هم از نظر پیچیدگی زمانی به صرفه نخواهد بود و برای $n\leq500$، Alg1 بهتر و با صرفه تر می باشد.

بخش ۴-۱:

۱۵) مستقیماً نشان دهید که $F(n)=n^2+3n^3 \varepsilon \theta(n^3)$. یعنی از تعاریف O و Ω استفاده کنید، تا نشان دهید که f(n) هم در $O(n^3)$ و هم $\Omega(n^3)$ است.

$\theta(f(n)) = o(f(n)) \cap \Omega(f(n))$

$1 \rightarrow n^2+2n^3 \leq 3n^3 \quad N=0\ ,\ c=3 \rightarrow n^2+2n^3 \varepsilon o(n^3) \quad n \geq N \quad n \geq 0$

$2 \rightarrow n^2+2n^3 \geq 1n^3 \quad N=0,\ c=1 \rightarrow n^2+2n^3 \varepsilon \Omega(n^3)$ 2,1 از $\rightarrow n^2+2n^3 \varepsilon \theta(n^3)$

۱۶)با استفاده از تعاریف O و Ω نشان دهید که : $6n^2+20n \notin \Omega(n^3)$ اما $6n^2+20n \notin O(n^3)$

$6n^2+20n \leq 10n^2 \quad N=5,\ c=10 \quad n \geq 5 \rightarrow 6n^2+20n \in O(n^3)$

$6n^2+20n \geq 1 \times n^3 \quad n=9 \rightarrow \quad 666<729 \quad N=0\ ,\ c=1 \rightarrow 6n^2+20n \notin \Omega(n^3)$

۱۷)با استفاده از ویژگیهای مرتبه در بخش ۲-۴-۱ نشن دهید که :

$5n^5+4n^4+6n^3+2n^2+n+7 \varepsilon \theta O(n^5) \quad c \times g(n)+d \times h(n) \varepsilon \theta f(n) \quad h(n) \in \theta f(n)\ ,\ g(n) \in Of(n) \quad 5n^5 \in \theta(n^5) \rightarrow \quad 5n^5+4n^4 \in \theta(n^5) \rightarrow$
$5n^5+4n^4+6n^3 \in \theta(n^5) \rightarrow 5n^5+4n^4+6n^3+2n^2 \in \theta(n^5)$

$5n^5+4n^4+6n^3+2n^2+n \in \theta(n^5) \rightarrow 5n^5+4n^4+6n^3+2n^2+n+7 \in \theta(n^5)$

۱۸)فرض کنید $p(n)=a_k n^k+a_{k-1} n^{k-1}+\ldots+a_1 n+a_0$ است که در آن $a_k>0$ است . با استفاده از ویژگیهای مرتبه در بخش ۲-۴-۱، نشان دهید که :

$P(n) \in \theta(n^k) \quad p(n)=n^k(a_k+a_{k-1}n^{-1}+a_{k-2}n^{-2}+\ldots)+a_1 n+a_0 \quad p(n) \in (n^k)$

۱۹) توابع زیر را از لحاظ دسته پیچیدگی گروه بندی کنید؟

$N\ln n \quad (\log n)^2 \quad 5n^2+7n \quad n^{5/2} \quad n! \quad 2^{n!} \quad 4^n \quad n^n \quad n^n+\ln n \quad 5^{\lg n} \quad \lg(n!) \quad (\lg n)! \quad \sqrt{n} \quad e^n \quad 8n+12 \quad 10^n+n^{20}$

$\sqrt{n} \quad N\ln n \quad 8n+12 \quad (\log n)^2 \quad 5n^2+7n \quad n^{5/2} \quad n^n \quad n^n+\ln n \quad e^n \quad 4^n \quad 5^{\lg n} \quad 10^n+n^{20} \quad (\lg n)! \quad \lg(n!) \quad n! \quad 2^{n!}$

۲۰) ویژگیهای او ۲و۶و۷ مرتبه را که در بخش ۲-۴-۱ مطرح شدند، اثبات کنید.

ویژگی ۱) $f(n) \in \Omega(g(n))$ اگر وفقط اگر $g(n) \in O(f(n))$ $f(n) \geq c \times g(n)$ $g(n) \leq C \times f(n)$ **بنا به رابطه**

پاد تقارنی رابطه اثبات می شود

ویژگی ۲) $f(n) \in \theta(g(n))$ اگر وفقط اگر $g(n) \in \theta(f(n))$ $f(n) \in O(g(n)) \cap \Omega(g(n))$ $g(n) \in O(f(n)) \cap \Omega(f(n))$

بنا به خاصیت تقارن رابطه اثبات می شود $f(n) \in \theta(g(n)) \Leftrightarrow g(n) \in \theta(f(n))$

۲۰)ترتیب دسته های پیچیدگی:

ویژگی ٦) $\theta(\lg n), \theta(n), \theta(n\lg n), \theta(n^2), \theta(n^j), \theta(n^k), \theta(a^n), \theta(b^n), \theta(n!)$

$b>a>1,\ k>j>2$. اگر تابع پیچیدگی $g(n)$ در دسته ای واقع در طرف چپ دسته حاوی $f(n)$ باشد در این صورت : $g(n) \in O(f(n))$

$g(n)=\log n$ $f(n)= n^2$ $\log n \in o(n^2)$ $\log n \leq c \times n^2$ $c=1$ $\lg n \in o(n)$, $n^{10} \in o(2^n)$, $2^n \in o(n!)$

اگر $h(n) \in \theta(f(n)),\ g(n) \in O(f(n)),\ c \geq 0$ در این صورت:

$c \times g(n)+d \times h(n) \in \theta(f(n))$ $g(n)=5n$ $h(n)=3\lg n$ $f(n)=n^2$ $5n+3\lg n \in \theta(n^2)$
$g(n) \in O(f(n)) \Leftrightarrow g(n) \leq c \times f(n)$ $5n \leq c \times n^2$ $c=6$

$h(n) \times g(n)\theta(f(n)) \Leftrightarrow O(f(n)) \cap \Omega(f(n))=\theta(f(n))$ $3\lg n \leq c \times n^2$ $c=1$

۲۱) ویژگیهای بازتابش ، تقارنی و انتقالی مقایسه های مجانبی (O,Ω,θ,o) را مورد بحث قرار دهید.

۱)خاصیت بازتابی (Reflexivity): (a) $f(n) =\theta(f(n))$ (b) $f(n)=O(f(n))$ (c) $f(N)=\Omega(f(n))$

خاصیت بازتابی را o و w ندارند.

۲)خاصیت تقارنی (symmetry): این خاصیت را فقط θ دارد. (a) $f(n)=\theta(g(n)) \Leftrightarrow g(n)= \theta(f(n))$

۳)خاصیت انتقالی (transitivity):

(a) $f(n)=\theta(g(n))$ and $g(n)=\theta(h(n)) \Rightarrow f(n)=\theta(h(n))$

(b) $f(n)=O(g(n))$ and $g(n)= O(h(n)) \Rightarrow f(n)= O(h(n))$

(c) $f(n)=\Omega(g(n))$ and $g(n)= \Omega(h(n)) \Rightarrow f(n)= \Omega(h(n))$

(d) $f(n)=o(g(n))$ and $g(n)= o(h(n)) \Rightarrow f(n)= o(h(n))$

(e) $f(n)=w(g(n))$ and $g(n)= w(h(n)) \Rightarrow f(n)= w(h(n))$

۲۲)فرض کنید کامپیوتری دارید که برای حل مسئله نمونه ای به اندازه n=100 به ۱ دقیقه زمان نیاز دارد اگر یک کامپیوتر جدید بخرید که ۱۰۰۰ بار سریع تر از اولی باشد، با فرض پیچیدگی های زمانی T(n) برای الگوریتم نمونه هایی به چه اندازه را می توان در عرض یک دقیقه حل کرد؟

(a) $T(n) \in \theta(n)$ n=1000*1000=1000000 $n=10^6 \Rightarrow$ دریک دقیقه با کامپیوتر جدید

(b) $T(n) \in \theta(n^3)$ n=100 $n^3=100*100*100=1000000$ $n=10^2$

(c) $T(n) \in \theta(10^n)$ n=6 $10^n=10^6$ $n=10^6$

کامپیوتر اول n=1000 را در یک دقیقه و کامپیوتر جدید n=1000 را در $\frac{1}{1000}$ دقیقه و $n=10^6$ را در یک دقیقه با کامپیوتر جدید حل می کنیم

۲۳)قضیه ۳-۱ را اثبات کنید

$$\lim_{n\to\infty}\frac{g(n)}{f(n)}=\begin{cases} c & g(n)\epsilon\theta(f(n) \quad c>0 \text{ اگر} \\ 0 & g(n)\epsilon o(f(n)) \\ \infty & g(n)\epsilon\infty(g(n)) \end{cases}$$

G(n) f(n)=5n2-3n+4

$$\lim_{n\to\infty}\frac{n}{5n^2-3n+4}=0 \Rightarrow g(n)\in o(f(n))$$ $g(n)\le c\times f(n)$ c=1

$$\lim_{n\to\infty}\frac{\frac{n^2}{\log n}}{n\log n}=\lim_{n\to\infty}\frac{n}{\log^2 n}\longrightarrow\lim_{n\to\infty}\frac{1}{2\times\log n\times\frac{1}{n}}=\lim_{n\to\infty}\frac{n}{2\log n}=\infty$$ f(n) $\in$ g(n) nlogn $\in o(\frac{n^2}{logn})$

F(n)≤c×g(n) c=2

۲۴)درستی گزاره های زیرا را نشان دهید.

اگر سمت راست دسته باشد ⟵(g(n)$\epsilon\, o(f(n))$ ⟵اگر در طرف چپ دسته باشد.

(a) Lg n $\in$ O(n) lgn $\in$ O(n) ⟵θ(lgn) θ(n) θ(nlgn) بنا به ترتیب دسته های پیچیدگی

(b) n$\in$ O(n lg n) n $\in$ O9nlgn) بنا به ویپگی دسته های پیچیدگی

(c) n lgn $\in$ O(n^2) nlg n≤ n^2 N=0, c=1 n≥0 بنا به ویژگی دسته های پیچیدگی

(d) $2^n \in \Omega(5^{\ln n})$ $2^n \geq 5^{\ln n}$ N=8, c=1 n≥8

(e) Lg3 n $\in$ O(n0.5) g(n) $\in$ O(f(n))-Ω(f(n)) $\lg^3 n \leq n^{0.5}$, $\lg^3 n \leq (n)^{0.5}$ ونیز بنا به ترتیب دسته ای

N=4 ,C=1

تمرينات

فصل دوم

بخش ۱–۲

۱) از جستجوی دودویی (الگوریتم ۱-۲) برای جستجو به دنبال عدد صحیح **120** در لیست (آرایه) اعداد صحیح زیر استفاده کنید: عملیات را مرحله به مرحله نشان دهید.

$$\overset{Low^{S_1}}{12}\quad \overset{S_2}{34}\quad \overset{S_3}{37}\quad \overset{S_4}{45}\quad \underbrace{\overset{S_5}{51}}_{mid}\quad \overset{S_6}{82}\quad \overset{S_7}{99}\quad \overset{S_8}{120}\quad \overset{S_9 high}{134}\qquad x=120$$

$$mid=\lfloor (9+1)/2 \rfloor=5 \qquad x \succ s[mid] \qquad 120 \succ 51 \Rightarrow \text{Return location } (mid+1, high)$$

$$mid+1=5+1=\underbrace{6}_{Low} \qquad high=9 \qquad \overset{La_6}{82},\underbrace{\overset{a_7}{99}}_{mid},\overset{a_8}{120},\overset{a_9 h}{134} \qquad mid=\lfloor (9+6)/2 \rfloor=7$$

$$x \succ s[mid] \quad 120 \succ 99 \quad mid+1=7+1=\underbrace{8}_{Low} \quad h=9 \quad \underbrace{\overset{La_8}{120}}_{mid},\overset{a_9 h}{134} \quad mid=\lfloor (9+8)/2 \rfloor=8$$

$$x=s[mid] \quad 120=s[8] \qquad 120=120 \Rightarrow \text{Return mid}$$

۲) فرض کنید در حالت جستجو در یک لیست ۷۰۰ میلیون عضوی با استفاده از جستجوی دودویی (الگوریتم ۱–۲) هستیمئ. حداکثر تعداد مقایسه‌هایی که این الگوریتم باید انجام دهد تا عنصری را بیابد، یا بفهمد که آن عضو وجود ندارد، چندتاست؟

$$\log\lfloor 700000000 \rfloor+1=29+1=30 \quad w(n)=\lfloor \log n \rfloor+1 \in \theta(\log n)$$

$$700000000 \rightarrow \overset{1}{350000000},\overset{2}{175000000},\overset{3}{87500000},\overset{4}{43750000},$$
$$\overset{5}{21875000},\overset{6}{10937500},\overset{7}{5468750},\overset{8}{2734375},\overset{9}{1367187},\overset{10}{683593},$$
$$\overset{11}{341796},\overset{12}{170898}$$

85449, 42724, 21362, 10681, 5340, 2670, 1335, 667, 333, 166, 83, 41, 20, 10, 5, 2, 1.

۳) فرض کنید ما همواره یک جستجوی موفق را اجرا می‌کنیم، یعنی در الگوریتم ۱-۲ عضو X همواره در لیست S قابل یافتن باشد. الگوریتم ۱-۲ را با حذف موارد غیر ضروری بهبود ببخشید؟

```
Index location (index low, index high)
{
  Index mid;
mid = ⌊(low + high) / 2⌋;
If (x = s[mid])
   Return mid;
Else if (x < s[mid])
     Return location (low, mid − 1)
Else
     Return location (mid + 1, high)
  }
}
```

۴) نشان دهید که پیچیدگی زمانی در بدترین حالت برای جستجوی دودویی (الگوریتم ۱-۲) به صورت زیر است: که n الزاماً توانی از ۲ است.

$$w(n) = \lfloor \log n \rfloor + 1$$

راهنمایی: نخست نشان دهید دستور بازگشتی برای w(n) عبارت است از:

به ازای $n \succ 1$ $\qquad w(n) = 1 + w\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$

$$w(1) = 1$$

یکی از بدترین حالتها زمانی رخ می دهد که عدد مورد جستجوی x آخرین (بزرگترین) عنصر آرایه باشد. در این حالت آرایه باید مرتباً نصف شود تا هنگامی که در نهایت آرایه یک خانه داشته باشد. آرایه‌ای با یک خانه نیز تنها به یک عمل مقایسه نیاز دارد. لذا:

$$\begin{cases} w(n) = w(\left\lfloor \frac{n}{2} \right\rfloor)+1 \\ w(1)+1 \end{cases}$$

با توجه به قضیه اصلی:

$$\begin{cases} T(n) = aT(\left\lfloor \frac{n}{b} \right\rfloor)+Cn^K \\ T(1) \end{cases} \Rightarrow T(n) = \theta\left(n^K \left\lfloor \log_2^n \right\rfloor\right) \qquad a = b^k \qquad \text{اگر}$$

$$a=1,\ b=2,\ c=1,\ K=0 \Rightarrow a=b^K \Rightarrow 1=2^{\circ} \quad w(n)=\theta(n^{\circ}\left\lfloor \log_2^n \right\rfloor)+1 \quad w(n)=\left\lfloor \log_2^n \right\rfloor+1$$

$1)\ w(n)=\left\lfloor \log_2^n \right\rfloor+1$ برای اعداد فرد و توان غیر از ۲

$2)\ w(n)=\log_2^n+1$ برای اعداد زوج و توان ۲

۵) فرض کنید در الگوریتم ۱-۲ (خط چهارم)، تابع افراز به mid = low تغییر یابد؛ این راهبرد جستجوی جدید را توضیح دهید. کارایی این راهبرد را تحلیل کرده، نتایج را با استفاده از نمادهای مرتبه نشان دهید:

If mid = low then

If $x = s[mid]$ then return mid

Else return O;

در صورتیکه آرایه تک عنصری باشد low = high بوده و در نتیجه mid = low خواهد شد و این دو حالت دارد. اگر x در آرایه موجود باشد، همان مقدار mid بوده و اگر موجود نباشد، تابع مقدار O را برمی‌گرداند.

