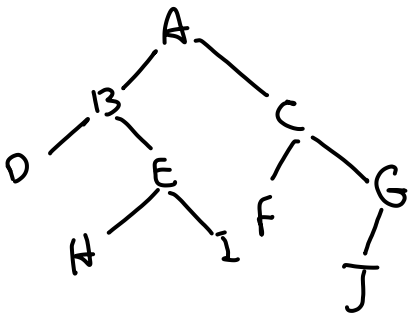
۳۳- می‌خواهیم با وارد کردن مقادیر ۱ و ۲ و ۳ به هر ترتیب دلخواه در یک درخت تهی دودویی جستجو (Null Binary Search Tree) یک درخت دودویی جستجو با ۳ گره بسازیم. چند درخت دودویی جستجوی متفاوت ممکن است ساخته شود؟ (مهندسی کامپیوتر ـ دولتی ۷۶)

۱) ٤ درخت ۲) ۳ درخت ۳) ٦ درخت ٤) ٥ درخت

$\frac{1}{n+1}\binom{2n}{n} = \frac{1}{4}\binom{6}{3}$

۳۸- اگر T یک درخت جستجوی باینری به صورت زیر باشد که در هر گره آن یک عدد صحیح ذخیره شده است چهارمین کوچکترین عنصر آن در کدام گره قرار دارد؟ (علوم کامپیوتر ـ دولتی ۸٦)

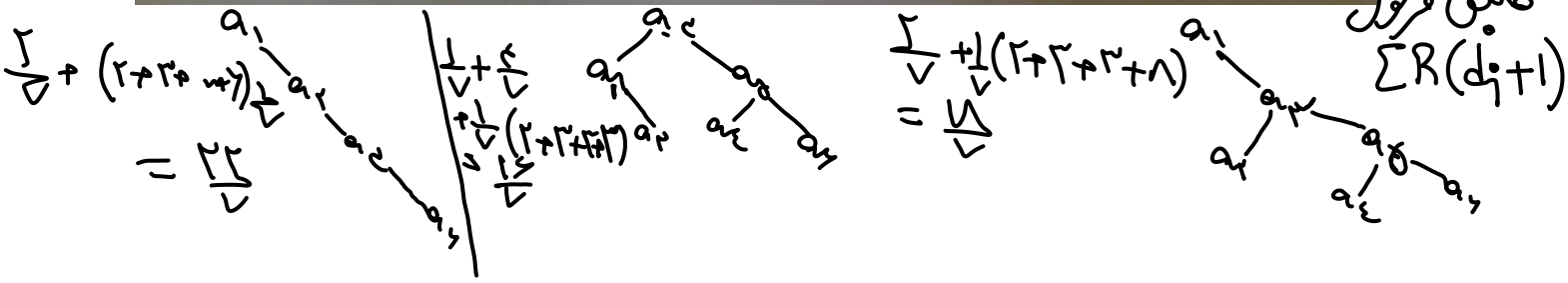


D B H E I A F C J G

۴۷- می‌خواهیم یک درخت دودویی جستجو با عنصر $a_6 > \ldots > a_2 < a_1$ بسازیم تا متوسط عمق عناصر در آن کمینه شود. اگر p_i احتمال a_i باشد، متوسط عمق برابر $\sum_{i=1}^{6} p_i \text{depth}(q_i)$ تعریف می‌شود. اگر $p_1 = \frac{2}{7}$ و $p_i = \frac{1}{7}$ برای $2 \le i \le 6$ باشد، متوسط عمق درخت بهینه چند است؟ (عمق ریشه صفر فرض می‌شود)

(علوم کامپیوتر ـ دولتی ۸۲)

۱) $\frac{8}{7}$ ۲) $\frac{9}{7}$ ۳) $\frac{10}{7}$ ٤) $\frac{11}{7}$



طولانی ترین زیردنباله مشترک:

در برخی کاربردهای بیولوژیکی لازم استDNA چند ارگانیسم با هم مقایسه شود. یک DNA از زنجیره ای از مولکول های پایه ای تشکیل شده است. برخی از آنها عبارتند از آدنین(A)، گوانین(G) ، سیتوسین(S) و تیمین(T). بنابراین یک رشته را می توان کلمه ای روی {A,G,S,T} در نظر گرفت. مثلا

S_1 = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA

S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAA

یکی از اهداف مقایسه دو رشته تعیین میزان همانندی آنهاست. لذا دنبال رشته سومی هستیم که مولکول پایه آن در هر دو رشته ظاهر شده باشد ولی لزومی ندارد متوالی باشند. در مثال فوق رشته مشترک بصورت زیر است:

GTCGTCGGAAGCCGGCCGAA

پیشوند i ام X: i حرف پشت سر هم از ابتدای رشته X را پیشوند i ام X گویند و با X_i نشان میدهند.

