

حل برخی از تمرینات کتاب

ساختهای داده ها و الگوریتم ها

کرد اوزنده:

www.fanavari-it.ir

فصل اول



حل بعضی از تمرینات

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

$$\begin{aligned} T(n) &= \sum_{i=1}^n \sum_{j=1}^n C \\ &= C \sum_{i=1}^n i = C(1+2+\dots+n) \\ &= C \left(\frac{n(n+1)}{2} \right) \end{aligned}$$

بنابراین می‌توان نوشت:

$$T(n) = C_1 n^2 + C_2 n$$

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 ۱ 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)

$$\exists c_1, c_2, n_0 \quad \forall n \geq n_0 \quad C_2 n^2 < 7n^2 - 6n + 2 < C_1 n^2$$

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

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

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

$T(n) = n! + \sqrt{n}^n$

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

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 می باشند.

۵- طبق صورت مساله داریم:

۱) $(T_1(n) \in \Omega(f(n)))$
 $\Rightarrow \exists C_1, n_{01}$ به طوری که $\forall n \geq n_{01} \quad C_1 f(n) \leq T_1(n)$

و همچنین داریم:

۲) $T_2(n) \in \Omega(g(n)) \Rightarrow \exists C_2, n_{02}$ به طوری که $\forall n \geq n_{02} \quad C_2 g(n) \leq T_2(n)$

 $\xrightarrow{+} C_1 f(n) + C_2 g(n) \leq T_1(n) + T_2(n) \quad (3)$

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

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

با توجه به اثبات بالا بر احتی می توان بقیه موارد را اثبات یا رد کرد.

۶- با توجه به تعریف θ داریم:

$$\exists C_1, C_2, n_0 \geq 0 \quad \forall n \geq n_0 \Rightarrow C_1 n^r \leq n^0 + 14n^r \leq C_2 n^r$$

طرف راست رابطه بالا برقرار نیست یعنی

$$n^0 + 14n^r \leq C_2 n^r$$

$$\Rightarrow n^0 + 14 \leq C_2 \Rightarrow n^0 \leq C_2 - 14 \quad (1)$$

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);
    }
}
```

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:

$$\begin{aligned}
 T(n) &= T(n-1) + d \\
 &= T(n-2) + 2d \\
 &= \dots \\
 &= T(n-i) + i * d
 \end{aligned}$$

بازای $i = 1$ به مقدار ثابت می‌رسد بنابراین:

$$\begin{aligned}
 n - i = 1 \Rightarrow i = n - 1 \\
 \Rightarrow T(n) = T(1) + (n-1) * d
 \end{aligned}$$

لذا خواهیم داشت $T(n) \in O(n)$

۱۵- الف) با یک تغییر متغیر، براحتی می‌توان مسئله را حل کرد. برای اینکار قرار می‌دهیم بنابراین:

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

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

$$x^2 = x - C$$

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

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

حال $\log_2^{15} n$ را محاسبه می‌کنیم داریم:

$$\log_2^{15}$$

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

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

فصل دوم



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 بر احتی 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 $a[i][j]$ در روش ذخیره‌سازی ستونی در ماتریس $a[l_1 \dots u_1][l_2 \dots u_2][l_3 \dots u_3]$ از رابطه زیر حاصل می‌شود:

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

-۸- فرض کنید آرایه زیر را تعریف کنیم

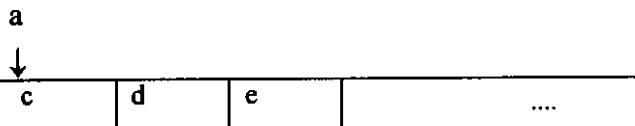
```
int n
elementtype list[n];
```

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 بر عکس تابع Insert سوال ۹ عمل می‌کند. بدین صورت که :