$w(n)=\log_2^n+1=\theta(\log n)$ توانی از ۲ $\log_2^1+1=0+1=1$ $\quad n=1,\ 2^{\circ}=1$ مقایسه

در برداری n عنصری جستجوی موفق یا ناموفق حداکثر با $O(\log_2^n)$ صورت می‌پذیرد. ولی الزاماً تعداد جستجوی ناموفق برای دو عنصر مختلف یکسان نیست.

تعداد مقایسه	بهترین حالت B(n)	حالت متوسط A(n)	بدترین حالت W(n)
جستجوی باینری	$O(1)$	$O(\log_2^n)$	$O(\log_2^n)$

٦) الگوریتمی بنویسید که لیست مرتب شده‌ای از n عنصر را با تقسیم آن به 3 لیست فرعی هریک با حدود $n/3$ عنصر جستجو کند. این الگوریتم لیست فرعی حاوی عنصر مفروض را یافته آن را به 3 لیست فرعی تقسیم می‌کند که اندازه‌ی آنها تقریباً مساوی است. این الگوریتم چندان ادامه می‌یابد که عنصر مورد نظر پیدا شود یا معلوم گردد که در لیست وجود ندارد. الگوریتم خود را تحلیل کنید و نتایج را با استفاده از نماد مرتبه نشان دهید:

Index location (index low, index high)

{

Index m_1, index

If $(Low \succ high)$

Return O;

Else {

$m = \lfloor (low + high) / 3 \rfloor$, $n = 2m$

If $(x = s[m])$

Return m;

Else if $(x = s[n])$

If $(x \prec s[m])$

Return location $(low, m-1)$;

Else if $(x \succ s[m] \,\&\, x \prec s[n])$

Return location $(m+1, n)$;

{

If $(x \succ s[n])$

Return location $(n+1, high)$

}

}

}

Return n;

۷) از روش تقسیم و حل برای نوشتن الگوریتمی استفاده کنید که بزرگترین عنصر از لیستی مرکب از n عنصر است. الگوریتم خود را تحلیل کنید و نتایج را به صورت نماد مرتبه نشان دهید:

Algorithm Maximum (low, high, max) $[l...n] \Rightarrow l = low \quad n = high$

If $low = high$ then $\max = s[low]$ آرایه تک عنصری است//

Else if $high = low + 1$ then آرایه دو عنصری است//

{

If $s[low] \prec s[high]$ then

{

$\max = s[high]$

}

Else $\max = s[low]$

}

Else

{

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

Maximum $(low, mid, l\max)$

Maximum $(mid + 1, high, r\max)$

If $l\max \succ r\max$ then $\max = l\max$

Else $\max = r\max$

}

اگر T(n) تعداد مقایسه ها برای آرایه‌ی د عنصری باشد، آنگاه:

$$T(n)=\begin{cases} T(\left\lfloor \frac{n}{2} \right\rfloor)+(\left\lceil \frac{n}{2} \right\rceil)+2, n\succ 2 \\ 1, n=2 \\ 0, n=1 \end{cases}$$

اگر فرض کنیم n توانی از 2 می‌باشد ($n=2^K$)، آنگاه:

$$T(n)=\begin{cases} 2T(\frac{n}{2})+2, n\succ 2 \\ 1, n=2 \\ 0, n=1 \end{cases} \Rightarrow \text{(با حل رابطه)} \qquad T(n)=\frac{3n}{2}-2$$

نکته: هر الگوریتمی که بتواند کوچکترین و بزرگترین عناصر آرایه‌ی n عنصر را فقط با مقایسه پیدا کند، حداقل تعداد مقایسه‌ها برابراست با T(n) که:

$$T(n)=\begin{cases} \frac{3n}{2}-2 \longrightarrow \boxed{\text{اگر n زوج باشد}} \\ \frac{3n}{2}-\frac{3}{2} \longrightarrow \boxed{\text{اگر n فرد باشد}} \end{cases}$$

بخش ۲-۲

۸) از مرتب‌سازی ادغامی (الگوریتم های ۲-۲ و ۴-۲) برای مرتب‌سازی لیست زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید:

الگوریتم ۲-۲:

$m = 8 - 4 = 4$ $\qquad$ $h = \lfloor 8/2 \rfloor = 4$

$v[h+1, n]$ $\qquad$ $u[l \ldots h]$

$m = 4 - 2 = 2$ $\qquad$ $h = \lfloor 4/2 \rfloor = 2$

$m = 2 - 1 = 1$ $\qquad$ $h = \lfloor 2/2 \rfloor = 1$

i	j	k	h	m
1	+ 2	+ 2	1	1

$\overset{S_1}{123}\ \overset{S_2}{34}\ \overset{S_3}{189}\ \overset{h\,S_4}{56}\ \underbrace{\overset{h+1\,S_5}{150}\ \overset{S_6}{12}\ \overset{S_7}{9}\ \overset{S_8}{240}}_{m=4}$

U $\quad \overset{1}{123}\ \overset{2}{\underset{h}{34}}\ \overset{3}{\underset{h+1}{189}}\ \overset{4}{56}$

V $\quad 150\ \overset{h}{12}\ \overset{h+1}{9}\ 240$

123 34	189 56	150 12	9 240

123	34	189	56	150	12	9	240

$S[K]$

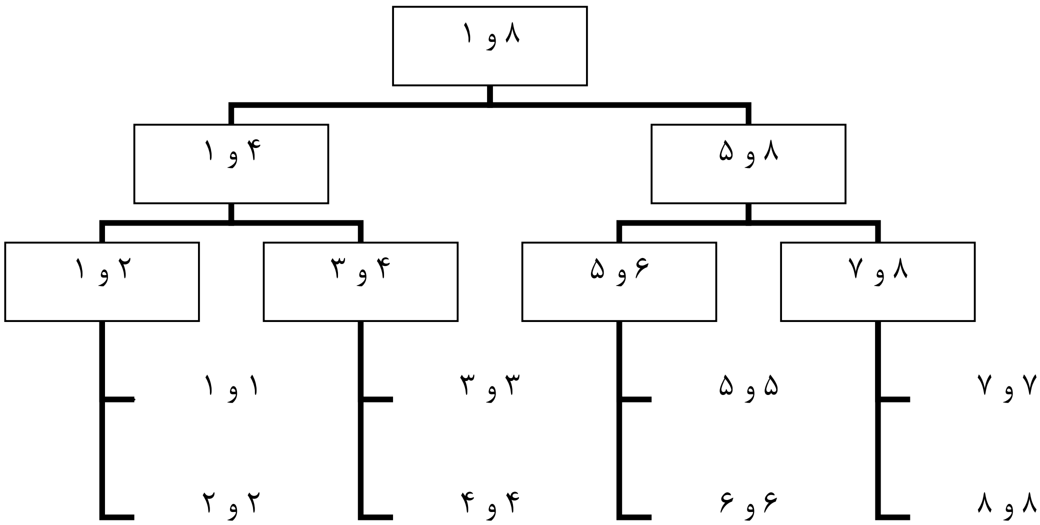
34 123	56 189	12 150	9 240

34 56 123 189	9 12 150 240

9	12	34	56	123	150	189	240

۹) برای تمرین ۸ یک درخت فراخوانی بازگشتی رسم کنید:

- زوج مقادیر داخل هر گره نشان دهنده مقادیر high و low هستند.



۱۰) برای مسئله زیر یک الگوریتم بازگشتی بنویسید که پیچیدگی زمانی در بدترین حالت برای آن بدتر از $\theta(n \log n)$ نباشد. با داشتن لیستی از n عدد صحیح مثبت متمایز، لیست را به دو لیست فرعی، هریک به اندازه‌ی $n/2$ ، افراز کنید به قسمی که اختلاف میان حاصل جمع اعداد صحیح به دو لیست فرعی حداکثر باشد. می‌توانید فرض کنید n مضربی از 2 است.

اگر (n) آزمان اجرای merge sort باشد، آنگاه با توجه به اینکه آرایه دو قسمت مساوی می‌شود (n توانی از 2 فرض شده است) و با مرتبه‌ی $\theta(n)$ می‌باشد)، می‌توان نوشت: $T(n) = 2T\left[\frac{n}{2}\right] + \theta(n)$

که با حل این رابطه $T(n) = \theta(n \log n)$ بدست می‌آید.

```
Void merge sort (low, high)
{
   If (low ≺ high)
  {
```

```
    mid = ⌊(mid + low)/2⌋;
    Merge sort (low, mid)
    Merge sort (mid+1, high)
    Merge (low, mid, high)
  }
}
```

۱۱) یک الگوریتم غیر بازگشتی برای مرتب‌سازی ادغامی (۴-۲ و ۲-۲) بنویسید:

- با فرض n توانی از 2 ($n = 2^K$):

```
Void merge sort (int n, key type S[L...n])
                    Index i, j)
   {
     For (i = n; i ⇐ 1; i = n/2)
     {
       For (j = m; j ⇐ 1; j = m/2)
     {
       Const int h = n/2, m = n/2
       Key type u[1..h], v[1..m]
       If (n ≻ 1) {
            Copy s[1] through s[h] to u[1] through u[h];
            Copy s[h+1] through s[n] to v[1] through v[m];
            }
```

}

}

Index i ، 1 تا h را به زیر آرایه‌های کوچکتر تقسیم می‌کند و Index j ، m تا n را به زیر آرایه‌های کوچکتر تقسیم می‌کند.

۱۲) نشان دهید که معادله بازگشتی مربوط به پیچیدگی زمانی در بدترین حالت برای مرتب‌سازی ادغامی (الگوریتم های ۲-۲ و ۴-۲) عبارت است از:

$$w(n) = w(\left\lfloor \frac{n}{2} \right\rfloor) + w(\left\lceil \frac{n}{2} \right\rceil) + n - 1$$

که در آن n به توانی از ۲ محدود نمی‌شود.

$n = 9 \qquad \left\lfloor \frac{9}{2} \right\rfloor = 4 \qquad \left\lceil \frac{9}{2} \right\rceil = 5 \qquad 4 + 5 = 9$

$$\underbrace{\left\lfloor \frac{n}{2} \right\rfloor}_{h} + \underbrace{\left\lceil \frac{n}{2} \right\rceil}_{m} = n \quad \Rightarrow w(n) = T(h) + T(m) + \underbrace{h + m}_{n} - 1 \qquad T(h) = \left\lfloor \frac{n}{2} \right\rfloor \qquad T(m) = \left\lceil \frac{n}{2} \right\rceil$$

$w(h,m) = h + m - 1$ زمان لازم برای ادغام $\qquad T(h) = u$ زمان مرتب‌سازی $\qquad T(m) = v$ زمان مرتب‌سازی

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

۱۳) الگوریتمی بنویسید که لیستی از n عنصر با تقسیم آن به سه لیست فرعی هریک با حدود $n/3$ عضو، مرتب‌سازی هر یک از لیست های فرعی به صورت بازگشتی و ادغامی مرتب کند. الگوریتم خود را تحلیل کنید و نتایج را با استفاده از نماد مرتبه نشان دهید:

```
Void merge sort (int n, S[1...n])
{
Const int h = ⌊n/3⌋, m = 2m , P = n − (h + m);
Key type u[1...h], v[1...m], w[1...p];
If n ≻ 1 {
Copy s[1] through s[h] to u[1] through u[h]
Copy s[h+1] through s[m] to v[1] through v[m]
Copy s[m+1] through s[n] to w[1] through w[p]
Merge sort (h,u);
Merge sort (m,v);
Merge sort (p,w);
Merge sort (h,m,p,u,v,w,s);
  }
}
```

با فرض n توانی از ۳:

$$w[n] = T(h) + T(m) + T(p) + \underbrace{h + m + p}_{n} - 1 \qquad w[n] = 3w(\frac{n}{3}) + n - 1 \qquad w[n] \in \theta(n \log n)$$

بخش ۳–۲

۱۴) برای رابطه بازگشتی زیر T(625) را بیابید:

$T(n)=7T(\frac{n}{5})+10n$ به ازای $n \succ 1$

$T(1)=1$

$T(625)=7T(125)+10(625)=(7\times 5793)+6250=46801$

$T(125)=7T(25)+10(125)=(7\times 649)+1250=5793$

$T(25)=7T(5)+10(25)=(7\times 57)+250=649$

$T(5)=7T(1)+10(5)=(7\times 1)+50=57$

۱۵) روال (solve (P, I, O زیر را در نظر بگیرید. این الگوریتم مسئله p را با یافتن خروجی (حل O) مربوط به هر ورودی I را حل می‌کند.

Procedure solve (P, I, O)

Begin

If size $(I)=1$ then

Find solution O directly

Else

Partition I into 5 inputs I_1, I_2, I_3, I_4, I_5, where

Size (I_j) : = size $(I)/3$ for $j=1,...5$;

For j: = 1 to 5 do

Solve (P, I_j, O_j)

End;

Com bin O_1, O_2, O_3, O_4, O_5 to get O for P with input I

End

End;

فرض کنید g(n) تعداد عمل اصلی لازم برای افراز و ترکیب باشد و برای نمونه‌ای به اندازه‌ی 1، عمل اصلی نداریم.

الف) یک معادله‌ی بازگشتی T(n) برای تعداد اعمال اصلی مورد نیاز جهت حل P بنویسید، هنگامیکه اندازه‌ی ورودی، n باشد.