حرف i ام: h امین حرف رشته X را با x_i نشان میدهند.

$$X =< x_1,\cdots,x_m >$$
$$X_i =< x_1,\cdots,x_i >$$
$$Z =< z_1,\cdots,z_k >$$

زیر دنباله: گوییم z زیردنباله‌ای از X است اگر:

$$\exists i_1,i_2,\cdots,i_k \mid i_1 < i_2 < \cdots < i_k : \forall j : 1 \le j \le k \; z_j = x_{i_j}$$

$$X=<A,B,B,A,D,A,B,F>$$

$$Z= <B,A,A,F>$$

$$\begin{cases} i_1 = 2 \Rightarrow z_1 = x_2 \\ i_2 = 4 \Rightarrow z_2 = x_4 \\ i_3 = 6 \Rightarrow z_3 = x_6 \\ i_4 = 8 \Rightarrow z_4 = x_8 \end{cases}$$

نکته: هر رشته m حرفی دارای 2^m زیردنباله است.

دراین مسئله در ورودی دو رشته به طول‌های n,m وجود دارد. که هدف مسئله بدست آوردن طولانی‌ترین زیردنباله مشترک می‌باشد.

ورودی:

دو دنباله X و Y که اولی به طول m و دومی به طول n میباشد.

خروجی: طول طولانی ترین زیر دنباله مشترک X و Y

رویکرد برنامه ریزی پویا:

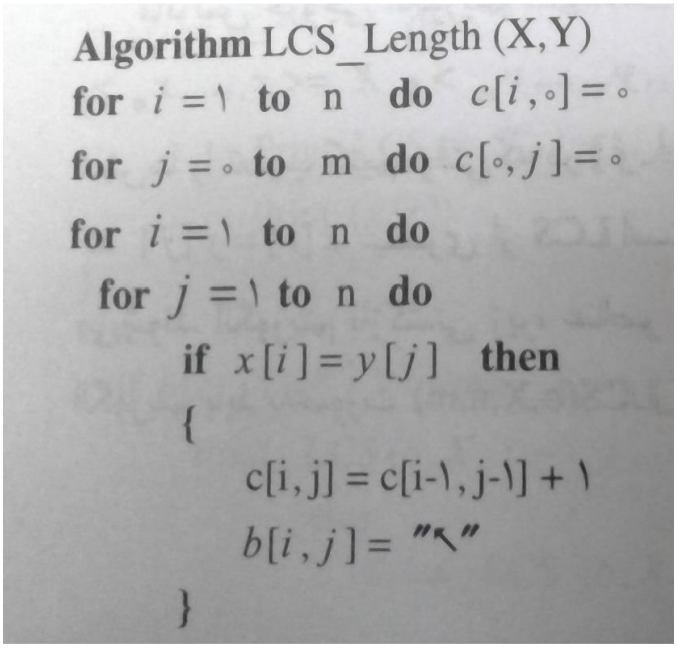
۱- طول طولانی ترین زیر دنباله مشترک X_i و Y_j = c[i,j]

۲- $\forall i,j$; c[i,0]=c[0,j]=0 (بدیهی)(وقتی یکی از رشته‌ها تهی باشد هیچ اشتراکی وجود نخواهد داشت)

۳- جواب = c[m,n]

۴- رابطه بازگشتی:

$$c[i,j]=\begin{cases} c[i-1,j-1]+1 & ;x_i = y_j \\ \max\{c[i-1,j],c[i,j-1]\} & ;else \end{cases}$$



```
Algorithm LCS_Length (X,Y)
for i = ۱ to n do c[i,۰] = ۰
for j = ۰ to m do c[۰,j] = ۰
for i = ۱ to n do
  for j = ۱ to n do
      if x[i] = y[j] then
      {
          c[i,j] = c[i-۱,j-۱] + ۱
          b[i,j] = "↖"
      }
```

