حل برخی از تمرینات کتاب

ساختمان داده ها و الگوریتم ها

گردآورنده:

www.fanavari-it.ir

فصل اوّل

حل بعضی از تمرینات

۱- الف) چون عمل S++ در یک ثابت زمانی C صورت می‌گیرد بنابراین داریم:

$$T(n)=\sum_{i=۱}^{n}\sum_{j=۱}^{n}C$$

$$=C\sum_{i=۱}^{n}i=C(۱+۲+.....+n)$$

$$=C\left(\frac{n(n+۱)}{۲}\right)$$

بنابراین می‌توان نوشت:

$$T(n)=C_1n^2+C_2n$$

قسمت‌های دیگر سوال ۱ نیز به روش الف قابل حل می‌باشند.

۲- الف) می‌توان سوال را با استفاده از تشکیل جدول (مطابق الگوی درس) حل کرد ولی راحت‌تر است که به صورت زیر عمل کنیم.

حلقه while زمانی به پایان می‌رسد که متغیر I به ۱ برسد و متغیر I زمانی به 1 می‌رسد که :

$$\frac{n}{2^i}=1$$

باشد در اینصورت می‌توان نتیجه گرفت که اگر:

$$i=\log_2^n$$

باشد به عبارت دیگر حلقه while به تعداد $\log_2^n$ تکرار می‌شود بنابراین:

$$T(n)=C\log_2^n$$

قسمت‌های دیگر سوال نیز به روش بالا قابل حل می‌باشد.

۳-

(۱

$\exists c_۱ , c_۲ , n_o$ به طوریکه $\forall n \geq n_0$ $\qquad C_2n^2<7n^2-6n+2<C_1n^2$

$$\xrightarrow{\text{تقسیم بر } n^2} C_2<7-\frac{6}{n}+\frac{2}{n^2}<C_1$$

با حل رابطه بالا n_0, C_2, C_1 را می‌توان پیدا کرد بنابراین:

$T(n) \in (n^2)$ (بازای $C_2 = 6 , C_1 = 9 , n_0 = 3$)

۳)

$T(n) = n! + 7n^5$

$\exists C, n_0 \geq 0 \quad \forall n > n_0 \Rightarrow T(n) \leq Cn^n$

بنابراین خواهیم داشت:

$T(n) = n! + 7n^5 \leq n^n + 7n^5$

$\leq n^n + 7n^n = 8n^n$

لذا بازای C=8 رابطه :

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

برقرار خواهد بود.

بقیه موارد نیز با استفاده قضایای بیان شده قابل اثبات می‌باشند.

۵- طبق صورت مساله داریم:

1) $(T_1(n) \in \Omega(f(n)))$

$\Rightarrow \exists C_1 , n_{o1}$ به طوری که $\forall n \geq n_{o1} \quad C_1 f(n) \leq T_1(n)$

و همچنین داریم:

2) $T_2(n) \in \Omega(g(n)) \Rightarrow \exists C_2 , n_{o2}$ به طوری که $\forall n \geq n_{o2} \quad C_2 g(n) \leq T_2(n)$

$\xrightarrow{1+2} C_1 f(n) + C_2 g(n) \leq T_1(n) + T_2(n)$ (۳)

با توجه به رابطه ۳ اگر $C = \min\{C_1, C_2\}$ و $n_0 = \max\{n_{01}, n_{02}\}$ آنگاه خواهیم داشت:

$C(f(n) + g(n)) \leq T_1(n) + T_2(n)$

$\Rightarrow C \max\{f(n), g(n)\} \leq T_1(n) + T_2(n)$

لذا عبارت زیر حاصل می‌شود:

$T(n) \in \Omega(\max\{f(n), g(n)\})$

با توجه به اثبات بالا براحتی می‌توان بقیه موارد را اثبات یا رد کرد.

۶- با توجه به تعریف θ داریم:

$$\exists C_1, C_2, n_0 \geq 0 \qquad \forall n \geq n_0 \Rightarrow C_1 n^3 \leq n^5 + 14n^3 \leq C_2 n^3$$

طرف راست رابطه بالا برقرار نیست یعنی

$$n^5 + 14n^3 \leq C_2 n^3$$

$$\Rightarrow n^2 + 14 \leq C_2 \Rightarrow n^2 \leq C_2 - 14 \quad (1)$$

رابطه ۱ به وضوح رابطه نادرستی است. بنابراین $T(n) \notin \theta(n^3)$.