ب) حل این معادله‌ی بازگشتی چیست اگر $g(n) \in \theta(n)$ باشد؟ (نیاز به اثبات نیست)

ج) با فرض $g(n) = n^2$، معادله‌ی بازگشتی را دقیقاً به ازای $n = 27$ حل کنید:

د) حل کلی را برای n که توانی از 3 باشد، بیابید:

الف)

$$T(n) = aT(F(n)) + g(n) \qquad \begin{cases} T(n) = 5T(\frac{n}{3}) + g(n) \\ T(1) = 0 \end{cases} \qquad n \succ 1$$

ب)

$$T(n) = aT(\frac{n}{b}) + cn^K \qquad T(n) \in \begin{cases} \theta(n^{K)}, a \prec b^K \\ \theta(n^K.\log n), a = b^k \\ \theta(n^{Log_b^a}), a \succ b^K \end{cases} \qquad 5 \succ 3^1$$

ج)

$g(n) = n^2 \ , \ n = 27 \Rightarrow g(n) = (27)^2 = 729$

$I_1 = \frac{27}{3} = 9 , \quad I_2 = 9 , \quad I_3 = 9 , \quad I_4 = 9 , \quad I_5 = 9$ solve (P, I_1, O_1), solve (P, I_2, O_2) $\ldots$ (P, I_5, O_5)

$I_1 = I_2 \ \ldots \ = I_5 = 3$

$I_1 = I_2 = \ \ldots \ I_5 = 1$

د)

$n \to$ ۳ توان $\quad n = 3^{\circ},\ 3^{1},\ \ldots\ 3^{n} \quad \begin{cases} T(n) = 5T(\frac{n}{3}) \\ T(1) = 0 \end{cases} \quad n \succ 1 \quad T(n) = 5T(\frac{3^K}{3})$

$T(n) = 5T(3^{K-1}) \quad t_K = T(3^K) \quad t_K = 5t_{K-1} = 5_{K-1} \quad t_K = c_1 5^K \quad T(n) = c_1 5^{\log n} = c_1 n^{\log 5}$

$\longrightarrow \boxed{T(n) = c_1 5^{2/5}}$

۱۶) فرض کنید در یک الگوریتم تقسیم و حل، همواره نمونه‌ای به اندازه‌ی n همواره به ۱۰ زیر نمونه به اندازه $n/3$ تقسیم شود و مراحل تقسیم و ترکیب تابع زمانی $\theta(n^2)$ باشند. یک معادله‌ی بازگشتی برای زمان اجرای T(n) نوشته، معادله‌ی T(n) را حل کنید:

با فرض n توانی از ۳:

$\begin{cases} T(n) = 10T(\frac{n}{3}) \\ T(1) = 1 \end{cases} \quad n \succ 1 \quad T(n) = 10T(\frac{3^K}{3}) + \theta(n^2)$

$T(n) = 10T(3^{K-1}) \quad t_K = T(3^K) \quad t_K = 10t_{K-1} = 10_{K-1} \quad t_K = c_1 10^K \quad T(n) = c_1 n^{\log 10} = c_1 n^1$

$T(n) = c_1.n^1 + \theta(n^n) \Rightarrow T(n) \in \theta(n^2)$

۱۷) یک الگوریتم تقسیم و حل برای مسئله برج های هانوی بنویسید. مسئله برج هانوی شامل سه میله و n دیسک به اندازه‌ی متفاوت می‌باشد.

الف) برای الگوریتم خود نشان دهید که $s(n) = 2^n - 1$ است. در اینجا s(n) نشانگر تعداد مراحل (حرکتها) و n تعداد دیسکهاست.

ب) ثابت کنید هر الگوریتم دیگری حداقل به تعداد حرکات $s(n) = 2^n - 1$ نیاز دارد.

```
Void Hanoi (int n, char A, char B, char C)
{
```

```
If (n = = 1) print F ("Mov a disk from %C to %C \n", A, C);
Else {
   Hanoi (n-1, A, C, B);
   Print F ("Mov a disk from %C to %C \n", A, C);
   Hanoi (n-1, B, A, C);
  }
}
```

$$\begin{cases} s(n) = s_{(n-1)} + s_{(n-1)} + 1 & n \geq 1 \\ s(1) = 1 & n = 1 \end{cases}$$

- حال می‌توان این رابطه بازگشتی را با Interation حل کرد:

$$s(n) = 2s(n-1) + 1 = 2(2s(n-2) + 1) + 1 = 2^2 T(n-2) + 2 + 1$$

$$\Rightarrow s(n) = 2^3 T(n-3) + 2^2 + 2 + 1 \Rightarrow s(n) = 2^{n-1} T(1) + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1$$

$$\Rightarrow s(n) = 2^{n-1} + 2^{n-2} + \ldots + 2 + 1 = \boxed{2^n - 1} \Rightarrow s(n) \in \theta(2^n)$$

می‌دانیم حاصل $a + aq + aq^2 + \ldots + aq^{n-1}$ تصاعد هندسی است و برابر است با:

$$a \frac{q^n - 1}{q - 1}$$

۱۸) هنگامیکه یک الگوریتم تقسیم و حل، نمونه‌ای از مسئله به اندازه‌ی n را به چند زیرنمونه هریک به اندازه‌ی n/c تقسیم می‌کند، رابطه‌ی بازگشتی معمولاً به صورت زیر داده می‌شود:

$$T(n) = aT(\frac{n}{c}) + g(n) \qquad \text{به ازای } n \succ 1$$

$$T(1) = d$$

که در آن g(n) هزینه‌ی فرآیندهای تقسیم و ترکیب و d یک مقدار ثابت است. فرض کنید $n = c^K$.

الف) نشان دهید :

$$T(c^K)=d\times a^K+\sum_{j=1}^{K}\left[a^{k-j}\times g(c^j)\right]$$

ب) رابطه‌ی بازگشتی را با این فرض که $g(n)\in\theta(n)$ باشد، حل کنید:

الف)

$$T(c^K)=d\times a^K+\sum_{j=1}^{K}\left[a^{k-j}\times g(c^j)\right] \qquad T(c^K)=aT(c^{K-1})+g(n)$$

$$T(c^1)=d\times a+\sum_{j=1}^{1}\left[a^0\times g(c^1)\right]=d\times a+\sum_{j=1}^{1}g(c)=d\times a+g(c)$$

$$T(c^2)=d\times a^2+\sum_{j=1}^{2}\left[a^1\times g(c^1)\right]=d\times a^2+\sum_{j=1}^{2}\left[a.g(c)\right]$$

ب)

$$\begin{cases}T(n)=aT(\frac{n}{c})+(g(n)\in\theta(n))\\ T(1)=d\end{cases} \qquad n=c^K \qquad T(n)=aT(\frac{c^K}{c}) \qquad T(n)=aT(c^{K-1})$$

$$t_k=T(c^k) \qquad t_k=at_{k-1}=a_{k-1} \qquad t_k=c_1.a^k \qquad T(n)=c_1.n^{\log a}$$

با استفاده از شرط اولیه $T(1)=d$، شرط اولیه دوم را بدست می‌آوریم و سپس مقدار ثابت c_1 را محاسبه می‌کنیم:

$$T(1)=c_1.1^{\log a} \qquad T(1)\Rightarrow c_1=d$$

بخش ۴–۲

۱۹)از مرتب سازی سریع (الگوریتم ۶–۲) برای مرتب سازی لیست زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید.

Low=pivot high

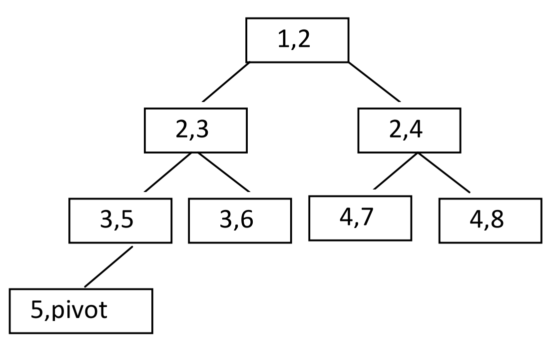
j→123 34 189 56 150 12 9 240

									j	i	piot	low	high
34	123	159	56	150	12	9	240		1	2	*s*[1]	*s*[1]	*s*[8]
34	$\boxed{123}$	189	$\boxed{56}$	150	12	9	240		2	3			
34	56	$\boxed{189}$	123	$\boxed{150}$	12	9	240		2	4			
34	56	$\boxed{150}$	123	189	$\boxed{12}$	9	240		3	5			
34	56	12	$\boxed{123}$	189	150	$\boxed{9}$	240		3	6			
34	56	12	9	$\boxed{189}$	150	123	240		4	7			
34	56	12	9	123	150	189	240		4	8			
									5				

عنصر pivot در جای اصلی خود قرار گرفت دو لیست بعدی نیز به همین روش مرتب می شوند.

لیست نهایی⇐ 9 12 34 56 123 150 189 240

۲۰) برای تمرین ۱۹ یک درخت بازگشتی رسم کنید؟ Cmp s[1] , s[2]



-زوج مقادیر داخل هر گره نشان دهنده (j,i) هایی می باشد که مقایسه می شوند و اگر شرط s[i]<pivot برقرار شد، تعویض می شوند.

۲۱) تساوی زیر را اثبات کنید:

$$\sum_{p=1}^{n} A(p-1) + A(n-p)] = 2\sum_{p=1}^{n} A(p-1)$$

این نتیجه در بحث تحلیل پیچیدگی زمانی در حالت میانگین برای الگوریتم مرتب سازی سریع به کار می رود.

$A(n)=\sum_{p=1}^{n}\frac{1}{n}A(p-1)+A(n-p)]+n-1$

$\frac{1}{n}$ = احتمال اینکه pivot point برابر p باشد

n-1 =زمان لازم برای افراز

$N=1 \Rightarrow \quad \sum_{p=1}^{1}A(1-1)+A(1-1)] = 2\sum_{p=1}^{1}(A(0)+A(0)) = 2\sum_{p=1}^{1}A(0) \qquad \surd$

$N=2 \Rightarrow$
$\sum_{1}^{2}[A(1-1)+A(2-1)+A(2-1)] = 2\sum_{p=1}^{2}A(1-1)+A(2-1) = \sum_{p=1}^{2}A(1)+A(1) = 2\sum_{p=1}^{2}A(1)$

۲۲)نشان دهید که اگر، $w(n) \le \frac{(p-1)(p-2)}{2} + \frac{(n-p)(n-p-1)}{2}$

در آن صورت : به ازای $1 \le p \le n$ $\quad w(n) \le \frac{n(n-1)}{2}$

مبنای استقرا :به ازای n=0 داریم: $w(0) = 0 \le \frac{0(0-1)}{2}$ فرض استقرا: فرض کنیم به ازای $0 \le k \le n$:
$w(k) \le \frac{k(k-1)}{2}$

گام استقرا: باید نشان دهیم که:

$$w(n) \le w(p-1) + w(n-p) + n - 1 \le \frac{(p-1)(p-2)}{2} + \frac{(n-p)(n-p-1)}{2}$$

$1 \le p \le n$ به ازای $w(n) \le \frac{n(n-1)}{2}$

P=1⇒w(n)= $\frac{(n-1)(n-2)}{2}$,p=n⇒ w(n)= $\frac{(n-1)(n-2)}{2}$

$\frac{(n-1)(n-2)}{2} \le \frac{n(n-1)}{2} \Rightarrow$

w(n)≤ $\frac{n(n-1)}{2}$

۲۳) یک الگوریتم غیر بازگشتی برای مرتب سازی سریع (الگوریتم ۶-۲) بنویسید. الگوریتم خود را تحلیل کرده و نتایج را با استفاده از نماد مرتبه نشان دهید.

Procedure quick sort (var x: Array list; left, right: interger);

Var I,j,k,l,pivot: integer;

Legin

For(k=left;k<=j-1); مرتب سازی زیر لیست }⇐

چپ}

For(l=j+1;k<=right); مرتب سازی زیر لیست }⇐

راست}

If left<right then begin

I:=left j:=right+1; pivot:=x[left];

Repeat

Repeat

I:=i+1;

Until x[i]>=pivot;

Repeat

J:=j-1;

```
Until x[j]<=pivot;
If  I<j then swap(x[i],x[j];
Unit i>=j;
Swap(x[left,x[j]);
```

{مرتب سازی زیر لیست راست}⇐

۲۴)با این فرض که مرتب سازی سریع از نخستین عنصر لیست به عنوان عنصر محوری استفاده می کند:

الف) لیستی از n عنصر (برای مثال آرایه ای متشکل از ۱۰ عدد صحیح) ارائه دهید که نشانگر سناریوی بدترین حالت باشد.

pivot→ 8 17 29 56 43 99 123 114 152 240

بدترین حالت زمانی رخ می دهد که عنصر لولا کوچکترین یا بزرگترین عنصر آرایه باشد. در این شرایط یک زیر آرایه تهی خواهد داشت.

ب) لیستی از n عنصر (آرایه با ۱۰ عدد صحیح) ارائه دهید که نشانگر سناریوی بهترین حالت باشد.

50 36 14 83 2 68 77 25 54 98

بهترین حالت زمانی رخ می دهد که عنصر لولا (محوری) عنصر میانه آرایه باشد تا آرایه را به دو زیر آرایه تقریباً مساوی افراز کند.

بخش ۵-۲:

۲۵) نشان دهید تعداد جمع های انجام شده توسط الگوریتم ۴-۱ (ضرب ماتریس ها) را می توان پس از قدری اصلاح تا حد n3-n2 کاهش داد.

$A=[a_{ij}]_{n\times n}\times B=[b_{ij}]_{n\times n}=C[c_{ij}]_{n\times n}$ تعداد ضرب ها$=n_3$

اگر بجای c[i,j]=0 بنویسیم ;c[i,j]:=A[I,1]*B[1,j] و حلقه سوم را از k:=2 شروع کنیم، آنگاه تعداد جمع n^3-n^2 خواهد بود: $\Rightarrow n^3-n^2$

تعداد جمع ها

۲۶) در مثال ۴-۲ ضرب دو ماتریس ۲*۲ به روش استراسن را ارائه کردیم. درستی این حاصل ضرب را تصدیق کنید.

$c=\begin{bmatrix} m_1+m_4-m_5+m_7 & m_3+m_5 \\ m_2+m_4 & m_1+m_3+m_2+m_6 \end{bmatrix}$ $m_1=(a_{11}+a_{22})(b_{11}+b_{22})$ $m_4=a_{22}(b_{21}-b_{11})$

$M_5=(a_{11}+a_{12})b_{22}$ $m_7=(a_{12}-a_{22})(b_{21}+b_{22})$

$m_1+m_4-m_5+m_7=a_{11}.b_{11}+a_{11}.b_{22}+a_{22}.b_{22}+a_{22}.b_{21}-a_{22}b_{11}\Rightarrow$

$-a_{11}b_{22}-a_{12}b_{22}+a_{12}b_{21}+a_{12}b_{22}-a_{22}b_{21}-a_{22}b_{22}=a_{11}\times b_{11}+a_{12}\times b_{21}$

$m_1+m_3-m_2+m_6=a_{21}\times b_{12}+a_{22}\times b_{22}$ √

$M_3+m_5=a_{11}\times b_{12}+a_{12}\times b_{22}$ √ $m_2+m_4=a_{21}\times b_{11}+a_{22}\times b_{21}$ √

۲۷) چند عمل ضرب در هنگام یافتن حاصل ضرب دو ماتریس ۶۴*۶۴ با استفاده از الگوریتم استاندارد مورد نیاز است؟

تعداد ضرب ها$=n3$ $\Rightarrow(64)^3=26144$

۲۸) چند عمل ضرب در هنگام یافتن حاصل ضرب دو ماتریس ۶۴*۶۴ با استفاده از الگوریتم استراسن، مورد نیاز است؟

تعداد ضرب ها$=n^{2.81} \Rightarrow (64)^{2.81}=(64)^{2}.(64)^{0.81}=(4096).(64)^{0.81}$

همانطور که مشاهده می شود نسبت به الگوریتم استاندارد ، الگوریتم استرانس بهبود حاصل یافته است.

