به نام خدا

Subject:
Year: Month: ۲ Day: ۸

حل تمرین سری سوم الگوریتم — مصطفی مضطفی تهری — ۹۸۲۲۸۰۲

۱. برای اثبات ۲ راه داریم:

Fibonacci numbers: 0 1 1 2 3 5 8

راه ۱: ما نیاز به ۲ داده داریم که داده ۱ مقادیر ۰ و ۱ هستند برای $Fib(0)=0$, $Fib(1)=1$

به کمک روند و فرمول و رابطه اصلی: $F(n)=F(n-1)+F(n-2)$

ما می دانیم برای محاسبه بازگشتی یک عدد در سری فیبوناچی به رابطه: $T(n)=T(n-1)+T(n-2)+O(1)$

می رسیم که هربار برای محاسبه هر رقم از دو رقم قبلی استفاده می شود تا به عددهای ۱ و ۰ که در ابتدای سری فیبوناچی تعریف کردیم برسیم و درواقع برای محاسبه به فرمول $x^2=x+1$ می رسیم که طبق حل آن داریم: $x^2=x+1$

هنگامی که معادله را حل کنیم به دو جواب $x=\begin{cases}x_1=\frac{1+\sqrt{5}}{2}\\ x_2=\frac{1-\sqrt{5}}{2}\end{cases}$ می رسیم که با جایگذاری $x^2-x-1=0$

آن در فرمول $F(n)=(x_1)^n+(x_2)^n$ داریم: $F(n)=\left(\left(\frac{1+\sqrt{5}}{2}\right)^n+\left(\frac{1-\sqrt{5}}{2}\right)^n\right)$

حال برای محاسبه Big O واضح است که تابع جداکننده را در نظر بگیریم پس:

$$T(n)=O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

که این عدد، عدد طلایی است و معادل ۱/۶۱۸ می باشد که طبق روابط پیچیدگی زمانی و به کار بردن به سمت بالا به رابطه $O(2^n)$ می رسیم!!

راه ۲:

طبق روابط پیچیدگی زمانی که خواندیم برای محاسبه هر تابع زمانی داریم: $T(n)=T(n-1)+T(n-2)+1$

$$T(n)=(T(n-2)+T(n-3)+1)+(T(n-3)+T(n-4)+1)+1=\dots$$

این رابطه دارای دو عبارت $T(n-1)$ و $T(n-2)$ است که در محاسبه برای n عدد به 2^n می رسیم.

(بخش $+1$ این رابطه به دلیل کوچک بودن نسبت به عدد $T(n-1)+T(n-2)$ در نظر گرفته نمی شود) و رابطه $O(2^n)$ از این رابطه به راحتی استخراج می شود:

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

2. با توجه به الگوریتم داده شده در صورت سوال، می توانیم با یک جابجایی ساده ترتیب مرتب سازی نزولی را به صعودی تبدیل کنیم، برای اسکلرتوفا لازم است ایستگاه ها را طوری مرتب کنیم که ایستگاه ها جای آشوند.

```
PRINT-STATIONS(l, n):
  i = l*
  if (n ≥ 2) {                    //  شرط برگشتن و توقف دادل
      PRINT-STATIONS(li[n-1], n-1) }    //  بازخوانی توابع قبلی
  Print "line" i ", station" j-1
```

با همان سوال حالت را برعکس کنیم

```
PRINT-STATIONS(l, n):
  i = l*
  for j=2 upto n
      do i = li[j]
      Print "line" i ", station" j-1
```

3. اگر به جای استفاده از یک ماتریس جداگانه برای هر l^* و f^* از یک ماتریس استفاده کنیم و فقط موارد لازم را ذخیره کنیم، الگوریتم به تقریب به نصف کاهش می یابد و از $4n+2$ به $2n+2$ می رسد

$A_{5\times10}$ $A_{10\times3}$ $A_{3\times12}$ $A_{12\times5}$ $A_{5\times50}$ $A_{50\times6}$: راه حل استاد

1 $m_{11} = m_{22} = m_{33} = m_{44} = m_{55} = m_{66}$

2
$m_{12} = \min(m_{11} + m_{22} + 5\times10\times3) = 150$
$m_{23} = \min(m_{22} + m_{33} + 10\times3\times12) = 360$
$m_{34} = \min(m_{33} + m_{44} + 3\times12\times5) = 180$
$m_{45} = \min(m_{44} + m_{55} + 12\times5\times50) = 3000$
$m_{66} = \min(m_{55} + m_{66} + 5\times50\times6) = 1500$