بقیه موارد نیز براحتی قابل اثبات است.

۹-

```
int    Bin Srch (a , low , high , x)
{
       if (low > high)
              return -1 ;
       else {
              mid= (low+ high)/2 ;
              if (x= = a [mid])
                     return mid ;
              else if (x<a [mid])
                     return Bin Srch (a, low , mid-1 , x);
              else
                     return BinSrch (a , mid+1 , high , x) ;
       }
}
```

A لیست،low کران پایین، high کران بالا می‌باشد.

تابع بالا در بدترین حالت $T(n) \in O(\log_2^n)$ می‌باشد.

۱۰- تابع زمانی تابع s برابر است با:

$$T(n) = \begin{cases} C & \text{if } n = 1 \\ T(n-1) + d & \text{if } n > 1 \end{cases}$$

حال رابطه بالا را با روش تکرار حل می‌کنیم:

$$T(n) = T(n-۱) + d$$
$$= T(n-۲) + ۲d$$
$$= ...$$
$$= T(n-i) + i*d$$

بازای $n-i=۱$ ، $T(n-i)$ به مقدار ثابت می‌رسد بنابراین:

$$n-i=1 \Rightarrow i=n-1$$
$$\Rightarrow T(n) = T(1) + (n-1)*d$$

لذا خواهیم داشت $T(n) \in O(n)$.

۱۵- الف) با یک تغییر متغیر، براحتی می‌توان مسئله را حل کرد. برای اینکار $\sqrt{n} = m$ قرار می‌دهیم بنابراین:

$$T(m^2) = T(m) + C$$

حالا فرض می‌کنیم $T(m^2) = x^2$, $T(m) = x$ باشد آنگاه:

$$x^2 = x - c$$

حال با استفاده از راه‌حل‌های ارائه شده در طراحی الگوریتم براحتی رابطه بالا قابل حل است.

د) طبق نکته گفته شده در نکات مهم فصل داریم:

$f(n) = n^2, b = 2 , a = 15$ حال $\log_b^a$ را محاسبه می‌کنیم داریم:

$$\log_2^{15}$$

بنابراین درجه f(n) از درجه $\log_2^{15}$ کمتر است. لذا طبق این نکته خواهیم داشت:

$T(n) \in O(\log_2^{15})$.

فصل دوّم

حل بعضی از تمرینات فصل

۶- برای نمایش چند جمله‌ایها می‌توان ساختار داده زیر را ارائه داد:

Cof1	Cof2	....
Pow1	Pow2	....

یعنی یک آرایه دو بعدی که در یک سطر کلیه توانها و در سطر دیگر آن کلیه ضرایب را قرار می‌دهیم. حال براحتی عملگرهای Add جمع دو چند جمله‌ای و mult ضرب دو چند جمله‌ای را می‌توان انجام داد.

۷- ما آرایه سه بعدی را ارائه می‌دهیم که با بسط آن حالت کلی نیز حاصل می‌شود.

محل عنصر a[i] [j] [k] در روش ذخیره‌سازی ستونی در ماتریس $a[l_1....u_1][l_2....u_2][l_3....u_3]$ از رابطه زیر حاصل می‌شود:

محل عنصر a[i][j][k] $= (k - L_3)(u_2 - l_2 + 1)(u_1 - l_1 + 1) + (j - L_2)(u_1 - L_1 + 1) + (i - L_1) + \alpha$

۸- فرض کنید آرایه زیر را تعریف کنیم

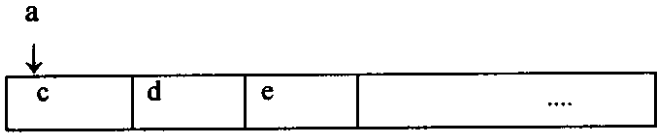
```
int  n
elementtype   list [n] ;
```

با یک تابع لیست را به صورت پویا از ورودی می‌خوانیم و توسط تابعی به نام sort با یکی از روش‌های معمول مرتب می‌کنیم.