۲۹) یک معادله بازگشتی برای الگوریتم اصلاح شده استرانس (توسط ساموئل وینوگراد) بنویسید که بجای ۱۸ عمل جمع و تفریق از ۱۵ عمل جمع و تفریق استفاده می کند . معادله بازگشتی را حل کرده پاسخ را با استفاده از پیچیدگی زمانی نشان داده شده در پایان بخش ۵-۲ تصدیق کنید.

منبع : کتاب براسارد و بارسلی (۱۹۸۸):

```
Void strassen(int n
                 n×n- matrixA,
```

$$\begin{cases} T(n) = 7T\left(\frac{n}{2}\right) + 15\left(\frac{n}{2}\right)^2 & n > 1 \text{ و } n \text{ توانی از } 2 \\ T(1) = 1 \end{cases}$$

```
                 n×n-matrixB,
                 n×n-matrixc)
{
If(n<= threshold
Compute  c=A×B using the standard  Algorithm;   T(n)=5n^2.81.5n^2
Else {
```

بخش ۶-۲

۳۰) از الگوریتم ۱۰-۲ (ضرب اعداد صحیح بزرگ برای یافتن حاصل ضرب 23103×1253 استفاده کنید؟

U=23,103 v=1253 n=max(5,4)=5 $m=\left\lfloor\frac{n}{2}\right\rfloor=\left\lfloor\frac{5}{2}\right\rfloor=2$ $x=231\times10^2$ y=03

w=12×102 z=53 $r=prod2(231\times10^2+03,12\times10^2+53)$

$p=prode2(231,12)\Rightarrow$ n=3 $m=\left\lfloor\frac{3}{2}\right\rfloor=1$ $23\times10^1+1$ $1\times10^1+2$

q=prod2(03,53) n=2 $m=\left\lfloor\frac{2}{2}\right\rfloor=1$ $0\times10^1+3$, $5\times10^1+3$

Return $p\times10^{2m}+(r-p-q)\times10^m+q$

$(231\times12)10^4+(231\times53+12\times3)10^2+3\times53=$

$27720000+1227900+159=28948059$

۳۱)چند عمل ضرب برای یافتن حاصل ضرب دو عدد صحیح در تمرین ۳۰ لازم است؟

۳عمل ضرب لازم می باشد. r=(x+y)(w+z) ,xw, yz

تعداد فراخوانی ها: 1)prod2(x+y,w+z) 2)prod2(x,w) 3)prod2(y,z)

۳۲) الگوریتم هایی بنویسید که عملیات زیر را اجرا کنید؟

$u\times10^m$; udivide 10^m; urem 10m;

$u\times10^m$; $mul(u,10^m)=\begin{cases} u & m=0 \\ mul(u,10^m-1)+u & m\neq0 \end{cases}$

u divide $10^m \Rightarrow D(u,10^m)=\begin{cases} 0 & u<10^m \\ D(u-10^m,10^m)+1 & a\geq10^m \end{cases}$

u rem $10^m \Rightarrow R(u,10^m)=\begin{cases} u & u<10^m \\ R(u-10^m,10^m) & u\geq10^m \end{cases}$

۳۳) الگوریتم ۹-۲ (ضرب اعداد صحیح بزرگ) را چنان اصلاح کنید که هر عدد صحیح را

الف) به سه عدد صحیح کوچکتر با n/3 رقم تبدیل کند (با فرض $n=3^k$)

ب) به چهار عدد صحیح کوچکتر با n/4 رقم تبدیل کند (با فرض $n=4^k$)

الف):

$M=\frac{n}{3}$ $\quad u=x\times10^m+y\times10^m+z$ $\quad (x=y=z)=\frac{n}{3}$ رقم

$V=p\times10^m+q\times10^m+r$ $\quad (p=g=r)=\frac{n}{3}$ رقم

$Uv=xp\times10^{2m}+xq\times10^{2m}+py\times10^{2m}+(xr+yr+pz+pr)\times10^m+zr$

X=u divide 10^m;y=u divide 10^m;z=u rem 10^m;

P=v divide 10^m; q=v divide 10^m;r=v rem 10^m;

Return(prod(x,p)+prod(x,q)+prod(p,y))×10^{2m}+(prod(x,r)+prod(y,r)+prod(p,z)+prod(p,r))×10^m+prod(z,r);

ب):

$M=\frac{n}{4}$ $\quad u=i\times10^m+j\times10^m+k\times10^m+L$ $\quad v=p\times10^m+q\times10^m+r\times10^m+s$

$u.v=(ip+iq+ir+pj+pk)\times10^{2m}+(is+jk+ks+pL+ql+rl)\times10^m+ls$

(i=j=k)=u divid 10^m; l=urem 10^m; (p=q=r)vdivide 10^m: s=v rem 10^m;

Return(prod
((I,p)+(I,q)+(I,r)+(p,j)+(p,k)))×10^{2m}+(prod((I,s)+(j,s)+(k,s)+(p,l)+(q,l)+(r,l)))×10^m+prod(l,s);

بخش ۷–۲:

۳۴) هر دو الگوریتم مرتب سازی تعویضی و سریع را روی کامپیوتر برای مرتب سازی یک لیست n عنصری اجرا کنید. مرز پایینی n را که کاربرد الگوریتم مرتب سازی سریع با سرباره آن را توجیه می کند، تعیین کنید.

با n-1*(زمان لازم برای افراز)*=32µs $\begin{cases} T(n) = T(n-1) + n - 1 & n>0 \\ T(0) = 0 \end{cases} \Rightarrow w(n) = \frac{n(n\ 1)}{2}$

فرض

زمان مرتب سازی تعویضی $\Rightarrow$ $\frac{n(n-1)}{2}\mu s$ w(n)=32×n^2 µs   W(n)=$T_{(n-1)}$+32µs

مقدار آستانه بهینه n<-0/01 *می باشد*   n<-0/015 $\frac{n(n-1)}{2}\mu s < 32 \times n^2\ \mu s$

۳۵) هر دو الگوریتم استاندارد و استراسن را برای ضرب دو ماتریس n×n $(n=2^k)$ روی کامپیوتر اجرا کنید. مرز پایینی n راکه کاربرد الگوریتم استراسن با سرباره های آن را توجیه می کند، تعیین کنید؟

با فرض16µs=زمان لازم برای تقسیم و ترکیب نمونه ای به اندازه n می باشد

N^3 = زمان الگوریتم استاندارد $\begin{cases} w(7T(\frac{n}{2})) & n>0 \\ n^3\mu s & n\le t \end{cases}$

$7(\frac{t}{2})^3$+16µs=t^3µs   w(7($\frac{n}{2}$))+16µs=n^3µs   T(n)=7T($\frac{n}{2}$) +16µs

بنابراین مقدار آشتانه بهینه ۵.۱ می   t=5.1   t=$\sqrt[3]{128}$ t^3=128   7($\frac{t^3}{8}$)+16=t^3

باشد.

۳۶) فرض کنید روی کامپیوتری، متلاشی کردن و سر هم کردن دوباره نمونه ای به اندازه n در مورد الگوریتم (۸-۲) به $12n^2\mu s$ زمان نیاز دارد. این زمان شامل زمان لازم برای انجام همه عملیات جمع و تفریق می شود. اگر ضرب دو ماتریس $n\times n$ با استفاده از الگوریتم استاندارد، $n^3\mu s$ طول بکشد، آستانه هایی را تعیین کنید که در آن باید به جای تقسیم بیشتر نمونه ، الگوریتم استاندارد را بیابید. آیا یک آستانه منحصر به فرد وجود دارد؟

$$W(n)=\begin{cases} T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 & n > t \\ n^3\mu s & n \le t \end{cases} \quad w(7(\left(\frac{n}{2}\right) \quad)+w(18\left(\frac{n}{2}\right)^2)+12n^2=n^3\mu s$$

$$7\left(\frac{t}{2}\right)^3 + 18\left(\frac{t}{2}\right)^6 + 12t^6 = t^3 \qquad t^3 = \frac{1}{168} \qquad t = \sqrt[3]{0/005}$$

مقدار آستانه بهینه $t=\sqrt[5]{0/005}$ می باشد.

تقریباً می توان گفت یک مقدار آستانه منحصر به فرد می توان یافت که در آن به جای تقسیم بیشتر نمونه ، الگوریتم استاندارد را فراخوانی کنیم. ولی t مقدار بسیار ناچیزی می باشد که حتی می توان از آن صرف نظر کرد.

بخش ۸–۲

۳۷) از روش تقسیم و حل برای نوشتن یک الگوریتم بازگشتی استفاده کنید که n! را محاسبه کند . اندازه ورودی را تعیین کنید. آیا تابع شما دارای پیچیدگی نمایی است؟ آیا این از مورد ۱ که در بخش ۸–۲ داده شده است عدول می کند؟ نخیر

```
Int   face(int  n)
{
If (n<=1)
Return  1;.
-
Else
Return (n*fact (n-1));
}
```

-T(n)=O(n):تابع دارای پیچیدگی خطی می باشد.

نمونه ای به اندازه n به دو یا چند نمونه با اندازه تقریبی N تقسیم می شود.

اندازه ورودی= $n=13=(1101)_2$

۳۸) فرض کنید در یک الگوریتم تقسیم و حل، همواره نمونه ای به اندازه n را به n زیر نمونه به اندازه n/3 تقسیم می کنیم و مراحل تقسیم ترکیب ، زمانی خطی هستند. یک معادله بازگشتی برای زمان اجرای T(n) بنویسید و این معادله بازگشتی را حل کنید. حل خود را با نماد مرتبه نشان دهید.

$$\begin{cases} T(n) = 3T\left(\frac{n}{3}\right) + n - 1 \\ T(1) - 0 \end{cases}$$

وبا فرض n توانی از ۳ $n>2$ $n=3^k$

$T(3^k)=3T(\frac{3^k}{3}) + 3^k - 1 = 3T(3^{k-1}) + 3^k - 1$ جایگذاری با $\Rightarrow T_k=T(3^k)\Rightarrow$ $Tk=3t_{k-1}+3^k-1$ بنا به قضیه $\Rightarrow t_k=c_1+c_2 3^k+c_3 k3^k \Rightarrow T(3^k)$ بجای t_k $T(3^k)=c_1+c_2 3^k+c_3 k3^k$

$T(n)=c_1+c_2\, 3^k+c_3 k3^k$ $t(n)=n\log n-(n-1)$ $T(n)\varepsilon\theta(n\log n)$

تمرينات

فصل سوم

بخش ۱–۳:

۱)تساوی ۱–۳ را که در این بخش داده شده بود، اثبات کنید.

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & 0<k<n \\ 1 & k=0 \text{ یا } k=n \end{cases}$$

طرف اول $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-1)\ldots(n-k+1)(n-k)!}{k!(n-k)!} = \frac{n(n-1)(n-2)\ldots(n-k+1)}{k!}$

طرف

دوم:

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \frac{(n-1)!}{(k-1)!\times(n-1)-(k-1)!} + \frac{(n-1)}{k!(n-1)-k)!} =$$
$$\frac{(n-1)(n-2)\ldots(n-k+1)(n-k)!}{(k-1)!(n-k)} + \frac{(n-1)(n-2)\ldots(n-k)(n-k-)!}{k!(n-1-k)!} = \frac{k[(n-1)(n-2)\ldots(n-k+1)]}{k\times(k-1)!} + \frac{(n-1)(n-2)\ldots(n-k)}{k!} =$$
$$\frac{(n-1)(n-2)\ldots(n-k+1)(k+(n-k))}{k!} = \frac{n(n-1)(n-2)\ldots(n-k+1)}{k!}$$

طرف اول =طرف دوم ← پس رابطه اثبات شد.

۲) از استقرا روی *n* استفاده کنید و نشان دهید که الگوریتم تقسیم و حل برای مسئله ضریب دو جمله ای (الگوریتم ۱–۳)، بر اساس تساوی ۱–۳، برای تعیین $\binom{n}{k}$ عبارت 1 $2\binom{n}{k}$ را محاسبه می کند.

مبنای استقرا N=1 $\binom{1}{k}$ k=1 k=0 $\Rightarrow\binom{1}{1}$ $\binom{1}{0} \Rightarrow 2\binom{1}{k} - 1 = 1$

فرض استقرا $\binom{n}{k} \rightarrow 2\binom{n}{k} - 1$

گام استقرا $\binom{n+1}{k} = 2\binom{n+1}{k} - 1$

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} = \left(2\binom{n+1}{k} - 1\right) + \left(2\left(\frac{n}{k}\right) - 1\right) \Rightarrow 2\left(\binom{n+1}{k} + \binom{n}{k} - 1\right)$$
$$= 2\binom{n+1}{k} - 1$$

۳)هر دو الگوریتم مسئله ضرب دو جمله ای (الگوریتم های ۱-۳و ۲-۳) را روی سیستم خود اجرا کنید و کارایی آنها با استفاده از نمونه های متفاوتی از مسئله بررسی کنید.

با استفاده از برنامه نویسی پویا به جای تقسیم و حل ، الگوریتمی با کارایی بسیار بالاتر بدست آوریم (الگوریتم ۲-۳) پارامترهای nو k در این الگوریتم ، اندازه ورودی نیستند. اندازه ورودی برای این الگوریتم عبارت از تعداد نمادهای لازم برای کد کردن آن ها ست. در الگوریتم ۲-۳ تعداد کل گذرها عبارت است از:

$$\frac{k(k+1)}{2}+(n-k+1)(k+1)=\frac{(2n-k+2)(k+1)}{2}\epsilon\theta(nk)$$

-در الگوریتم ۱-۳ مشکل اینجاست که در هر بار فراخوانی بازگشتی ، نمونه ها چندین بار حل می شوند. در الگوریتم پویا (۲-۳) از ویژگی بازگشتی برای حل تکراری نمونه ها به ترتیب و با شروع از نمونه کوچک تر، به جای استفاده نابجا از بازگشت ، استفاده می شود به این ترتیب هر نمونه کوچک تر فقط یک بار حل می شود. هنگامی که تقسیم و حل به یک الگوریتم ناکارآمد منجر می شود ، خوب است برنامه نویسی پویا سر کار بیاید.