$m_{13} = \min\begin{pmatrix} m_{11} + m_{23} + 5\times10\times12 \\ m_{12} + m_{33} + 5\times3\times12 \end{pmatrix} = 330$

$m_{24} = \min\begin{pmatrix} m_{22} + m_{34} + 10\times3\times5 \\ m_{23} + m_{44} + 10\times12\times5 \end{pmatrix} = 330$

$m_{35} = \min\begin{pmatrix} m_{33} + m_{45} + 3\times22\times50 \\ m_{34} + m_{55} + 3\times5\times50 \end{pmatrix} = 930$

$m_{46} = \min\begin{pmatrix} m_{44} + m_{56} + 12\times5\times6 \\ m_{45} + m_{66} + 12\times50\times6 \end{pmatrix} = 1860$

$m_{14} = \min\begin{pmatrix} m_{11} + m_{24} + 5\times10\times5 \\ m_{12} + m_{34} + 5\times3\times50 \\ m_{13} + m_{44} + 5\times12\times5 \end{pmatrix} = 555$

$m_{25} = \min\begin{pmatrix} m_{22} + m_{35} + 10\times3\times50 \\ m_{23} + m_{45} + 10\times12\times50 \\ m_{24} + m_{55} + 10\times5\times50 \end{pmatrix} = 2430$

$m_{36} = \min\begin{pmatrix} m_{33} + m_{46} + 3\times12\times6 \\ m_{34} + m_{56} + 3\times5\times6 \\ m_{35} + m_{66} + 3\times50\times6 \end{pmatrix} = 1830$

$m_{15} = \min\begin{pmatrix} m_{11} + m_{25} + 5\times10\times50 \\ m_{12} + m_{35} + 5\times3\times50 \\ m_{13} + m_{45} + 5\times12\times50 \\ m_{14} + m_{55} + 5\times50\times5 \end{pmatrix} = 1805$

$m_{26} = \min\begin{pmatrix} m_{22} + m_{36} + 10\times3\times6 \\ m_{23} + m_{46} + 10\times12\times6 \\ m_{24} + m_{56} + 10\times5\times6 \\ m_{25} + m_{66} + 10\times50\times6 \end{pmatrix} = 2070$

$m_{16} = \min\begin{pmatrix} m_{11} + m_{26} + 5\times10\times6 \\ m_{12} + m_{36} + 5\times3\times6 \\ m_{13} + m_{46} + 5\times12\times6 \\ m_{14} + m_{56} + 5\times5\times6 \\ m_{15} + m_{66} + 5\times50\times6 \end{pmatrix} = 2070$

$$M = \begin{bmatrix} 0 & 150 & 330 & 555 & 1805 & 2070 \\ & 0 & 360 & 930 & 2430 & 2070 \\ & & 0 & 180 & 930 & 1830 \\ & & & 0 & 3000 & 1860 \\ & & & & 0 & 1500 \\ & & & & & 0 \end{bmatrix}$$

Subject:
Year. Month. Day.

۷- برای حل این سؤال از یک راه حل ساده و سریع استفاده می کنیم:

$$((A_{5\times 10} \times B_{10\times 3})((C_{3\times 12} \times D_{12\times 5}) \times (E_{5\times 50} \times F_{50\times 6})))$$

$G_{5\times 6}$

$H_{3\times 5}$ $G_{5\times 6}$

$I_{5\times 3}$ $(H_{3\times 5}$ $G_{5\times 6})$

$(I_{5\times 3}$ $J_{3\times 6})$

$K_{5\times 6}$

حال هر مرحله بزرگترین ابعاد مشترک ماتریس ها را در نظر گرفته و پرانتز گذاری کرده و ساده می کنیم و همچنین حاصل ضرب آن ها را حساب می کنیم تا کمترین عدد هم بدست بیاید:

و رابطه نهایی به صورت:

$$((A\times B)((C\times D)(E\times F)))$$

$$(5\times \overset{1500}{50}\times 6)+(\overset{180}{3\times 12}\times 5)+(\overset{150}{5\times 10}\times 3)+(\overset{90}{3\times 5}\times 6)+(\overset{90}{5\times 3}\times 6)=\frac{2010}{2}$$

۵- یکی از راه های حل این مسئله علاوه بر راه بازگشتی، برنامه نویسی پویا است، به طور خلاصه می توان این الگوریتم را به صورت زیر توضیح داد:

مراحل زیر را باید انجام داد تا به m_{1n} برسیم:

مرحله ۰: دنباله های ممکن به طول 1، این مرحله معادل $(i=1,2,\dots,n)$ $m_{ii}=0$ است.