۹-

```
void  Insert (elementtype list [ ] , int * n , elementtype item)
{
آرایه به صورت صعودی مرتب است
(*n) + + ;
for (i=0 ;i<n  ;  i++)
     if (item < list [i]){
          for (int  j=n  ;  j>=i  ; j- -)
                a [j+1] =a [j] ;
           a[i]= item ;
            break ;
          }
}
```

۱۰- بوضوح بدترین حالت در الگوریتم Insert بالا زمانی رخ می‌دهد که عنصر جدید در اولین جای آرایه بخواهد قرار گیرد:



در اینصورت n عمل جابجایی باید رخ دهد. در بهترین حالت هم ممکن است عنصر جدید در آخرین خانه آرایه قرار گیرد که فقط یک عمل جایگزینی رخ می‌دهد. البته اگر زمان جستجو را به این عملیات اضافه کنیم قضیه متفاوت است. در هر صورت تابع بالا در بدترین حالت و بهترین حالت از مرتبه O(n) خواهد بود.

۱۱- این تابع برعکس تابع Insert سوال ۹ عمل می‌کند. بدین صورت که :

```
delete (elementtype list [ ] , int *n , elementtype item)
{
    for (i=o ; i< *n ; i + +)
        if (item = = list [i]){
                for (j=i ; j <*n-1 ; j + +)
                        list[i] = list [i+1] ;
                (*n)- - ;
                break ;
            }
}
```

۱۲- تقریبا مانند مساله ۱۰ در بهترین حالت و بدترین حالت O(n) پیچیدگی زمانی تابع delete خواهد بود.

۱۳- مطابق تمرین ٦ عمل کرده و دو آرایه دو بعدی تشکیل می‌دهیم یکی برای p(x) از درجه n و دیگری Q(x)از درجه m . تابع Add را در حالت کلی به صورت زیر می‌نویسیم:

```
Void  Add (elementtype  p[ ] [2] , Q[ ] [2] , R [ ] [2])
```

که در آن R حاصل مجموع دو چند جمله‌ای می‌باشد. تابع را به صورت زیر ارائه می‌دهیم:

```
i=0  ; j=0  ; k=0  ;
while (i<n &&   j<m){
            if (p[i] [1] > Q[j] [1]){
                R[k] [l] = p[i] [1] ,
                R [k] [0]=p[k] [0] ; i++ ; k + + ;
             }
             else if (p [i] [1] = = Q[j] [1]) }
                 R [k] [1]=p[i] [1] ; //Q[j] [1]
                 R [k] [0]= p[i] [0] +Q [j] [0] ;
                 i + + ; j+ + ; k + + ;
            }
```

```
        else {
                R [k] [1] = Q [j] [1] ;
                R [k] [0] =Q [j] [0] ;
                K++ ; j ++ ;
            }
    }// while
```

بعد از حلقه هم ادامه چند جمله‌ای که باقی مانده را در چند جمله R اضافه می‌کنیم بوضوح پیچیدگی زمانی تابع بالا از مرتبه $O(m+n)$ می‌باشد.

۱٤- با استفاده از مساله ۱۳ قابل حل می‌باشد.

۱۵- ایده مساله ۱۳ به صورت برعکس در صورتی که دو لیست به صورت صعودی مرتب باشند برای حل این مساله کافی است.

فصل سوّم

حل بعضی از تمرینات فصل

۲- همانطور که در متن درس گفتیم اول پرانتزگذاری سپس انتقال پرانتز به سمت راست در (postfix) یا انتقال پرانتز به سمت چپ (prefix) و در نهایت پرانتزها را حذف می‌کنیم.

برای نمونه b,a را حل می‌کنیم (postfix)

a) $((A+B)-C) \Rightarrow AB+C-$

b) $(((A+B)*(C-D))-(E*F))$

$= AB+CD-*EF*-$

۳- در حل این تمرینات برعکس مراحل تمرین ۲ را می‌توان انجام داد. برای نمونه b,a را حل می‌کنیم:

a) $ab+cd-* = ((ab)+(cd)-)*$

$= ((a+b)*(c-d))$

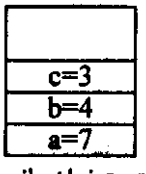
b) $abc+-d* = ((a(bc))+-d)*$

$= ((a-(b+c))*d)$

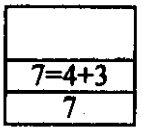
٤- یک روش، تبدیل به infix و محاسبه دستی مقدار عبارت می‌باشد. در حالیکه، روش اصلی استفاده از پشته می‌باشد. برای نمونه a را با مقادیر , c=3 , d=7 a=7 , b=4 حل می‌کنیم:

a) abc+/d+

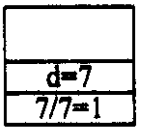
مرحله اول، سه عملگر a,b,c در پشته قرار می‌گیرند:



مرحله دوم، عملگر + از عبارت a فراخوانی می‌شود و مجموع b, c محاسبه می‌شود:



مرحله سوم، عملگر / از عبارت a فراخوانی می‌شود و مقدار تقسیم a و ۷ محاسبه می‌شود:



مرحله چهارم، عملگر * از عبارت a فراخوانی می‌شود و مقدار حاصلضرب d و ۱ محاسبه می‌شود. بنابراین خروجی عدد 7 می‌باشد.

۵- براحتی با استفاده از پشته معمولی می‌توان تابع فوق را نوشت در متن درس نیز به تابع مربوطه اشاره شده است.

٦- با استفاده از یک پشته اضافی این کار به صورت زیر انجام می‌شود:

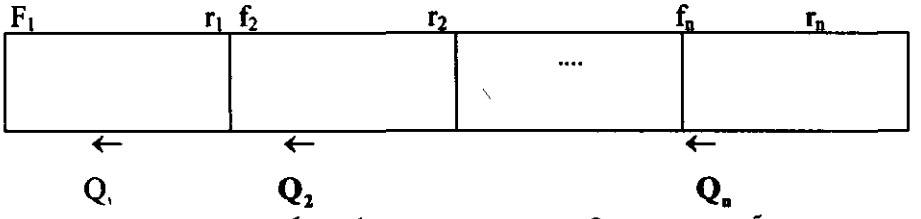
c	a	c
b	b	b
a	c	a
S1	s2	s3

بقیه سوالات یا در متن درس توضیح داده شده یا به عنوان تمرین برای خواننده واگذار می‌شود.

فصل چهارم

حل بعضی از تمرینات فصل

۵- مثل پشته چندگانه عمل می‌کنیم می‌توان به صورت شکل زیر عمل کرد:



برای بدست آوردن r_iها و f_i ها از پشته‌های چند گانه الگوبرداری کنید.

۶- می‌توانیم از صف اولویت برای حل این مساله استفاده کنیم.

۷- سه صف بنام‌های:

Upper
Low
High

تعریف می‌کنیم. سپس از صف اولیه کاراکتر به کاراکتر خوانده با توجه به نوع هر کاراکتر در صف مربوطه می‌نویسیم.

در کل تمرینات این فصل با توجه به توضیحات داده شده به عنوان تمرین به خواننده واگذاری می‌شود.

فصل پنجم

حل بعضی از تمرینات فصل

۲- بند الف و د را حل می‌کنیم.

الف- با فرض اینکه اشاره به لیست به خانه آخر اشاره می‌کند مساله را حل می‌کنیم

```
Node * temp;
temp = getnode();
temp → Info = x;
temp → next = NULL
```

ایجاد گره جدید

```
p → next = temp
```

الحاق گره جدید به انتهای لیست

د- لیست P را با فیلد Info از نوع صحیح در نظر می‌گیریم:

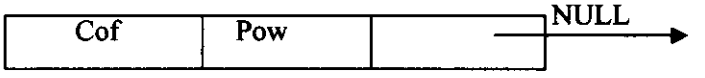
```
While (p) {
      sum + = p → Info ;
      p =p → next ;
}
 cout << sum ;
```

۳- در لیست کلاً، دسترسی به گره‌ها ترتیبی است. بنابراین برای جستجوی لیست ترتیب عناصر نقشی ندارند. بنابراین الگوریتم جستجو همان الگوریتم جستجوی خطی است و در حالت متوسط $\frac{N+1}{2}$ مقایسه نیاز دارد. ولی در آرایه وضع چنین نیست.