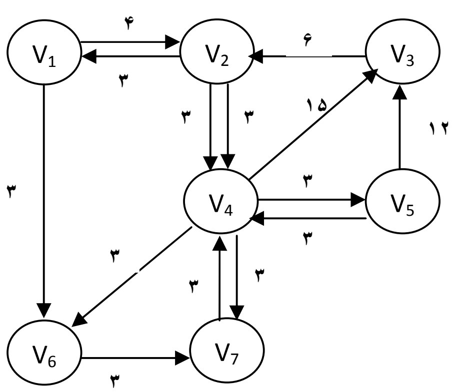
۴)الگوریتم ۲-۳ (ضریب دو جمله ای با استفاده از برنامه نویسی پویا) را چنان اصلاح کنید که فقط از یک آرایه تک بعدی با اندیس های صفر تا *k* استفاده کند.

```
Int  bin(int n , int B[0…k])          B[0…k]⇒1,(1,1),(121),(1331),(1,4,6,4,1),…
{
If(k==0││n==k)return 1;
Els return  bin(n-1,k-1);
}
```

$$\binom{n}{k}\begin{cases}\binom{n-1}{k-1} & 0<k<n\\ 1 & k=0\ or\ k=n\end{cases}$$

بخش۲–۳:

۵) از الگوریتم فلوید برای کوتاهترین مسیر ۲ (الگوریتم ۴–۳) استفاده کرده، برای گراف زیر، ماتریس D را بسازید که حاوی طول کوتاه ترین مسیرهاست و ماتریس P را بسازید که حاوی بزرگترین اندیس رئوس واسطه روی کوتاهترین مسیرها است . عملیات را مرحله به مرحله نشان دهید.



$$w = \begin{array}{c} 1\\2\\3\\4\\5\\6\\7 \end{array} \begin{bmatrix} 0 & 4 & \infty & \infty & \infty & 10 & \infty \\ 3 & 0 & 6 & 18 & \infty & \infty & \infty \\ \infty & 6 & 0 & \infty & \infty & \infty & \infty \\ \infty & 5 & 15 & 0 & 2 & 15 & 5 \\ \infty & \infty & 12 & 1 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix}$$

W=D(0)

$D^{(0)}, D^{(1)}, D^{(2)}, \ldots D^{(7)}$

عناصر قطر اصلی برابر صفر: D(1)...D(7)⇒D[1][1],D[2][2],...,D[7][7]=0

$$D^{1}_{[1][2]} = \min\left\{D^{(0)}_{[1][2]}, D^{(0)}_{[1][1]} + D^{(0)}_{[1][2]}\right\} = 4$$

$$D^{(1)} = \begin{array}{c} 1\\2\\3\\4\\5\\6\\7 \end{array} \begin{bmatrix} 0 & 4 & \infty & \infty & \infty & 10 & \infty \\ 3 & 0 & 6 & 18 & \infty & \infty & \infty \\ \infty & 6 & 0 & \infty & \infty & \infty & \infty \\ \infty & 5 & 15 & 0 & 2 & 15 & 5 \\ \infty & \infty & 12 & 1 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix} \overline{D^{(n)}}$$

$$D== \begin{array}{c} 1\\2\\3\\4\\5\\6\\7 \end{array} \begin{bmatrix} 0 & 4 & \infty & \infty & \infty & 10 & \infty \\ 3 & 0 & 6 & 18 & \infty & \infty & \infty \\ \infty & 6 & 0 & \infty & \infty & \infty & \infty \\ \infty & 5 & 15 & 0 & 2 & 19 & 5 \\ \infty & \infty & 12 & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty & 8 & \infty & \infty \end{bmatrix}$$

$$p=\begin{array}{c}1\\2\\3\\4\\5\\6\\7\end{array}\begin{bmatrix}0&0&5&2&4&0&6\\0&0&0&0&4&1&6\\2&0&0&2&4&2&4\\2&0&5&0&0&1&0\\4&4&0&0&0&4&4\\7&7&7&7&7&0&0\\4&4&5&0&4&4&0\end{bmatrix}$$

۶) از الگوریتم چاپ کوتاهترین مسیر (الگوریتم ۵-۳) برای یافتن کوتاهترین مسیر از راس V7 به راس V3 در گراف تمرین ۵ با استفاده از ماتریس pکه در آن مسئله یافتید، استفاده کنید. عملیات را مرحله به مرحله نشان دهید.

از روی ماتریس D (تمرین ۵) می توان فهمید که کوتاهترین مسیر از V7 به V3 p[7][3]

مسیر <V7,V4,V5,V3> به وزن ۲۲ می باشد. (8+2+12)

path (7,p[7][3])

⋮

Path (p[7][3],3)

در ماتریس pو p[7][3] می باشد، که بزرگترین اندیس از یک راس واسطه روی کوتاهترین مسیر از V7 به V3 می باشد.

$V_7 \xrightarrow{8} V_4 \xrightarrow{2} V_5 \xrightarrow{12} V_3 \quad \Rightarrow$ خروجی: $V_4 \quad V_5$

۷)الگوریتم چاپ کوتاهترین مسیر (الگوریتم ۵-۳) را تحلیل کنید و نشان دهید دارای پیچیدگی زمانی خطی است.

فقط متغیرهایی می توانند وارد روالهای بازگشتی شوند که مقادیر آنها قابل تغییر باشند از این رو آرایه p ورودی parth نیست. در بدترین حالت یک راس با گذر از n-2 راس باقیمانده به راس مورد نظر می رسد، و این زمانی اتفاق می افتد که کوتاهترین مسیر بین دو راس دلخواه بقیه n راس باقی را ملاقات می

کند، تا به جواب نهایی در آرایه p برسد. در نتیجه عمل اصلی ملاقات رئوس و حداکثر n-2 می باشد، پس الگوریتم دارای پیچیدگی زمانی $w(n) \in \theta(n)$ است.

۸) الگوریتم فلوید را برای مسئله کوتاهترین مسیر ۲ (الگوریتم ۴-۳) روی سیستم خود پیاده کنید و کارایی آن را با استفاده از گراف های متفاوت بررسی کنید.

یک الگوریتم ساده ولی بسیار کند برای پیدا کردن کوتاهترین مسیر آن است که تمامی مسیرهای بین گره i به j را پیدا کرده و کمترین آنها را بدست آوریم . در این صورت هنگام حرکت از گره i تعداد n-2 انتخاب داریم، پس از آن برای رفتن روی گره سوم n-3 انتخاب داریم و الی آخر. پس تعداد کل راه های ممکن $(n-2)(n-3)\times \ldots \times 2 \times 1 = (n-2)! = O(n!)$:می باشد O(n!) ولی در الگوریتم فلوید، ابتدا فقط طول کوتاهترین مسیرها را بدست آوریم . سپس قدری آن را اصلاح می کنیم تا کوتاهترین مسیر را نیز برای ما نمایش دهد. توجه کنید از آنجا که پس از محاسبه ماتریس D_K دیگر نیازی به ماتریس D_{k-1} نداریم در هر مرحله جواب D_k را در ماتریس قدیمی D_{k-1} ذخیره کرده ایم و در نهایت ماتریس D_n جواب مورد نظر است. از آنجا که الگوریتم فوق از سه حلقه تو در تو تشکیل شده است، بدیهی است که مرتبه اجرایی آن $\theta(n^3)$ است.

$T(n)=n*n*n=n3 \in \theta(n^3)$

۹) آیا الگوریتم فلوید را می توان برای مسئله کوتاهترین مسیر ۲ (الگوریتم ۴-۳) طوری اصلاح کرد که کوتاهترین مسیر از یک راس مفروض به یک راس دیگر مشخص شده در گراف را بدهد؟ پاسخ خود را توضیح دهید.

بله – با قدری اصلاح الگوریتم ٤-۳ که حداقل طول مسیر از هر گره i به هر گره j را نشان می دهد، می توان خود مسیر را نیز مشخص نمود . تابع path (الگوریتم ۵-۳) رئوس واسطه بین گره i تا j را چاپ می کند . ماتریس p یک ماتریس n*n با اندیس های ۱ تا n است که مقدار اولیه عناصر آن صفر است. بدین ترتیب p[i][j] است اگر هیچ راس واسطه ای بین iو j وجود نداشته باشد و در غیر اینصورت اگر p[i][j]=k باشد، یعنی کوتاهترین مسیر از i به j از راس k می گذرد . در واقع k آخرین راس واسطه در روی کوتاهترین مسیر است که i را به j وصل می کند.

۱۰)آیا الگوریتم فلویر برای مسئله کوتاهترین مسیر ۲ (الگوریتم ۴-۳) را می توان برای یافتن کوتاهترین مسیر در یک گراف با وزن های منفی به کار برد پاسخ خود را توضیح دهید.

الگوریتم فلوید برای یال های با وزن منفی درست کار نمی کند. الگوریتم دایجسترا برای تعیین طول کوتاهترین مسیرها از V0 به دیگر رئوس در G می باشد، این الگوریتم برای یال های غیر منفی است. در الگوریتم یافتن کوتاهترین مسیر همه زوج ها (الگوریتم فلویر) محدودیت کمتری لازم است. در الگوریتم فلوید فقط نیاز داریم که گراف هیچ سیکلی با طول منفی نداشته باشد، چرا که اگر داشتن سیکل با طول منفی برای G مجاز باشد، آنگاه کوتاهترین مسیر بین هر دو راس در این سیکل طول ∞- دارد.

بخش ۳–۳:

۱۱) یک مسئله بهینه سازی بیابید که در آن اصل بهینگی صدق نکند و بنابراین حل بهینه با استفاده از برنامه نویسی پویا قابل حصول نباشد. پاسخ خود را توضیح دهید.

در مسئله یافتن طولانی ترین مسیر بین دو راس اصل بهینگی صادق نیست و نمی توان آن را از طریق برنامه ریزی پویا حل کرد. مسئله طولانی ترین مسیر را به مسیرهای ساده ای محدود می کنیم ، زیرا با یک چرخه می توان ، همواره با عبور های مکرر از چرخه ، مسیرهایی با طول دلخواه ایجاد کرد. به عنوان مثال در گراف زیر ، طولانی ترین مسیر ساده (مسیر بهینه) از a به d مسیر <a,c,b,d> می باشد، ولی زیر مسیر <a,c> یک مسیر بهینه (طولانی ترین) از a به c نیست و مسیر طولانی ترین به صورت <a,b,c> می باشد.

۱۲) ترتیب بهینه را برای تعیین حاصلضرب A1*A2*A3*A4*A5 بیابید، اگر :

A_1:(10*4) A_2:(4*5) A_3:(5*20) A_4:(20*2) A_5:(2*50)

قطر اصلی برابر ۰ می شود: $A_{11}=A_{22}=A_{33},A_{44}=A_{55}=0$ $Mij=min(mik+mk+1j+ri-1\times r_k\times r_j$

یک قطر بعد از قطر اصلی را در ماتریس ایجاد می کنیم:

$$\begin{array}{c} 1\\2\\3\\4\\5 \end{array}\begin{bmatrix} 0 & 200 & 1200 & 320 & 1320 \\ & 0 & 400 & 240 & 720 \\ & & 0 & 200 & 700 \\ & & & 0 & 2000 \\ & & & & 0 \end{bmatrix}$$

$M12=min(m11+m22=10\times4\times5)=200(k=1)$

$M23=min(m22+m33+4\times5\times20)=400$ (k=2)

$M34=min(m33+m44+5\times20\times2)=200$ (k=3)

$M45=min(m44+m55+20\times2\times50)=2000$ (k=4)

دو قطر بعد از قطر اصلی را در ماتریس ایجاد می کنیم:

۱۳)الگوریتم حداقل ضرب ها (الگوریتم ۶-۳) و الگوریتم چاپ ترتیب بهینه (الگوریتم ۷-۳) را پیاده سازی و کارایی آن ها را با استفاده از نمونه های متفاوتی از مسئله بررسی کنید.

در الگوریتم (٦-٣) ابعاد n ماتریس یعنی مقادیر d0تا dnتنها ورودیهای الگوریتم هستند . ماتریس ها خودشان ورودی نیستند، زیرا مقادیر آنها به مسئله ربطی ندارد. آرایه p که توسط الگوریتم تولید شد، برای چاپ ترتیب بهینه (الگوریتم ۷-۳) به کار می رود، به عنوان عمل اصلی در نظر می گیریم. الگوریتم (٦-٣) دارای پیچیدگی $\theta(n^3)$ می باشد.

در الگوریتم (۷-۳) ورودی ها عدد صحیح و مثبت n و آرایه P (که توسط الگوریتم (٦-٣) تولید می شود) می باشند. طبق قراردادی که برای روال های بازگشتی داشتیم ، P,n ورودیهای order نیستند، بلکه ورودی الگوریتم هستند. الگوریتم (۷-۳) دارای پیچیدگی $T(n)=\theta(n)$ می باشد زیرا آرایه p را خود ایجاد نکرده و آن را از الگوریتم (٦-٣) دریافت می کند.

۱۴) نشان دهید که یک الگوریتم تقسیم و حل بر اساس تساوی ۵-۳ دارای پیچیدگی زمانی نمایی است.

$$\begin{cases} M[i][j] = \min imum(M_{[i][k]} + M_{[k+1][j]} + d_{i-1}d_k d_j) & i < j \\ M[i][j] = 0 & i = j \end{cases}$$

به خاطر عدم وجود حلقه ، قطر اصلی برابر صفر خواهد بود

$$\begin{cases} M[i][j] = A_j \text{ تا } A_i & \text{حداقل تعداد ضرب هاي لازم براي } , i < j \\ M[i][j] = 0 \text{ و } & i = j \Rightarrow \text{(تعداد ضربهاي لازم براي ايجاد يك ماتريس تنهاي } (A_i) \text{ كه هيچ ضربي نياز ندارد)} \end{cases}$$

بدست اوردن حداقل تعداد ضربهای لازم برای Ai تا Aj خودش یک حلقه محسوب می شود، که در بدترین حالت از 100n محاسبه می شود. $(w(n)\in\theta(n))$

اگر T(n) تعداد ترتیبهای متفاوت برای ضرب n ماتریس A1,A2,…,An باشد، یک زیر مجموعه از این ترتیب ها حالتی است که در آنها A1 آخرین ماتریسی است که در مجموعه ضرب می شود، یعنی:

$$A\underbrace{(A2A3\ldots An)}_{T(n-1)}$$

An آخرین ماتریسی است که ضرب می شود، پس تعداد ترتیب های مختلف در این زیر مجموعه T(n-1) می باشد:

$T(n)T(n-1)+T(n-1)=2T(n-1)$ $\quad T(n)\geq 2T(n-1) \quad T(n)\geq 2^{n-2}$
$\Rightarrow T(n)\varepsilon\theta(2^{n})$

پس پیچیدگی زمانی نمایی است.

۱۵)تساوی زیر را اثبات کنید:

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

$$\sum_{d=1}^{n=1} i^{2}=\frac{n(n+1)(2n+1)}{6}\Rightarrow\sum_{d=1}^{n=1}(n-d)\times d=\sum_{d=1}^{n=1}(nd)-\sum_{d=1}^{n=1}d^{2}=\frac{n(n-1)(n+1)}{6}$$

$$\sum_{d=1}^{n=1}(nd)-\frac{n(n+1)(2n+1)}{6}=\frac{n(n-1)(n+1)}{6}$$

۱۶) نشان دهید برای قرار دادن کامل عبارتی حاوی n ماتریس درون پرانتزها ، به n-1 جفت پرانتز نیاز است.