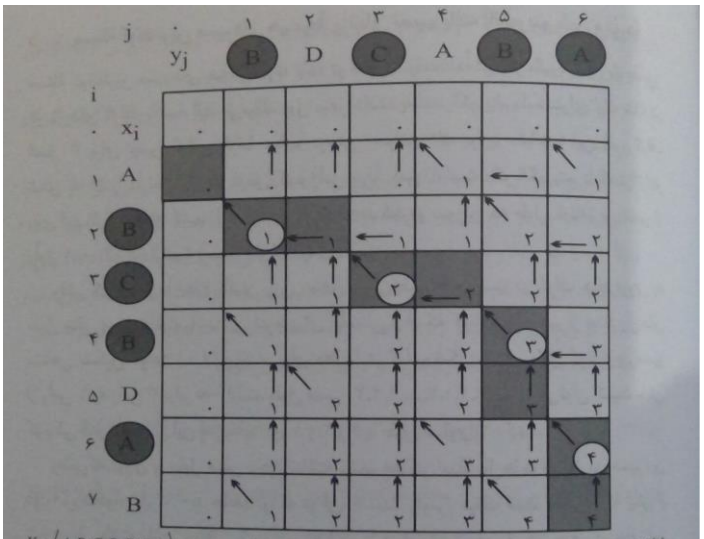
```
        else
        if  c[i-1, j] ≥ c[i, j-1]   then
        {
            c[i, j] = c[i-1, j]
            b[i, j] = "↑"
        }
        else
        }
            c[i, j] = c[i, j -1]
            b[i, j] = "←"
        }
Returen  c  and  b
```

پیچیدگی زمانی این الگوریتم $\theta(nm)$ است.

مثال۵-۲۶ مسئلهٔ LCS را برای دو رشتهٔ $X=\langle A, B, C, B, D, A, B\rangle$ و $Y=\langle B, D, C, A, B, A\rangle$ حل کنید.

حل. خلاصهٔ محاسبات در شکل ۵-۱۸ آمده است. در سطر **i** و ستون **j** این جدول، مقدار $c[i,j]$ و نیز اشاره‌گر مربوط به $b[i,j]$ ذخیره شده‌اند. مقدار ۴ در $c[۷,۶]$، طول **LCS** دو رشتهٔ **X** و **Y** است. $LCS(X,Y)=<B,C,B,A>$. برای به‌دست آوردن $LCS(X,Y)$ کافی است از $b[۷,۶]$ شروع کرده و مسیر فلشها را تعقیب کنیم. در شکل زیر این مسیر به‌صورت خاکستری نمایش داده شده است. هر فلش "↖" که با دایرهٔ کوچک مشخص شده است بیانگر حالت $x[i]=y[j]$ است و معرف عنصری از **LCS** است.

ماتریس خروجیِ الگوریتم LCS_Length را به سهولت می‌توان برای تشکیل LCS(X,Y)، $X=<x_1,...,x_n>$ و $Y=<y_1,y_2,...,y_m>$ به‌کار برد. کافی است از $b[n,m]$ شروع و فلش‌ها را تعقیب کنیم. وقتی‌که در $b[i,j]$ به علامت "↖" برخورد می‌کنیم مفهومش این است که $x[i]=y[j]$ عنصری از LCS است. در این روش، عناصر LCS به‌ترتیب معکوس ملاقات می‌شوند. الگوریتم بازگشتی زیر، عناصر LCS را به‌ترتیب صحیح آن چاپ می‌کند. فراخوانی این الگوریتم باید به‌صورت Print_LCS(b,X,n,m) باشد.



شکل۵-۱۸ ماتریس‌های c و b محاسبه شده با الگوریتم LCS_Length برای رشته‌های $X=\langle A,B,C,B,D,A,B\rangle$ و $Y=\langle B,D,C,A,B,A\rangle$. سطر i و ستون j این جدول شامل مقدار $c[i,j]$ و فلش مربوط به $b[i,j]$ است. عدد ۴ واقع در $c[۷,۶]$ طول $LCS(X,Y)=<B,C,B,A>$ است. برای تشکیل عناصر LCS کافی است فلشهای $b[i,j]$ را از $b[۷,۴]$ تعقیب کنیم. در شکل بالا این مسیر به‌صورت خاکستری است. هر "↖" در این مسیر حالت $x_i=y_j$ را نشان می‌دهد که عنصری از LCS است.

```
Algorithm Print_LCS ( b, X, i, j )
if i = ۰ or j = ۰ then return
if b[i, j] = "↖" then
{
        Print_LCS (b, X, i − ۱, j − ۱)
        Print (x[i])
}
else
  if b[i, j] = "↑" then
      Print_ LCS (b, X, i − ۱, j)
          else
              Print_ LCS (b, X, i, j − ۱)
```