٤- به عنوان نمونه بند الف را حل می‌کنیم. با فرض اینکه فیلد Info از نوع صحیح است، قطعه کد زیر را می‌نویسیم:

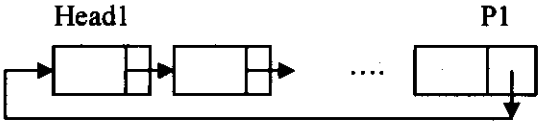
```
Max=p → Info ; // first Node
p=p → next ;
While (p) {
                if (p → Info>max)
                      Max =p → Info ;
                p=p → next ;
}
cout << max ;
```

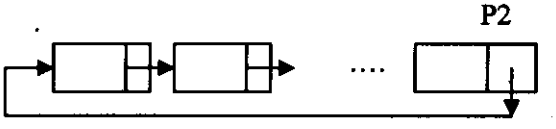
۷- ساختار چند جمله‌ای با لیست پیوندی برای هر گره را به صورت زیر تعریف می‌کنیم:



بنابراین براحتی می‌توان چند جمله‌ای را در لیست پیاده‌سازی کرد. سپس تابع جمع دو چند جمله‌ای را با توجه به تمرین حل شده در فصل ۲ می‌توان نوشت.

۹- با فرض اینکه ساختار لیست حلقوی به صورت زیر می‌باشد:





در اینصورت توسط تابع زیر می‌توان دو لیست را به هم الحاق کرد:

```
Head1= p1 → next ;
p1 → next=p2 → next;
p2 → next=Head1;
```

بنابراین دو لیست به هم الحاق می‌شوند.

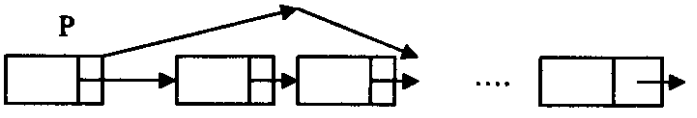
۱۴- فرض کنید L1 , L2 دو لیست مرتب باشند. تابع در حالت کلی به صورت زیر می‌نویسیم (عناصر غیرتکراری هستند):

```
Node  *L3=*temp = Null ;
Head = L3 ;
While ( L1 && L2 ) {
    temp= getnode ( );
    if (L1 → Info<L2 → Info){
              temp → Info=L1 → Info ;
              L1=L1 → next ;
   else {
            temp  →  Info = L2 → Info ;
             L2=L2 → next ;
    }
     if (L3==NULL)
              L3=temp ;
```

```
    else
            L3 → next=temp ;
            L3=L3 → next ;
   }
}
If (L1)
      L3 → next= L1 ;
 If (L2)
      L3 → next = L2 ;
```

۱۹- فرض کنید بعد از گره P دو گره بخواهیم، حذف کنیم:

```
p → next = (p → next) → next ;
```

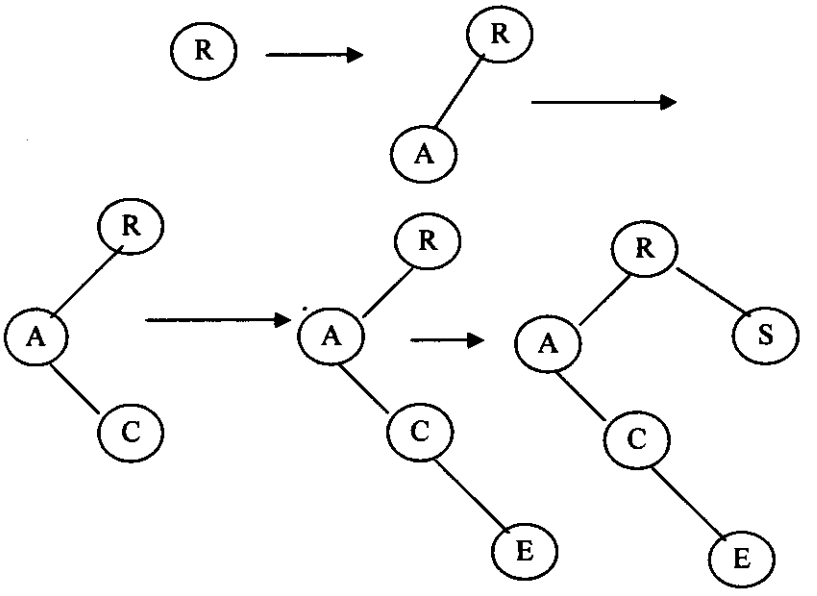


فصل ششم

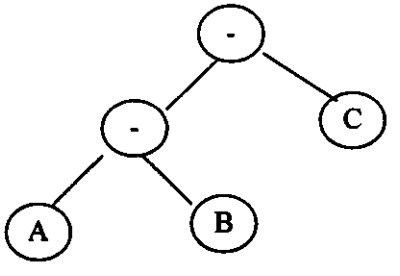
حل بعضی از تمرینات فصل

۱- به عنوان نمونه a را حل می‌کنیم

a) R , A , C , E , S



۳- برای نمونه h را حل می‌کنیم:



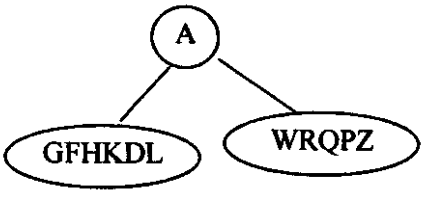
پیمایش postorder به صورت زیر است:

A B – C –

پیمایش preorder آن نیز، به صورت زیر خواهد بود:

- - A B C

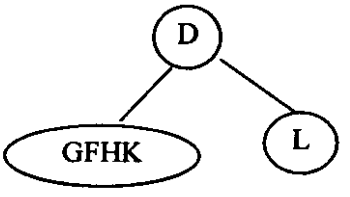
۵- از اولین کاراکتر پیمایش preorder ، ریشه درخت مشخص می‌شود. بنابراین با استفاده از این کاراکتر در پیمایش inorder ، زیر درخت چپ و راست مشخص می‌شود:



حال برای هر زیر درخت دوباره همین کار را انجام می‌دهیم. برای زیر درخت چپ داریم:

Inorder : G F H K D L
Preorder : D F G H K L

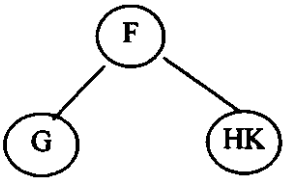
بوضوح D ریشه این زیر درخت است بنابراین:



دوباره برای این زیر درخت داریم:

Inorder : G F H K
Preorder : F G H K

بنابراین F ریشه زیر درخت جدید می‌باشد:

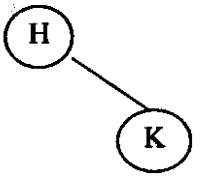


حال برای زیر درخت راست درخت بالا داریم:

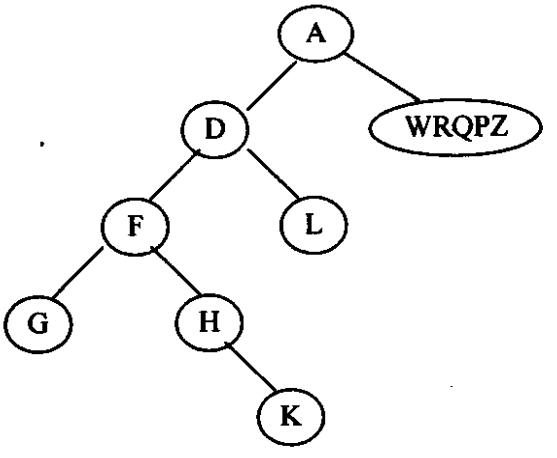
Inorder : H K

Preorder : H K

پس خواهیم داشت:



حال با جایگزینی درخت‌های بالا درخت زیر حاصل می‌شود:



اگر مراحل بالا را برای زیر درخت راست هم انجام دهیم مساله حل خواهد شد.

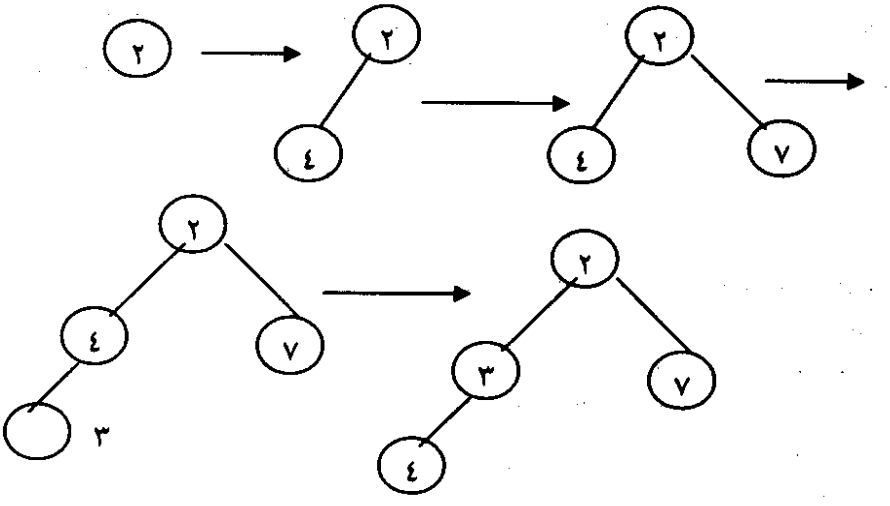
۷- در سوالات تستی ۲، ۹ و ۱۷ به سوالات پاسخ داده شده است.