برای قرار دادن دو ماتریس A1 و A2 درون پارانتزها به یک پرانتز نیاز داریم: $(A_1\times A_2)$ $\quad 2-1=1$

برای قرار دادن سه ماتریس A1,A2,A3 درون پرانتزها : $(A_1\times A_2)\times A_3)$ $\quad 3-1=2$

برای ٦ ماتریس A1 تا A6: $\quad 6-1=5$

$(A_1((((A_2\times A_3)A_4)A_5)A_6))$

و برای n ماتریس A1 تا An: $(A_1(((\ldots(A_2\times A_3)\times\ldots)\ldots)A_n))$

n-1

۱۷)الگوریتم ۷-۳ را تحلیل کنید و نشان دهید که دارای پیچیدگی زمانی خطی است.

آرایه p که توسط الگوریتم حداقل ضرب ها تولید می شود، برای چاپ ترتیب بهینه (الگوریتم ۷-۳) به کار می رود. در این الگوریتم ورودیها عدد صحیح و مثبت n و آرایه p می باشند . طبق قراردادی که برای روالهای بازگشتی داریم، p و n ورودیهای order نیستند، بلکه ورودی الگوریتم هستند. الگوریتم ۷-۳ دارای پیچیدگی $T(n)=\theta(n)$ خطی می باشد. (زیرا آرایه P را خود ایجاد نکرده و آنرا از الگوریتم ٦-۳ دریافت می کند)

۱۸) الگوریتم کارآمدی بنویسید که ترتیب بهینه را برای ضرب n ماتریس A1×A2…×An بیابید که در آن ابعاد هر یک از ماتریس ها 1*1و d*1*1یا d*dاست و d یک عدد صحیح مثبت است. الگوریتم خود را تحلیل کنید و نتایج را با استفاده از نماد مرتبه نشان دهید.

For i:=1 to n do

ماتریس مربع $M_{[d]\times[d]} \Rightarrow i=j \Rightarrow$

C[I,i]:=0

$M_{11},M_{22},\ldots,M_{dd}$

For L:=2 to n do

For i:=1 to n-L+1 do

{

J:=i+L-1

$A_1(2\times2),A_2(2\times2),A_3(2\times2)$

$C[i,j]:=\min(c[i,k]+c[k+1,j]+d_i*d_k*d_j)$

$p=2\times2\times2+2\times2\times2=23+23=16$

}

$d\times d\times..\times d+d\times d\times\ldots\times d=d^n+d^n+\ldots d^n$

تا n-1

n=تعداد ماتریس ها

$$\sum_{i=2}^{n}\sum_{i=1}^{n-1+1}\sum_{k=i+1}^{j}2=\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}2(j-i)\quad\xrightarrow{j=i+l-1}\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}2(i+L-1-i)$$

مجموع خواندن C:

$$=\sum_{i=2}^{n}\big(2(l-1)(n-l+1-1+1)\big)=\sum_{L=2}^{n}2(L-1)(n-L+1)$$

درایه های

تمرينات

فصل چهارم

بخش ۱-۴:

۱)نشان دهید روش حریصانه همواره یک حل بهینه برای مسئله بقیه پول می یابد اگر سکه ها بصورت $D^0,D^1,D^2,\ldots,D^i$ به ازای $i>0$ و $D>0$ باشد.

$D=1\,,\, i=6 \Rightarrow 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$

$D=2\,,\, i=6 \Rightarrow 1 \quad 1 \quad 2 \quad 4 \quad 8 \quad 16 \quad 32 \quad 64$

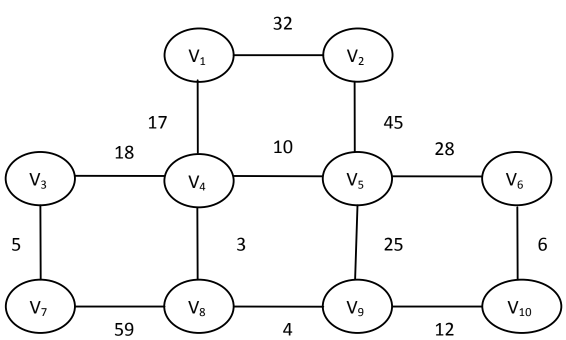
$D=3\,,\, i=6 \Rightarrow 1 \quad 1 \quad 3 \quad 9 \quad 27 \quad 81 \quad 243 \qquad 729$

$\vdots$

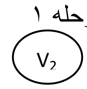
$D=n\,,\, i=6 \Rightarrow 1 \quad 1 \quad n^1 \quad n^2 \quad n^3 \quad n^4 \quad n^5 \quad n^6$

اگر بقیه پول توانی از i باشد (D^0 تا D^i) همواره روش حریصانه یک جواب منحصر به فرد و بهینه ارائه می کند . اگر بقیه پول از مجذور کامل هر عددی، یک واحد بیشتر باشد، تنها با افزودن ۱($i=0$) راه حل بهینه خواهد داد. چون تنوع سکه ها با افزایش i بیشتر می شود، الگوریتم حریصانه تقریباً پیوسته جواب بهینه را ارائه می دهد. مثلاً برای مثال کتاب یعنی عدد ۱۶ بجای ۱۲ با ۱۴ تا سکه ۱ سنتی ، سکه ۱۶ سنتی را بر می گزینید.

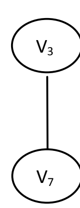
۲)از الگوریتم پریم (الگوریتم ۱-۴) برای یافتن درخت پوشای کمینه گراف زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید. از راس دلخواه V_3 شروع می کنیم.



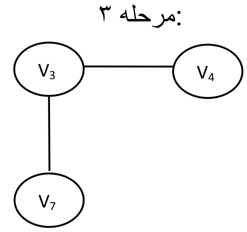
مرحله ۱:



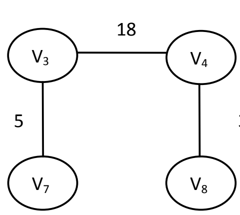
مرحله ۲:



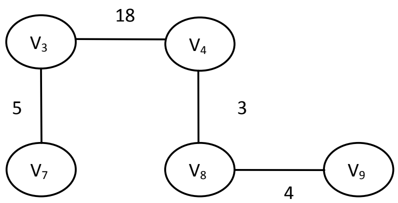
مرحله ۳:



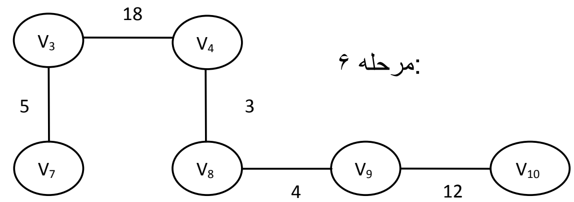
مرحله ۴:



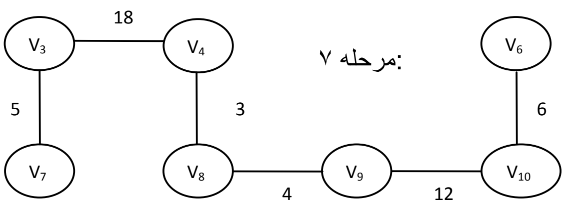
مرحله ۵:



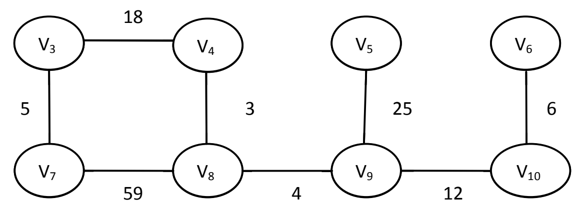
مرحله ۶:



مرحله ۷:

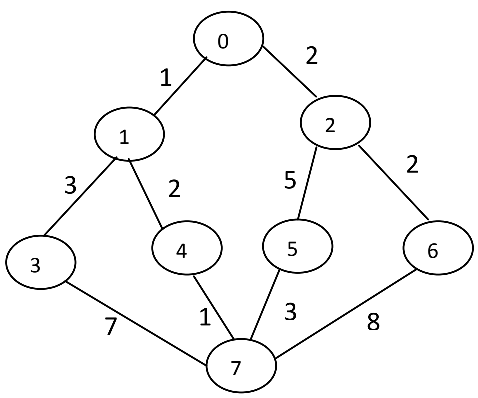


مرحله ۸:

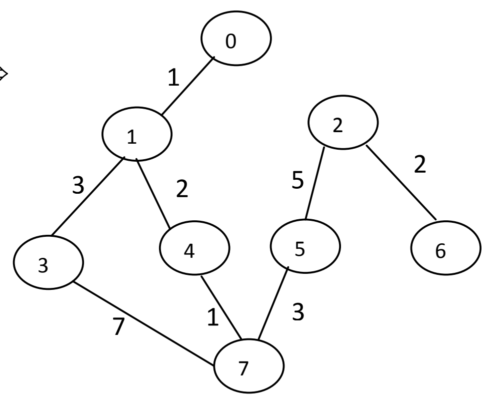


۳)گرافی رسم کنید که بیش از یک درخت پوشای کمینه دارد.

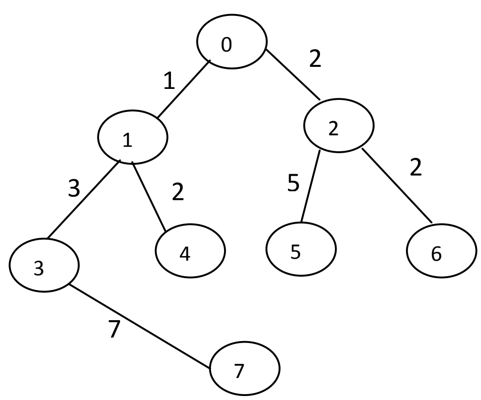
گراف اولیه



⇒



پوشايdfs



پوشايbfs

۴) الگوریتم پریم (الگوریتم۱-۴) را روی سیستم خود پیاده کنید و کارایی آن را با استفاده از گراف های متفاوت مطالعه کنید.

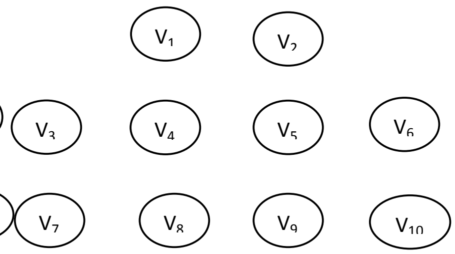
اگوریتم پریم را به سادگی به کمک ماتریس همجواری W پیاده سازی می کنیم . از آنجا که در الگوریتم پریم هر گره با گره های قبلی مقایسه می شود و نیز حلقه repeat ، به تعداد (n-1) بار تکرار می شود،پیچیدگی زمانی $T(n)=2(n-1)(n-1)\in\theta(n^2)$ می باشد . الگوریتم پریم همواره یک درخت پوشای کمینه تولید می کند . در یک گراف کامل K_n با د راس به تعداد n^{n-2} درخت پوشا وجود دارد و بدیهی است کهالگوریتم پوشا همواره از این بین یکی از حالتهای کمینه را انتخاب می کند.

۵)از الگوریتم کروسکال (الگوریتم ۲-۴) برای یافتن درخت پوشای کمینه گراف تمرین ۲ استفاده کنید. عملیات را مرحله به مرحله نشان دهید.

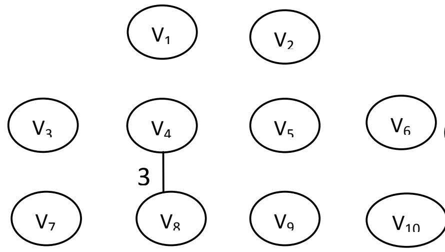
۱)یالها بر حسب طول مرتب می شوند:

(V4,V8)3 (V8,V5)4 (V3,V7)5 (V6,V10)6 (V4,V5)10
(V9,V10)12 (v1,V4)17 (V3,V4)18 (V5,V9)25
(v5,V6)28 (V1,V2)32 (V2,V5)45 (V7,V8)59

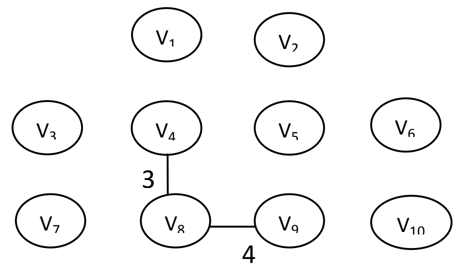
۲)مجموعه هاي مجزا ساخته مي شوند:



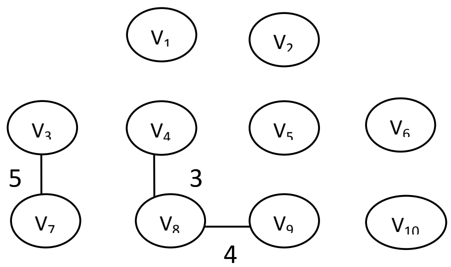
۳)يال (V4,V8) انتخاب مي شود:



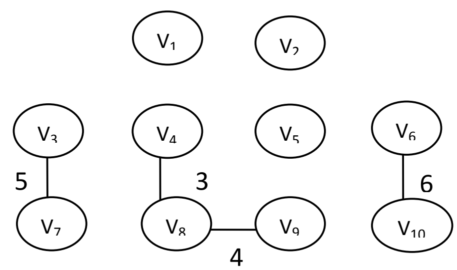
۴)يال (v8,V9) انتخاب مي شود



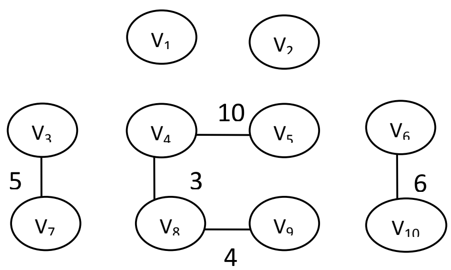
۵)يال (v3,V7) انتخاب مي شود



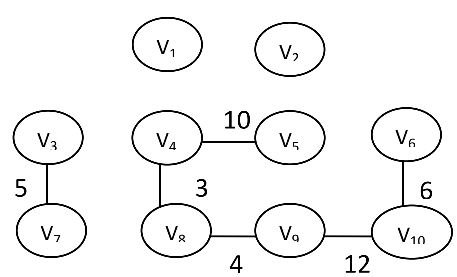
6)يال (v6,V10) انتخاب مي شود



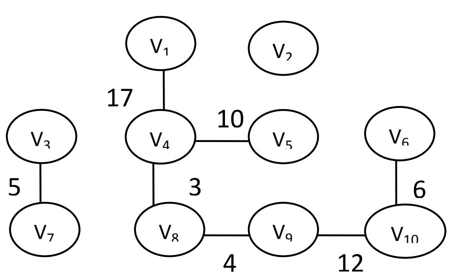
7)يال (v4,V5) انتخاب مي شود



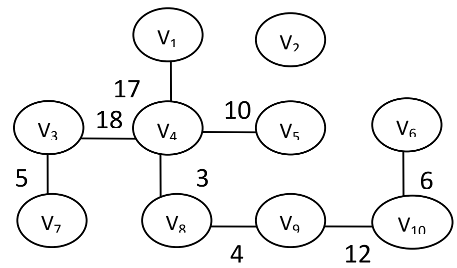
8)يال (v9,V10) انتخاب مي شود



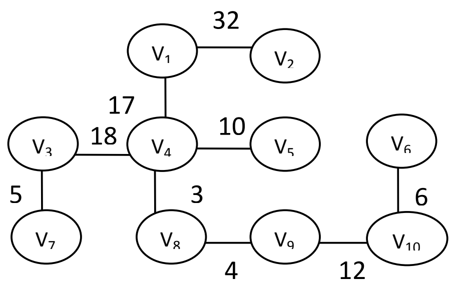
9)يال (v1,V4) انتخاب مي شود



10)يال (v3,V4) انتخاب مي شود



11)يال (v1,V2) انتخاب مي شود



۶) الگوریتم کروسکال (الگوریتم ۲-۴) را روی سیستم خود پیاده کنید و کارایی آن را با استفاده از گراف های متفاوت بررسی کنید.

