3.3.3 最长公共子序列 LCS

相关概念

X 的子序列 Z ： 设序列 X, Z,

$$X = < x_1, x_2, ..., x_m >$$

$$Z = < z_1, z_2, ..., z_k >$$

若存在 X 的元素构成的严格递增序列 $< x_{i_1}, x_{i_2}, ..., x_{i_k} >$

使得 $z_j = x_{i_j}, j = 1,2,...,k,$ 则称 Z 是 X 的子序列

X 与 Y 的公共子序列 Z：Z 是 X 和 Y 的子序列

子序列的长度：子序列的元素个数

问题描述

给定序列 $X=<x_1, x_2, \ldots, x_m>$，$Y=<y_1, y_2, \ldots, y_n>$

求 X 和 Y 的最长公共子序列

实例

X: A B C B D A B

Y: B D C A B A

最长公共子序列: B C B A

蛮力算法：检查 X 的每个子序列在Y 中出现

每个子序列 $O(n)$ 时间，X 有 2^m 个子序列，最坏情况下时间复杂度：$O(n2^m)$

子问题划分及依赖关系

子问题边界： X和Y起始位置为1，X的终止位置是 i，Y的终止位置是j，记作

$$X_i=<x_1,x_2,\ldots,x_i>,\quad Y_j=<y_1,y_2,\ldots,y_j>$$

依赖关系：

$$X=<x_1,x_2,\ldots,x_m>,\ Y=<y_1,y_2,\ldots,y_n>,\ Z=<z_1,z_2,\ldots,z_k>,$$

Z 为 X 和 Y 的 LCS，那么

(1) 若$x_m=y_n \Rightarrow z_k=x_m=y_n$,且$Z_{k-1}$是$X_{m-1}$与$Y_{n-1}$的 LCS；

(2) 若$x_m\neq y_n$, $z_k\neq x_m \Rightarrow Z$是$X_{m-1}$与$Y$ 的 LCS；

(3) 若$x_m\neq y_n$, $z_k\neq y_n \Rightarrow Z$是$X$与$Y_{n-1}$的 LCS.

满足优化原则和子问题重叠性

递推方程、标记函数

令 X 与 Y 的子序列

$$X_i = \langle x_1, x_2, \ldots, x_i \rangle, \quad Y_j = \langle y_1, y_2, \ldots, y_j \rangle$$

$C[i,j]$： X_i 与 Y_j 的 LCS 的长度

递推方程

$$C[i,j] = \begin{cases} 0 & \text{若 } i = 0 \text{ 或 } j = 0 \\ C[i-1, j-1] + 1 & \text{若 } i, j > 0, x_i = y_j \\ \max\{C[i, j-1], C[i-1, j]\} & \text{若 } i, j > 0, x_i \neq y_j \end{cases}$$

标记函数：$B[i, j]$，其值为字符↖、←、↑，分别表示$C[i,j]$取得最大值时的三种情况

算法3.4 LCS(X,Y,m,n)

1. for $i \leftarrow 1$ to m do　　　//行1-4边界情况
2. 　　$C[i,0] \leftarrow 0$
3. for $i \leftarrow 1$ to n do
4. 　　$C[0,i] \leftarrow 0$
5. for $i \leftarrow 1$ to m do
6. 　　for $j \leftarrow 1$ to n do
7. 　　　　if $X[i]=Y[j]$
8. 　　　　then $C[i,j] \leftarrow C[i-1,j-1]+1$
9. 　　　　　　$B[i,j] \leftarrow$ '↖'
10. 　　　　else if $C[i-1,j] \geq C[i,j-1]$
11. 　　　　　　then $C[i,j] \leftarrow C[i-1,j]$
12. 　　　　　　　　$B[i,j] \leftarrow$ '↑'
13. 　　　　　　else $C[i,j] \leftarrow C[i,j-1]$
14. 　　　　　　　　$B[i,j] \leftarrow$ '←'

利用标记函数构造解

算法 Structure Sequence(B, i, j)

输入：$B[i,j]$

输出：X与Y的最长公共子序列

1. if i=0 or j=0 then return //一个序列为空
2. if $B[i,j]$ = "↖"
3. then 输出$X[i]$
4. Structure Sequence(B, $i-1$, $j-1$)
5. else if $B[i,j]$= "↑" then Structure Sequence (B, $i-1$, j)
6. else Structure Sequence (B, i, $j-1$)