```

delete (elementtype list [ ] , int *n , elementtype item)
{
    for (i=0 ; i< *n ; i++)
        if (item == list [i]){
            for (j=i ; j < *n-1 ; j++)
                list[i] = list [i+1];
            (*n)--;
            break;
        }
}
  
```

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

۱۳- مطابق تمرین ۶ 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] [1] = 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.
- ۱۴- با استفاده از مساله ۱۳ قابل حل می‌باشد.
 - ۱۵- ایده مساله ۱۳ به صورت بر عکس در صورتی که دو لیست به صورت صعودی مرتب باشند برای حل این مساله کافی است.

فصل سوم



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:

$$a) ab+cd-* = ((ab)+(cd))-* \\ = ((a+b)*(c-d))$$

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

۴- 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 $d=7, c=3$ حل می‌کنیم: $b=4, a=7$

$$a) abc+/d+$$

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

c=3
b=4
a=7

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

7=4+3
7

مرحله سوم، عملگر / از عبارت a فراخوانی می شود و مقدار تقسیم a و 7 محاسبه می شود:

$d=7$
$7/7=1$

مرحله چهارم، عملگر * از عبارت a فراخوانی می شود و مقدار حاصلضرب d و 1 محاسبه می شود. بنابراین خروجی عدد 7 می باشد.

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

۶- با استفاده از یک پشته اصلی این کار به صورت زیر انجام می شود:

c	a	c
b	b	b
a	c	a

s_1 s_2 s_3

بقیه سوالات یا در متن درس توضیح داده شده یا به عنوان تمرین برای خواننده واکذار می شود.

فصل چهارم



WWW.FANAVARI-IT.IR
INFORMATION TECHNOLOGY ERA

حل بعضی از تمرینات فصل

۵- مثل پشته چندگانه عمل می کنیم می توان به صورت شکل زیر عمل کرد:

F_1	r_1	f_2	r_2	...	f_n	r_n

$\leftarrow \quad \leftarrow \quad \leftarrow$

Q_1

Q_2

Q_n

برای بدست آوردن r_i ها و f_i ها از پشته های چند گانه الگوبرداری کنید.

۶- می توانیم از صفت اولویت برای حل این مساله استفاده کنیم.

۷- سه صفت بنام های:

Upper
Low
High

تعریف می کنیم. سپس از صفت اولیه کاراکتر به کاراکتر خواننده با توجه به نوع هر کاراکتر در صفت مربوطه می نویسیم.

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

فصل پنجم



WWW.FANAVARI-IT.IR

INFORMATION TECHNOLOGY ERA

حل بعضی از تمرینات فصل

۲- بند الف و د را حل می کنیم.

الف- با فرض اینکه اشاره به لیست به خانه آخر اشاره می کند مساله را حل

می کنیم

```

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 مر ظیع گیریم:

```

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

```

۳- در لیست کلاه 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.

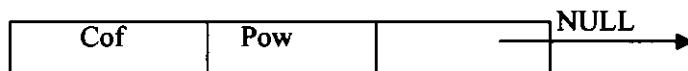
۴- به عنوان نمونه بند الف را حل می کنیم. با فرض اینکه فیلد Info از نوع صحیح است، قطعه کد زیر را می نویسیم:

```

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

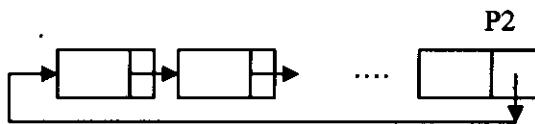
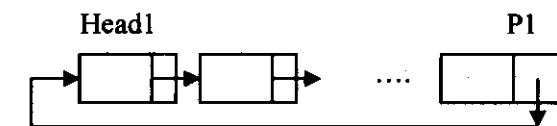
```