این الگوریتم برای یافتن درخت پوشای کمینه یک گراف به کار می رود . این الگوریتم همواره به حل بهینه می انجامد و همواره یک درخت پوشای کمینه ایجاد می کند. در این الگوریتم ابتدا یالها از کمترین وزن به بیشترن وزن مرتب می گردند ، سپس یالها به ترتیب انتخاب شده و اگر یالی ایجاد حلقه کند، کنار گذاشته می شود. عملیات هنگامی خاتمه می یابد که تمام راس ها به هم وصل شوند یا اینکه تعداد یال های موجود در F برابر n-1 می شود که n تعداد راس هاست الگوریتم کروسکال در هر زمان یک لبه از درخت پوشای حداقل هزینه را می سازد. مجموعه لبه های انتخاب شده در الگوریتم کروسکال در هر مرحله یک جنگل را تشکیل می دهند. الگوریتم کروکسال دارای پیچیدگی $\theta(\text{elge})$ می باشد:

پیچیدگی الگوریتم کروسکال $\Rightarrow \theta(\text{elge})\begin{cases}\theta(nlgn) & \text{براي گراف خلوت}\\ \theta(n^2lgn) & \text{براي گراف شلوغ}\end{cases}$

۷)آیا تصور می کنید برای یک درخت پوشای کمینه امکان دارد که دارای چرخه باشد یا خیر؟ برای پاسخ خود دلیل بیاورید.

نخیر- درخت پوشا (spanning tree) برای یک گراف G، یک زیر گراف متصل می باشد که حاوی همه راس های گراف G بوده و همچنین یک درخت است. به عبارت دیگر درخت پوشا، شامل همه

رئوس و برخی یال های گراف است به نحوی که متصل بوده و چرخه نیز ندارد. در درخت پوشا اگر یالی ایجاد حلقه کند، کنار می گذاریم، چون درختی که چرخه داشته باشد ، گراف است

۸)فرض کنید در شبکه کامپیوترها هر دو کامپیوتری را می توان به هم متصل کرد. با داشتن هزینه تقریبی هر اتصال ، آیا باید از الگوریتم ۱-۴ (پریم) یا الگوریتم ۲-۴(کروسکال) استفاده شود؟ پاسخ خود را توضیح دهید.

اگر گرافی یالهای کمی دارد بهتر است از روش کروسکال و اگر یالهای زیادی دارد بهتر است از روش پریم استفاده کنیم . پس اگر تعداد کامپیوترهای شبکه کم باشد الگوریتم کروسکال ولی تعداد کامپیوترها زیاد باشد. الگوریتم پریم کارایی خواهد داشت. برای شبکه ای که تعداد کامپیوترهای آن (m) نزدیک به کرانه پایینی این حدود باشد (شبکه ای که بسیار متراکم است) ، الگوریتم کروسکال $\theta(nlgn)$ است، یعنی الگوریتم کروسکال باید سریع تر باشد . ولی برای شبکه ای که تعداد کامپیوترهای آن نزدیک به کرانه بالایی باشد (شبکه ای که بسیار متصل باشد) الگوریتم کروسکال ، $\theta(n^2lgn)$ است، یعنی الگوریتم پریم باید سریع تر باشد.

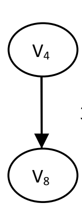
الگوریتم پریم: $T(n)=\theta(n2)$

الگوریتم کروسکال : $w(m,n)\in \theta\ (mlg\ m)\ ,\ \ w(m,n)\in \theta\ (n^2lgn)$

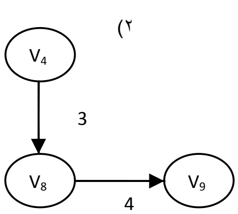
۹)از الگوریتم دیکسترا (الگوریتم ۳-۴) برای یافتن کوتاهترین مسیر از راس V_4 به همه رئوس دیگر در گراف تمرین ۲ استفاده کنید. عملیات را مرحله به مرحله نشان دهید. فرض کنید هر یال بدون جهت ، دو یال جهت دار با وزن یکسان را نشان می دهد.

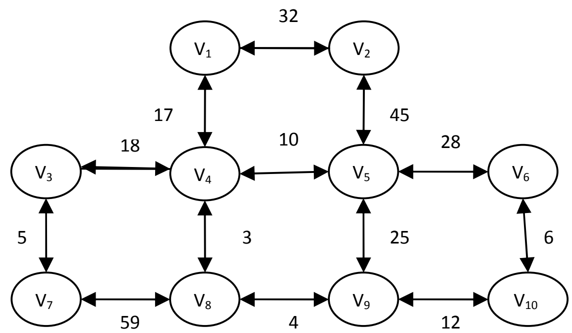
محاسبه کوتاهترین مسیر از V_1

۱)

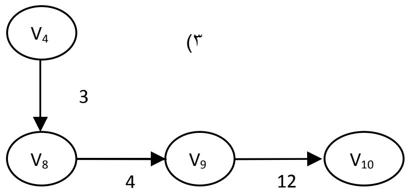


۲)

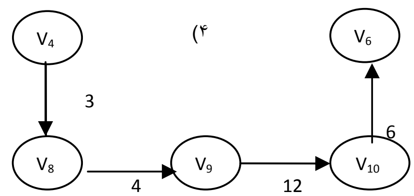



V_1 V_2 V_3 V_4 V_5 V_6 V_7 V_8 V_9 V_{10}
32
17
45
18
10
28
5
3
25
6
59
4
12

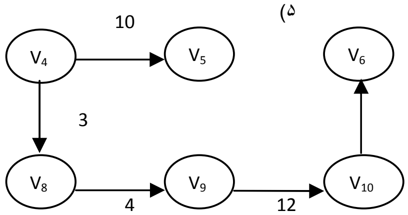
۳)


V_4 V_8 V_9 V_{10}
3
4
12

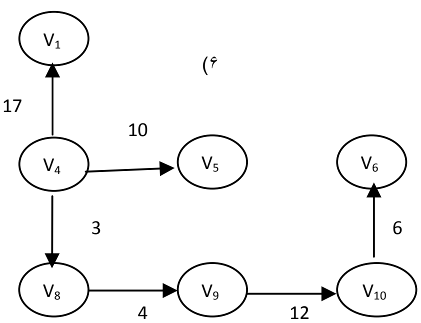
۴)


V_4 V_8 V_9 V_{10} V_6
3
4
12
6

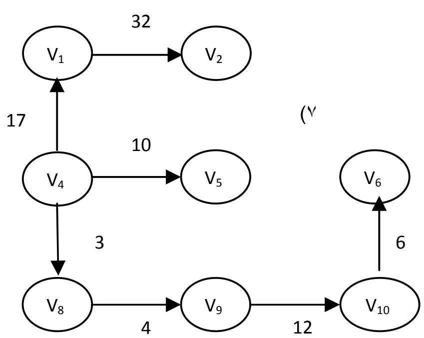
۵)


V_4 V_5 V_8 V_9 V_{10} V_6
10
3
4
12

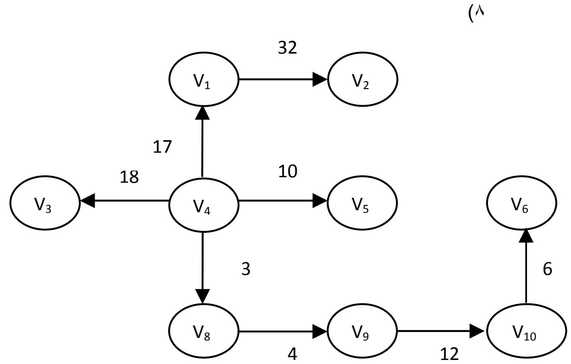
۶)


V_1 V_4 V_5 V_6 V_8 V_9 V_{10}
17
10
3
6
4
12

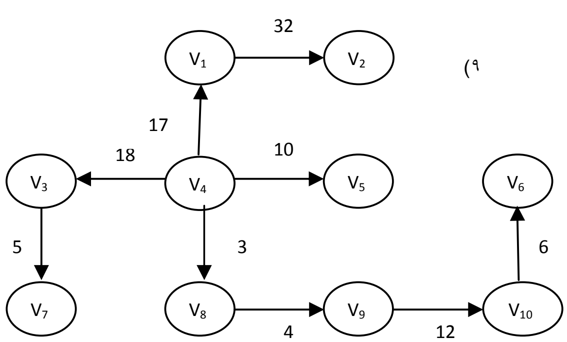
۷)


V_1 V_2 V_4 V_5 V_6 V_8 V_9 V_{10}
32
17
10
3
6
4
12

۸)


V_1 V_2 V_3 V_4 V_5 V_6 V_8 V_9 V_{10}
32
17
18
10
3
6
4
12

۹)


V_1 V_2 V_3 V_4 V_5 V_6 V_7 V_8 V_9 V_{10}
32
17
18
10
5
3
6
4
12

۱۰)الگوریتم دیکسترا (الگوریتم ۳-۴) را روی کامپیوتر خود پیاده کنید و کارایی آن را با استفاده از گراف های متفاوت بررسی کنید.

الگوریتم دیکسترا که به نام کوتاهترین مسیر تک منبع (single shortest path) نیز معروف است، مشابه الگوریتم پریم می باشد. از آنجا که در الگوریتم فوق در هر بار فاصله هر گره با گره های قبلی مقایسه می شود، الگوریتم از مرتبه $\theta(n^2)$ می باشد، که n تعداد رئوس گراف است. این الگوریتم همواره کوتاهترین مسیر را می دهد. الگوریتم دیکسترا برای گراف بدون جهت نیز قابل استفاده است. الگوریتم دیکسترا را می توان با هرم (heap) یا هرم فیبوناچی پیاده سازی کرد. پیاده سازی هرمی آن به θ (e lgn) و پیاده سازی هرم فیبوناچی آن به θ (e+ lgn) زمان نیاز دارد که n تعداد رئوس و e تعداد یال هاست.

۱۱)الگوریتم دیکسترا را طوری اصلاح کنید که طول کوتاهترین مسیرها را محاسبه کند. الگوریتم اصلاح شده را تحلیل کرده نتایج را با استفاده از نماد مرتبه نشان دهید.

در هر مرحله گره ای که کمترین فاصله تا مبدا را دارد به مجموعه S اضافه می شود. آرایه D برای ثبت طول کوتاهترین مسیر تا لحظه کنونی مورد استفاده قرار می گیرد. زمانیکه تمامی گره ها به S افزوده شوند، آرایه D طول کوتاهترین مسیر را برای تمامی گره ها نشان می دهد.

```
Dijkstra( )
{
S={1};                                    T(n)∈θ(n²)
For(i=2  to  n)
D[i]=c[1,i]:
For (i=1  to  n-1)
{
Choose  a  vertex  w in   V-S such that D[w] is minimum;
Add w to s;
```

For (each vertex V in V-s)

D[V]=min(D[V], D[w]+C[W,V]   ماتریسC ، ماتریس هزینه است

۱۲) آیا الگوریتم دیکسترا (الگوریتم ۳-۴) را می توان برای یافتن کوتاهترین مسیر در گرافی با وزن های منفی به کار گرفت؟

الگوریتم دیکسترا برای تعیین طول کوتاهترین مسیرها از V0 به دیگر رئوس در G می باشد. این الگوریتم برای یال های غیر منفی است. این الگوریتم برای گرافی درست کار می کند که یال منفی نداشته باشد. الگوریتم دیکسترا برای گراف های جهت دارو بدون جهت به شرطی که تمامی یال ها غیر منفی باشند کارایی دارد.

الگوریتم بلمن فورد تعمیم یافته الگوریتم دیکسترا می باشد که گراف می تواند وزن های منفی یا مثبت داشته باشد، ولی همانند الگوریتم فلوید نباید دور منفی داشته باشد. این الگوریتم در کتاب آقای قلی زاده شرح داده شده است.

۱۳)از استقرا استفاده کنید و درستی الگوریتم دیکسترا (الگوریتم ۳-۴) را اثبات کنید.

الگوریتم دیکسترا همواره کوتاهترین مسیر را ایجاد می کند.

اثبات: برای اینکه نشان دهیم مجموعه F پس از هر بار تکرار حلقه Repeat امید بخش است از استقرا استفاده می کنیم.

مبنای استقرا : مجموعه ϕ امید بخش است.

فرض استقرا: پس از یک بار تکرا حلقه Repeat ، مجموعه یالهای بدست آمده (یعنیf)امید بخش است.

گام استقرا: باید نشان دهیم مجموعه F∪{e} که در آن e یال انتخاب شده در تکرار بعدی است، امید بخش می باشد. چون یال e که در تکرار بعدی انتخاب می شود دارای طول کمینه است و یک راس از V-f متصل می کند، بنا به لم ۱-٤،F∪{e} امید بخش است. استقرا کامل می شود.