مرحله 1: دنباله های ممکن به طول 2، در این مرحله مقادیر $m_{i,i+1}$ $(i=1,2,\dots,n-1)$ را حساب و در جدول می گذاریم

مرحله 2: دنباله های ممکن به طول 3، در این مرحله مقادیر $m_{i,i+2}$ $(i=1,2,\dots,n-2)$ را حساب و در جدول می گذاریم

⋮

مرحله n: دنباله های ممکن به طول $n-1$، در این مرحله m_{1n} را حساب می کنیم.

پس از انجام مراحل به جای اینکه Minimum را در نظر بگیریم maximum را می نویسیم که حداکثر پرانتز گذاری را می توانیم به دست بیاوریم. الگوریتم opt-m:

```
Algorithm opt-mm(S, int n):
Input: Sequence S of n matrices to be multiplied
output: number of operations in an optimal aranethization of S
    for i=1 to n-1 do
        m_ii = 0
    for l=1 to n-1 do
        for i=1 to n-1 do
            j = i+1
            m_ij = ∞
            for k=i to j-1 do
                m_ij = max(m_ik + m_k+1,j + P_i-1 × P_k × P_j)
```

* بررسی این سؤال را برعکس آخر که در درس آمده است حل می کنیم.

P_0	10	$(A_{10\times\underline{20}}\ B_{\underline{20}\times 50})(C_{50\times\underline{1}}\ D_{\underline{1}\times 100})$	con .5
P1	20	$F_{10\times 50}$ $E_{50\times 100}$	
P2	50		
P3	1	$(A\times B)(C\times D)$	
P4	100		

50 00 10000 50 000

$(50\times 1\times 100)+(10\times 20\times 50)+(10\times 50\times 100)=65000$

ওয়াসিম

Subject:
Year. Month. Day.

ک. با توجه به رابطه‌ای که در درس خواندیم داریم:

```
Define  LCS(S1, S2, i, j):
    if((x == -1) or (y == -1))  return 0
    else if(S1[i] == S2[j])  return 1+LCS(S1, S2, i-1, j-1)
    else return max LCS(S1, S2, i-1, j) LCS(S1, S2, i, j-1))
```

حال با استفاده از الگوریتم بالا ماتریس مورد نظر را رسم می‌کنیم:

رشته اول ستون رشته دوم سطر ماتریس

			0	1	0	1	1	0	1	1	0
		-1	0	1	2	3	4	5	6	7	8
	-1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1
0	1	0	1	1	2	2	2	2	2	2	2
0	2	0	1	1	2	2	2	3	3	3	3
1	3	0	1	2	2	3	3	3	4	4	4
0	4	0	1	2	3	3	3	4	4	4	5
1	5	0	1	2	3	4	4	4	5	5	5
0	6	0	1	2	3	4	4	5	5	5	6
1	7	0	1	2	3	4	5	5	6	6	6

LCS: 001010

سطر و ستون اول را همه صفر می‌گذاریم سپس هر سطر را پر می‌کنیم، اگر اعداد زیر رشته آن هر دو از یک نوع باشند مثلاً 0 یا هر دو 1 باشند خانه‌ی شمال غربی را 1+ در آن قرار می‌دهیم، اگر دو خانه متفاوت بودند یعنی یکی 0 و یکی 1 باشند Max خانه‌ی شمال و غرب را در آن می‌گذاریم، این کار را ادامه می‌دهیم تا جدول کامل شود، سپس با استفاده از آخرین خانه به بالا یا چپ رفتن خانه‌های ↖ را در زیر رشته نظر می‌گیریم و زیر رشته مشترک را استخراج می‌کنیم.

Subject:
Year. Month. Day.

۷- همانند سوال قبل عمل می کنیم امّا برای هر n یکبار زیر رشته را در مکانی از حافظه ذخیره می کنیم و حافظه را آزاد می کنیم تا فضای کمتری در حافظه اشغال کنیم، این فضا فضای نسبتاً زیادی است که از حافظه آزاد می شود. همچنین می توان با الگوریتم های مختلف برنامه نویسی این الگوریتم را سریع تر و بهینه تر کنیم، همچنین برای محاسبه خانه های اطراف قطر نیاز است، می توانیم خانه های دیگر را محاسبه کنیم. در این صورت فضای مورد نیاز $2 \times \min(m, n)$ خواهد بود.