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



بنابراین براحتی می‌توان چند جمله‌ای را در لیست پاده‌سازی 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 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 ;
19- فرض کنید بعد از گره P دو گره بخواهیم، حذف کنیم:
p → next = (p → next) → next ;

```



فصل ششم



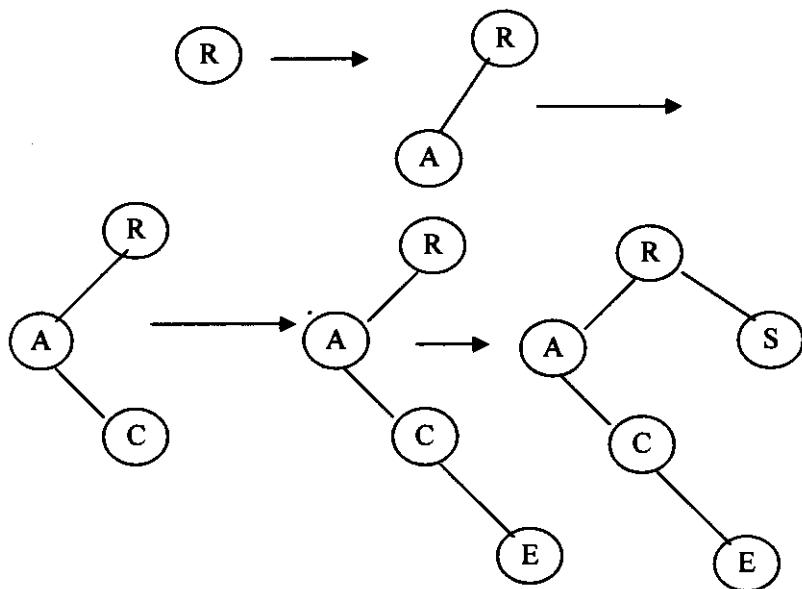
WWW.FANAVARI-IT.IR

INFORMATION TECHNOLOGY ERA

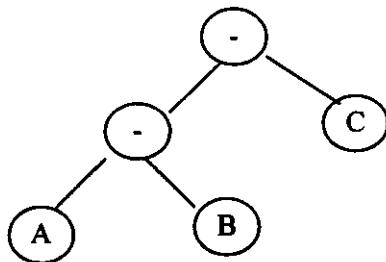
حل بعضی از تمرینات فصل

۱- به عنوان نمونه a را حل می کنیم

a) R , A , C , E , S



۳- برای نمونه h را حل می کنیم:



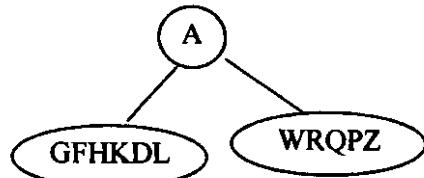
پیمایش postorder به صورت زیر است:

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:



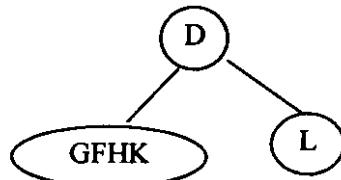
حال برای هر زیر درخت دوباره همین کار را انجام می‌دهیم. برای زیر درخت

چپ داریم:

Inorder : G F H K D L

Preorder : D F G H K L

بوضوح D ریشه این زیر درخت است بنابراین:

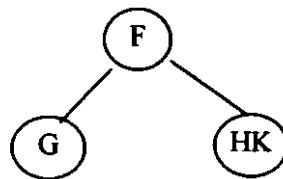


دوباره برای این زیر درخت داریم:

Inorder : G F H K

Preorder : F G H K

بنابراین F ریشه زیر درخت جدید می‌باشد:

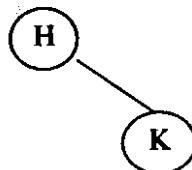


حال برای زیر درخت راست درخت بالا داریم:

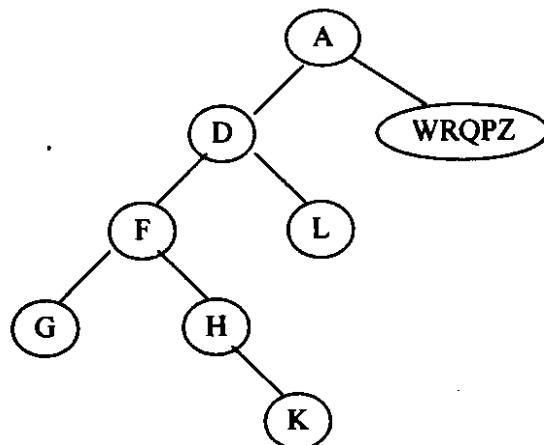
Inorder : H K

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.

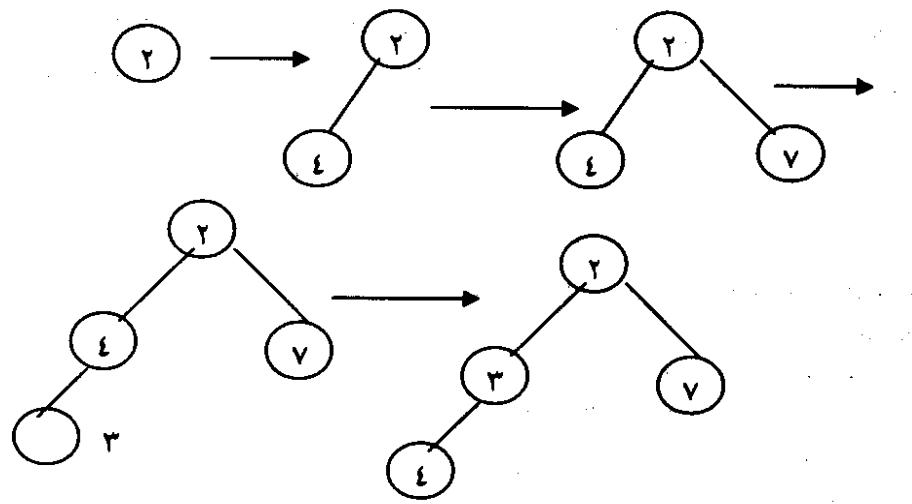
- در سوالات تستی ۲، ۹ و ۱۷ به سوالات پاسخ داده شده است.

- با استفاده از تمرین ۷ می‌توان حل بکنید.

- برای نمونه اولین قسمت را حل می‌کنیم:

2, 4, 7, 3, 1, 8

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



فصل هشتم



WWW.FANAVARI-IT.IR

INFORMATION TECHNOLOGY ERA

حل بعضی از تمرینات فصل

۵- قسمت C را حل می کنیم:

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