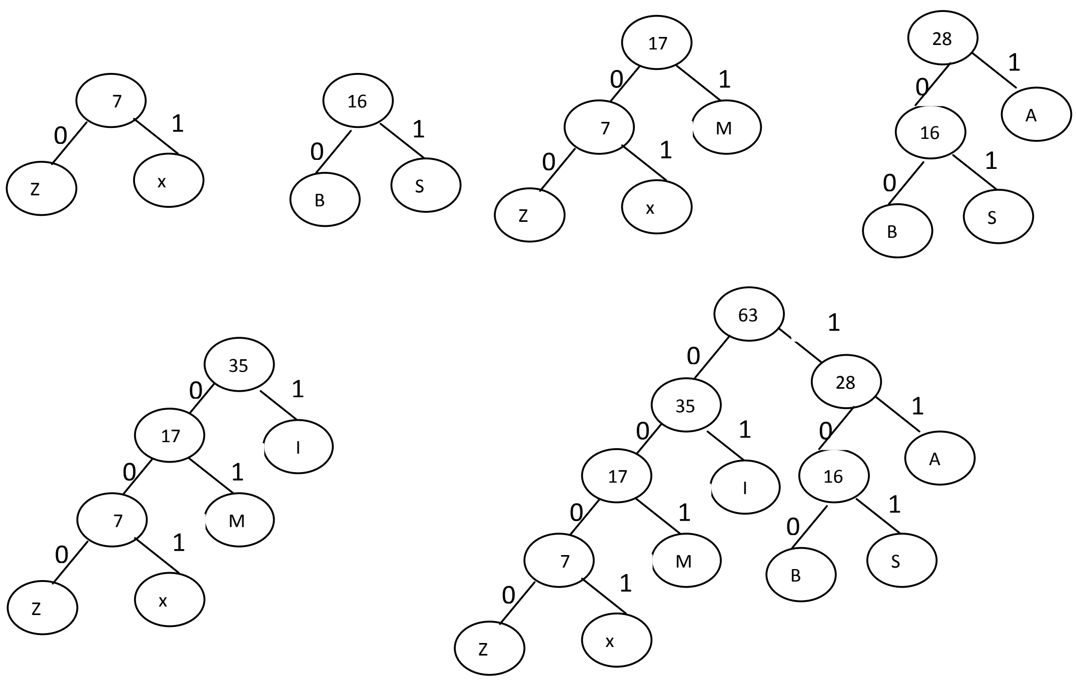
$F \cup \{e\} \subseteq (f \cup \{\acute{e}\} - \{\acute{e}\}$ $F \cup \{e\} \subseteq \acute{f}$ طبق اثبات استقرایی مجموعه نهایی یالها امید بخش است . چون این مجموعه شامل یال های یک درخت پوشا می شود، آن درخت باید یک درخت پوشای کمینه باشد. (به عنوان مثال درخت پوشای تمرین ۹ که کوتاهترین مسیر را ایجاد می کند)

بخش ۴-۴:

١٤)با استفاده از الگوریتم هافمن ، برای حروف جدول زیر یک کد پیشوندی دودویی بهینه ایجاد کنید

حروف: Z X S M I B A

فراوانی: 2 5 9 10 18 7 12



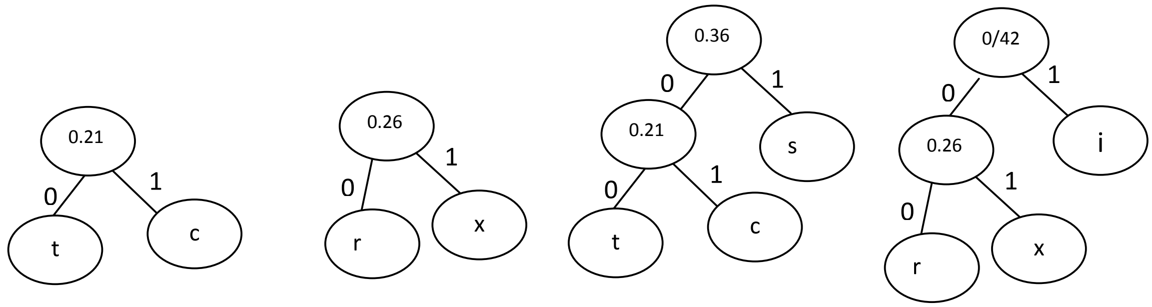
Z=0000 x=0001 M=001 I=01 B=100

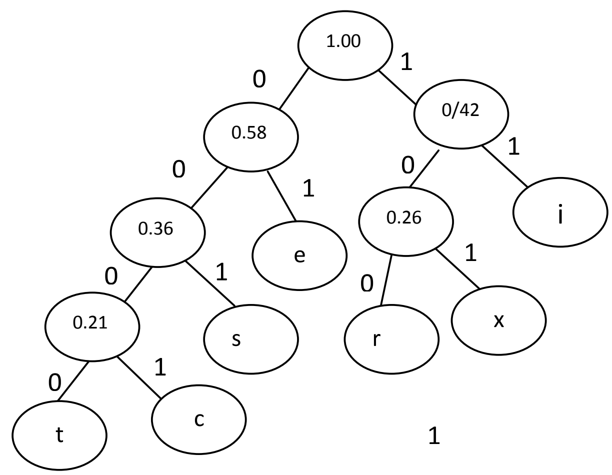
S=101 A=11

۱۵) با استفاده از الگوریتم هافمن برای حروف جدول زیر یک کد پیشوندی دودویی بهینه ایجاد کنید.

حروف: x t s r i e c

فراوانی: 0.14 0.10 0.15 0.12 0.16 0.22 0.11





t=0000 c=0001 s=001

e=01 r=100 x=101 i=11

۱۶) با استفاده از کد دودویی تمرین ۱۴، هر یک از رشته های بیتی زیر را رمز گشایی کنید.

الف) $01100010101010 \Rightarrow$ IBIIII35

ب) $1000100001010 \Rightarrow$ BIZS35

ج) $11100100111101 \Rightarrow$ ABBAAI

د) $1000010011100 \Rightarrow$ BMMA17

۱۷) هر یک از کلمات زیر را با استفاده از کد دودویی تمرین ۱۵ کد گذاری کنید.

الف) rise $\Rightarrow$ 1001100101

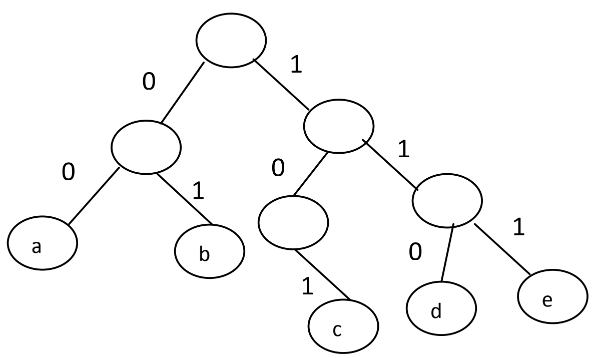
ب) exit $\Rightarrow$ 01101110000

ج) text $\Rightarrow$ 0000011010000

د) exercise $\Rightarrow$ 0110101100000011100101

۱۸) کد مربوط به a,b,c,d,e به صورت زیر مشخص شده اند ، که در آن x,y,z در صفر و یک هستند. X,y,z را طوری تعیین کنید که کد حاصل ، کد پیشوندی باشد.

a:00 b:01 C:101 d:x10 e:yz1 x=1 y=1 z=1



تمرینات

اضافی

۱)در حال حاضر می توانیم مسئله نمونه ای بااندازه ۱۰۰ را در عرض ۱ دقیقه با استفاده از الگوریتم A که $\theta(2^n)$ است، حل کنیم . به زودی باید مسائل با اندازه دو برابر را در همین ۱ دقیقه حل کنیم. آیا به یک کامپیوتر سریع تر و گرانتر نیاز داریم؟

الگوریتمی مانند B نیاز است که دارای پیچیدگی زمانی خطی بوده و نسبت به $\theta(2^n)$ که نمایی است، کارایی بهتری داشته باشد و قادر به حل مسائل با اندازه دو یا چند برابر الگوریتم A و در زمان ۱ دقیقه برابر با زمان A باشد.

۲)پیچیدگی زمانی T(n) برای حلقه های تو درتویی زیر چیست؟ فرض کنید که n توانی از ۲ یک عدد صحیح مثبت است.

i	j	تعداد تکرار
1	8	1
2	4	1
3	2	1
4	1	1
5		⋮
		1+1+...+ 1
		N بار

```
For (i=1, i<=n;i++){
J=n;
While(j>=1){
<body of the while loop >|| needsθ(1)
J=⌊j/2⌋
   }
}
```

۳)پیچیدگی زمانی (T9n برای حلقه های تودرتویی زیر چیست؟ با فرض $2\ (n=2^k)$ =k. عدد صحیح مثبت

```
I=n;
While (i>=1){
J=I;
While(j<=n){
<body of the inner while loop>||needs θ(1)
```

$n=2^3=8$

$T(n)=[\log n]+1$

4 بار اجرا

i	j	تکرارحلقه
8		1
4	16	1
2		1
1		1
		⋮

J=2*j;

}

$I=\left\lfloor\frac{i}{2}\right\rfloor$;

}

٤)الگوریتم ۷-۱ (جکله n ام فیبوناچی تکراری) به طور واضح نسبت به n خطی است. نشان دهید این الگوریتم برحسب اندازه ورودی اش زمانی نمایی است.

$T(n)>2^{n/2}$ $\qquad$ $n=2\xrightarrow{\text{از روی درخت بازگشتی}}$ $\qquad$ $T(2)=3>2=2^{2/2}$

$n=3\rightarrow T(3)=5>2.83\approx 2^{3/2}$

مقدار *T(n)* برابر با حاصل جمع *T(n-1)* و *T(n-2)* به علاوه یک گروه در ریشه می باشد.

$T(N)>2^{n/2}$ $\qquad$ $T(n)=T(n-1)+T(n-2)+1$

$>2^{(n-1)/2}+2^{(n-2)/2}+1$

$>2^{(n-1)/2}+2^{(n-2)/2}+1=2\times 2^{(n/2)-1}=2^{n/2}$

۵)پیچیدگی زمانی الگوریتم ۶-۱ (جمله *n*ام فیبوناچی، بازگشتی) را بر حسب اندازه ورودی تعیین کنید؟

در الگوریتم فیبوناچی ، *n* ورودی و تعداد بیت های لازم برای کد کردن *n* را می توان به عنوان اندازه ورودی در نظر گرفت. اندازه $\quad \log 64=6\Leftarrow 64$

$N=13=(1101)_2 \quad \rightarrow \lfloor\log n\rfloor+1 \quad T(n)=\lfloor\log n\rfloor+1$

٦)آیا می توانید درستی الگوریتم های خود برای مسائل ۷-۱ را به اثبات برسانید؟

$S=[3^{s1} \quad 2 \quad 6 \quad 1 \quad 8 \quad 910^{s7}]$ $\quad$ $Max=s[1]=3$ $\quad$ $i+1=2$ $\quad$ $s[2]=2$

$3>2$ *تمرین۱:*

$i+1=3 \quad s[3]=6 \quad 3<6 \quad \rightarrow Max=s[3]=6\ldots Max=12$

تمرین ۲:

$S=[3^{s1}\quad 2\quad 6\quad 1\quad 8\quad 910^{s7}]$ min=s[1]=3 i+1=2 s[2]=2 $3>2\rightarrow$ min=2

I+1=3 s[3]=6 $2<6$ i+1=4 s[4]=1 $2>1 \ldots\Rightarrow$ min=1

تمرین ۵:

$(780^{m}, 155^{n})$ Bmm(780,155) m%n=5

15

Bmm(155,5)

15

Bmm(5,0) if(n==0)→m Bmm=5

۷)ورودیهای مرتب سازی در جی 8n2 بار و ورودیهای مرتب سازی ادغامی 64_nlogn بار اجرا می شوند. برای چه مقادیر n مرتب سازی در جی ، مرتب سازی ادغامی را شکست می دهد. $(8n^2<64\ \log n)$

$(8n^2<64\ \log n)\Rightarrow\ \forall\ n\varepsilon N\ \forall\ \exists\ c,N_0 \mid f(x)\geq c\ g(n)$ c=64

برای nهای بزرگتر یا مساوی ٦٤ ، الگوریتم درجی ادغامی را شکست می دهد.

$8n^2>64\ n\log n$ n=4 $8\times4^2<64\times4\times\log4$ $8\times16<64\times8$ $\times$

$N=32\Rightarrow$ $8\times(32)^2<64\times(5\times32)$ $8192<10240$ $\times$

$N=64\Rightarrow$ $8\times(64)^2<64\times(64\times\log_2{64})$ $8(64)^2>6(64)^2$ $\surd$

۸) مرتب سازی درجی را طوری تغییر دهید تا مرتب سازی را به صورت صعودی به جای ترتیب غیر نزولی انجام دهد. این الگوریتم در هر مرحله کوچکترین عضو را پیدا کرده و در ابتدا قرار می دهد.

```
   Exchange sort (A[n])
{
For (i=1; i<=n-1;i++)
For(j=i+1;j<=n;j++)
If(A[i]>A[j])
Swap(A[i],A[j])
}
```

درگام N-1 آرایه مرتب می شود.

مرتبه اجرایی الگوریتم همواره $O(n^2)$ می باشد.

این دستور A[i] و A[j] را exchange می کند .

۹)برای مسئله جستجو کدی بنویسید که آن را به صورت خطی حل نماید.

```
Void  s search (int  n,
          Cost keytype  s[1..n],
               Keytype  m,
          Index & i);
                  {
              i=1;
              While(i<=n  &&  s[i]!=m)
               i++;
               if(i>n)
```

عددm در کدام خانه آرایه s با n مقدار قرار دارد.

-اگر m در آرایه نباشد، تابع مقدار O را برمی گرداند.

-T(n)=n و پیچیدگی زمانی خطی می باشد.

I=0;

۱۰) مسئله جمع دو عدد صحیح دودویی n بیتی را در نظر بگیرید که در ۲ آرایه n عنصریA,B ذخیره شده اند. جمع دو عدد باید به فرم دودویی در آرایه n+1 عضوی ذخیره شود. الگوریتمی برای حل مسئله ارائه دهید.

```
Number sum (int n,
                        Cost number A[1..n]
                         Cost number B[1..n]
                         Number c[1..n+1]);
{
Index  I,j
Number sum;
Sum=0;
For(i=1; i<=n;i++)
For(j=1;j<=n;j++)
Sum=sum+A[i]+B[j];
Sum=c[ ];
}    return c[ ];
```

۱۱) چند درخت جستجوی دودویی می توان با شش کلید متمایز بنا کرد؟   n=6

با کلیدn مجزا می توان $\frac{1}{n+1}\binom{2n}{n}$ درخت BST مجزا ساخت. $\frac{1}{n+1}\binom{2n}{n} \Rightarrow \frac{1}{6+1}\binom{12}{6}=132$

۱۲)جواب بازگشت معادلات بازگشتی زیر را بدست آورید.

$$\begin{cases} an = a_{n-1} + 2a_{n-2} \\ a_0 = 2 \\ a_1 = 7 \end{cases} \quad (۱$$

$a_n = c_1 r_1^n + c_2 r_2^n$ $\quad a_n=c_1 2^n+c_2(-1)^n$ $\quad a_0=c_1 2^0+c_2(-1)^0=c_1+c_2$

$a_1=c_1 2^1+c_2(-1)^1=2c_1-c_2$

$r^2=r+2 \Rightarrow r_1=2, r_2=-1$

$\begin{cases} c1+c2=2 \\ 2c1-c2=7 \end{cases} \quad \begin{matrix} c1=3 \\ c2=-1 \end{matrix}$ $\quad a^n=3\times(2)^n+(-1)\times(-1)^n$

$2= \begin{cases} an=3a_{n-1}-2a_{n-2} \\ a_0=2 \\ a_1=3 \end{cases}$ $\quad \begin{cases} c_1+c_2=2 \\ 2c_1+c_2=3 \end{cases} \quad \begin{matrix} c_1=1 \\ c_2=1 \end{matrix}$ $\quad a^n=1\times 2^n+1\times 1^n \Rightarrow$

$a^n=2^n+1^n$ جواب بازگشت

$r^k=c_n.r^{k-1}+c_{n-1}.r^{k-1} \Rightarrow \quad r^2=3r-2 \quad r_1=2 \quad , \quad r_2=1 \quad a_n=c_1 2^n+c_2 1^n$

$\Rightarrow \begin{cases} a_0=c_n 2^0+c_2 1^0=c_1+c_2 \\ a_1=c_1 2^1+c_2 1^1=2c_1+c_2 \end{cases}$