۱۵- با استفاده از تمرین ۷ می‌توان حل بکنید.

۲۵- برای نمونه اولین قسمت را حل می‌کنیم:

2 , 4 , 7 , 3 , 1 , 8

درخت را همانطور که در متن درس گفتیم مرحله به مرحله می‌سازیم **(min heap)**


٢
٣
٤
٢
٤
٧
٢
٤
٧
٣
٢
٣
٧
٤

فصل هشتم

حل بعضی از تمرینات فصل

۵- قسمت C را حل می‌کنیم:

C) 70 , 57 , 99 , 34 , 56 , 89

نخست عنصر دوم با اول مقایسه کرد لیست دو عنصری را مرتب می‌کنیم سپس عنصر سوم با دو عنصر قبلی تشکیل لیست مرتب سه تایی می‌دهند و الی آخر.

70 , 57 → 57, 70 مرحله اول

99 70 57 → 99 70 57 مرحله دوم

99 70 34 57 → 99 70 57 34 مرحله سوم

99 70 57 56 34 → 99 70 57 56 34 مرحله چهارم

99 89 70 57 56 34 → 99 89 70 57 56 34 مرحله پنجم

٦- قسمت c را از از این سوال حل می‌کنیم:

C) 70 57 99 34 56 89

در هر فاز بزرگترین عنصر در انتهای لیست مربوطه قرار می‌گیرد و در هر مرحله لیست کوچکتر حاصل می‌شود و در نهایت لیست دوعنصره مرتب می‌شود:

مرحله اول : 50 57 99 34 56 89 → 50 57 34 56 89 99

مرحله دوم : 50 57 34 50 89 → 50 34 56 57 89

مرحله سوم : 50 34 56 57 → 34 50 56 57

مرحله چهارم : 34 50 56 → 34 50 56

لیست حاصل مرتب می‌باشد.

۸- قسمت C را با الگوریتم مرتب‌سازی سریع، مرتب می‌کنیم

C) 70 57 99 34 56 89

عنصر ۷۰ را به عنوان عنصر محور در نظر می‌گیریم، سپس اولین کوچکترین عنصر از سمت راست و اولین بزرگترین عنصر از سمت چپ (در مقایسه با عنصر محور) را پیدا می‌کنیم:

70 57 99 34 56 89

سپس جای این دو عنصر یعنی 56 , 99 را با هم جابجا می‌کنیم:

70 57 56 34 99 89

این روند را دوباره انجام می‌دهیم. دو عنصر 34 , 99 بدست می‌آید.

در اینجا اشاره‌گرهای چپ و راست به هم برخورد می‌کنند. بنابراین عنصر محور را با 34 جابجا می‌کنیم:

34 57 56 70 99 89

همانطور که ملاحظه می‌کنید تمام عناصر سمت چپ از عنصر محور کوچکتر و تمام عناصر راست بزرگ‌تر هستند. بنابراین عنصر 70 در جای مناسب قرار گرفته است. حال روند بالا را برای لیست چپ و راست ادامه می‌دهیم:

لیست چپ : 34 57 56

لیست راست : 99 89

با ادامه روند، کل لیست مرتب می‌شود.

۱۲- قسمت C را حل می‌کنیم:

C) 70 57 99 34 56 89

q_0	70	
q_1		
q_2		
q_3		
q_4	34	
q_5		
q_6	56	
q_7	57	
q_8		
q_9	99	89

→

q_0		
q_1		
q_2		
q_3	34	
q_4		
q_5	56	57
q_6		
q_7	70	
q_8	89	
q_9	99	

همانطور که ملاحظه می‌کنید لیست حاصل که عبارت است از:

34 56 57 70 89 99

مرتب می‌شود (در دو مرحله چون اعداد حداکثر ۲ رقمی بودند)

با استفاده از تمرینات حل شده و متن درس و نکات ارائه شده بقیه تمرینات به خواننده واگذار می‌کنیم.