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

عنصر سوم با دو عنصر قبلی تشکیل لیست مرتب سه تایی می دهنده و الی آخر.

مرحله اول $70 \rightarrow 57, 70$

مرحله دوم $57 \ 70 \ 99 \rightarrow 57 \ 70 \ 99$

مرحله سوم $57 \ 70 \ 99 \ 34 \rightarrow 57 \ 70 \ 34 \ 99 \rightarrow 34 \ 57 \ 70 \ 99$

مرحله ۴ $34 \ 57 \ 70 \ 99 \ 56 \rightarrow 34 \ 57 \ 70 \ 56 \ 99 \rightarrow 34 \ 56 \ 57 \ 70 \ 99$

چهارم

مرحله پنجم $34 \ 56 \ 57 \ 70 \ 99 \ 89 \rightarrow 34 \ 56 \ 57 \ 70 \ 89 \ 99$

۶- قسمت C را از این سوال حل می کنیم:

C) 70 57 99 34 56 89

در هر فاز بزرگترین عنصر در انتهای لیست مربوطه قرار می گیرد و در هر مرحله

لیست کوچکتر حاصل می شود و در نهایت لیست دو عنصره مرتب می شود:

مرحله اول $50 \ 57 \ 99 \ 34 \ 56 \ 89 \rightarrow 50 \ 57 \ 34 \ 56 \ 89 \ 99$

مرحله دوم $50 \ 57 \ 34 \ 50 \ 89 \rightarrow 50 \ 34 \ 56 \ 57 \ 89$

مرحله سوم $50 \ 34 \ 56 \ 57 \rightarrow 34 \ 50 \ 56 \ 57$

مرحله چهارم $34 \ 50 \ 56 \rightarrow 34 \ 50 \ 56$

لیست حاصل مرتب می باشد.

۸- قسمت C را با الگوریتم مرتب سازی سریع، مرتب می کنیم

C) 70 57 99 34 56 89

عنصر ۷۰ را به عنوان عنصر محور در نظر می گیریم، سپس اولین کوچکترین

عنصر از سمت راست و اولین بزرگترین عنصر از سمت چپ (در مقایسه با عنصر

محور) را پیدا می کنیم:

70 57 99 34 56 89

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

70 57 56 34 99 89

این روند را دوباره انجام می دهیم. دو عنصر 34 , 99 بدست می آید.

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:

34 57 56 : لیست چپ

99 89 : لیست راست

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		
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 حداقل ۲ رقمی بودند)

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