۸- با استفاده از برنامه نویسی پویا این سوال را حل می کنیم. ابتدا بزرگترین دنباله تقسیم را با هر عنصر به پایان می رسانیم و در نهایت حداکثر را برمی گردانیم.

$$d_i = \max_{\substack{i<j \\ a_j > a_i}} (d_j + 1)$$

lds = longest dividing subsequnce

Python:

```
def lds(arr, n):
    lds = [0 for i in range(n)]
    lds[0] = 1
    for i in range(n):
        lds[i] = 1
        for j in range(i):
            if (lds[j] != 0 and
                arr[i] % arr[j] == 0):
                lds[i] = max(lds[i], lds[j]+1)
    return max(lds)
```

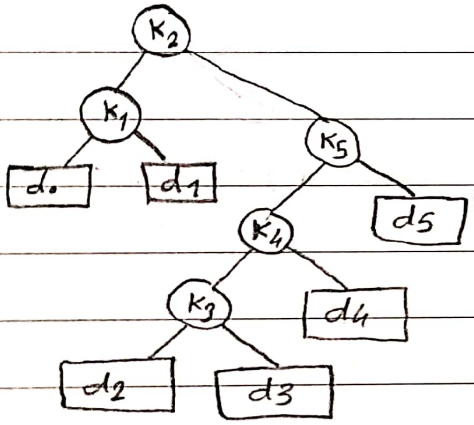
همچنین الگوریتم دیگر آن برای محاسبه LCS: همانند سؤالات قبلی:

```
LCS(a, y, i, j)
if((i == 0) || (j == 0))
    return ;
if(a[i,j] == a[i-1, j-1])
    return LCS(a, y, i-1, j-1)
    Cout<<x;
```

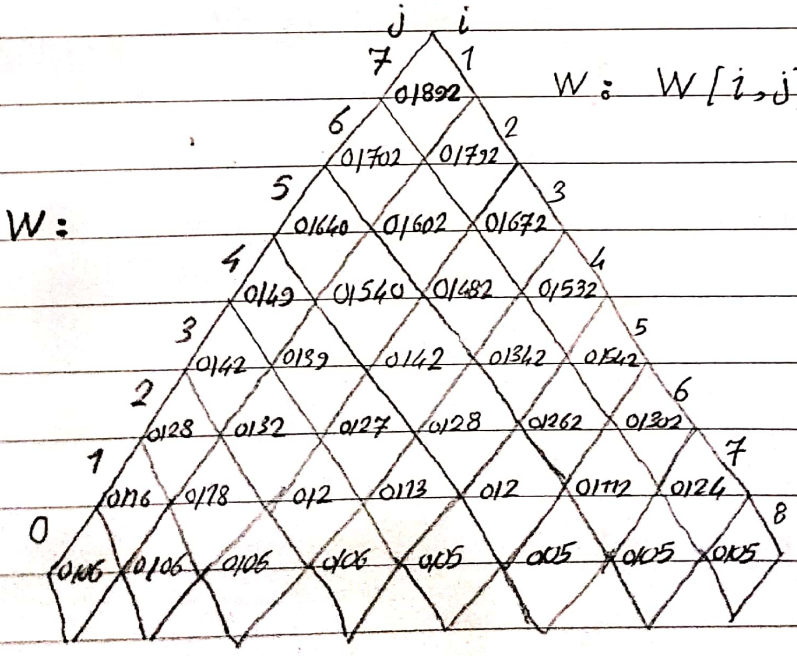
Subject:
Year. Month. Day.

۹. می‌توانیم از الگوریتم Construct_optimal_BST موجود در اسلایدهای استاد و اینترنت استفاده کنیم.

```
Construct_optimal_BST(root, i, j, last):
    if (i == j)        return null
    if (last == 0)     Print root[i,j] + " is the root"
    else if (j < last) Print root[i,j] + " is the left child of " + last
    else               Print root[i,j] + " is the right child of " + last
    Construct_optimal_BST(root, i, root[i,j] - 1, root[i,j])
    Construct_optimal_BST(root, root[i,j] + 1, j, root[i,j])
```



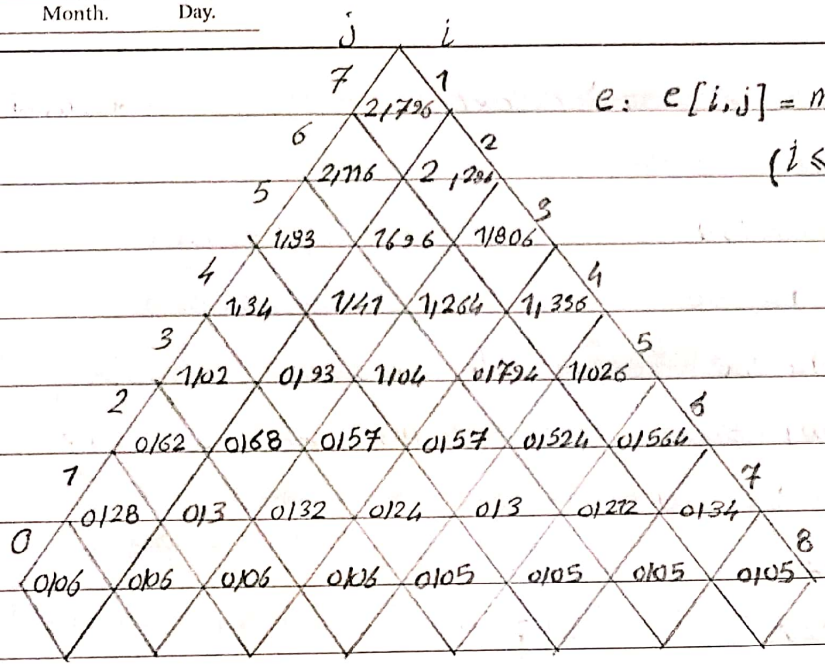
10



Subject:

Year. Month. Day.

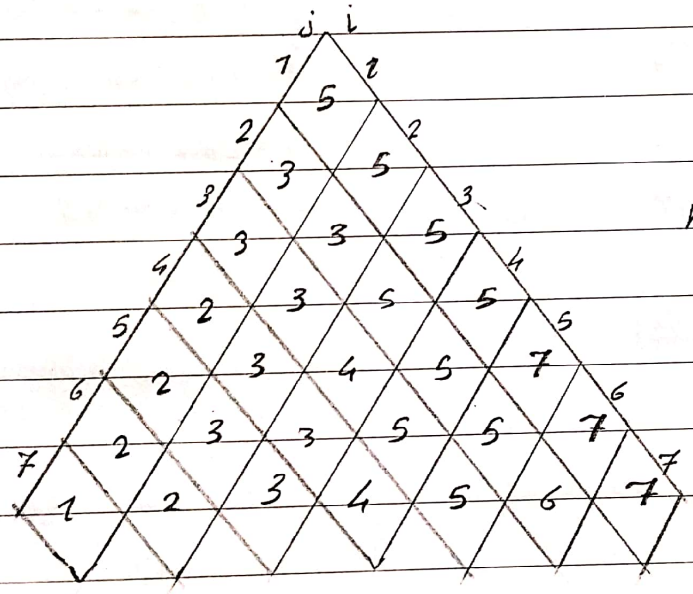
e:



$e: e[i,j] = \min(e[i,r-1] + e[r+1,j] + w[i,j])$

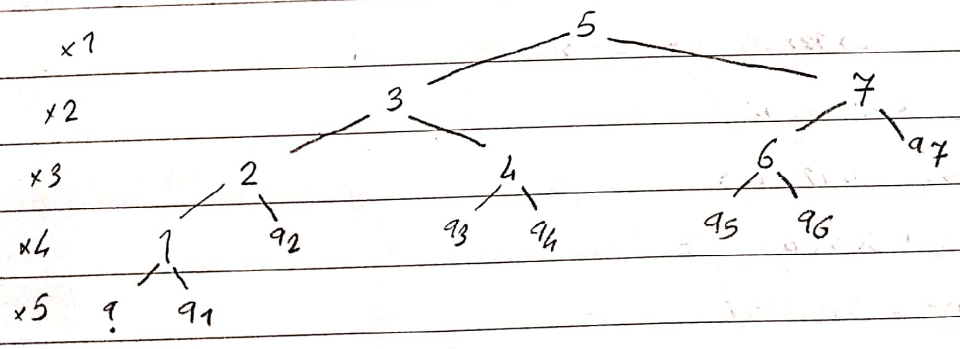
$(i \le r \le j)$

root



$root: root[i,j] = r$

رسم نمودار:



$1\times(0.7) + 2\times(0.22) + 3\times(0.142) + 4\times(0.37) + 5\times(0.72) = 2.796$

0/44 0.426 1/24 0/6

* از محاسبات برای به دست آوردن هرخانه ماتریس های e، w و root فاکتور گرفته شده است و تنها جواب نهایی درخانه ها قرار داده شده است.