算法的计算复杂度

计算优化函数和标记函数：时间为$O(mn)$

构造解：每一步至少缩小X 或 Y 的长度，时间$\Theta(m+n)$

空间：$\Theta(mn)$

实例

输入：X=<A,B,C,B,D,A,B>, Y=<B,D,C,A,B,A>,

标记函数：

	1	2	3	4	5	6
1	$B[1,1]=\uparrow$	$B[1,2]=\uparrow$	$B[1,3]=\uparrow$	$B[1,4]=\nwarrow$	$B[1,5]=\leftarrow$	$B[1,6]=\nwarrow$
2	$B[2,1]=\nwarrow$	$B[2,2]=\leftarrow$	$B[2,3]=\leftarrow$	$B[2,4]=\uparrow$	$B[2,5]=\nwarrow$	$B[2,6]=\leftarrow$
3	$B[3,1]=\uparrow$	$B[3,2]=\uparrow$	$B[3,3]=\nwarrow$	$B[3,4]=\leftarrow$	$B[3,5]=\uparrow$	$B[3,6]=\uparrow$
4	$B[4,1]=\uparrow$	$B[4,2]=\uparrow$	$B[4,3]=\uparrow$	$B[4,4]=\uparrow$	$B[4,5]=\nwarrow$	$B[4,6]=\leftarrow$
5	$B[5,1]=\uparrow$	$B[5,2]=\uparrow$	$B[5,3]=\uparrow$	$B[5,4]=\uparrow$	$B[5,5]=\uparrow$	$B[5,6]=\uparrow$
6	$B[6,1]=\uparrow$	$B[6,2]=\uparrow$	$B[6,3]=\uparrow$	$B[6,4]=\nwarrow$	$B[6,5]=\uparrow$	$B[6,6]=\nwarrow$
7	$B[7,1]=\uparrow$	$B[7,2]=\uparrow$	$B[7,3]=\uparrow$	$B[7,4]=\uparrow$	$B[7,5]=\uparrow$	$B[7,6]=\uparrow$

解：$X[2]$,$X[3]$, $X[4]$, $X[6]$, 即 B, C, B, A

3.3.4 图像压缩

像素点灰度值：0~255，表示为 8 位二进制数

像素点灰度值序列：$<p_1, p_2,\ldots, p_n>$，p_i为第 i个像素点灰度值

变位压缩存储格式：将$<p_1, p_2,\ldots, p_n>$分割 m 段 $S_1, S_2, \ldots, S_m$

i 段有$l[i]$个像素，每个像素 $b[i]$位，h_i 为 该段最大像素的位数

$$h_i \le b[i] \le 8, \quad h_i = \left\lceil \log(\max_{p_k \in S_i} p_k + 1) \right\rceil$$

约束条件：每段像素个数 $l[i] \le 256$

段头11位： $b[i]$的二进制表示(3 位) + $l[i]$的二进制表示(8位)

i 段占用空间：$b[i] \times l[i] + 11$

问题：给定像素序列$<p_1, p_2, \ldots, p_n>$，确定最优分段，即

$$\min_T \{\sum_{i=1}^{j} (b[i] \times l[i] + 11)\}, \quad T = \{S_1, S_2, \ldots, S_j\}为分段$$

北京大学

实例

灰度值序列 P=<10,12,15,255,1,2,1,1,2,2,1,1>

分法1： S_1=<10,12,15>， S_2=<255>, S_3=<1,2,1,1,2,2,1,1>

分法2： S_1=<10,12,15,255,1,2,1,1,2,2,1,1>

分法3： 分成12组，每组一个数

存储空间

分法1：$11\times3+4\times3+8\times1+2\times8=69$

分法2：$11\times1+8\times12=107$

分法3：$11\times12+4\times3+8\times1+1\times5+2\times3=163$

结论：分法1是其中最优的分法

递推方程

设$s[i]$是像素序列$<p_1, p_2, \ldots, p_i>$的最优分段所需存储位数

$$s[i] = \min_{1 \le j \le \min\{i,256\}} \{s[i-j] + j \times b[i-j+1, i] + 11\}$$

$$b[i-j+1, i] = \left\lceil \log(\max_{p_k \in S_m} p_k + 1) \right\rceil \le 8$$

P_1 P_2 $\ldots$ P_{i-j}	P_{i-j+1} $\ldots$ P_i
P_{i-j}	j个灰度

算法

Compress (*P*,*n*) //计算最小位数*S*[*n*]

```
1. Lmax←256; header←11; S[0]←0   //最大段长Lmax, 头header
2. for  i←1  to  n  do
3.     b[i]←length(P[i])          //b[i]是第i个灰度P[i]的二进制位数
4.     bmax←b[i]                  //3-6行分法的最后一段只有P[i]自己
5.     S[i]←S[i–1]+bmax
6.     l[i]←1
7.     for  j←2  to  min{i,Lmax}  do    //最后段含 j 个像素
8.         if  bmax<b[i–j+1]            //统一段内表示像素的二进制位数
9.         then  bmax←b[i–j+1]
10.        if  S[i]>S[i–j]+j*bmax
11.        then S[i]←S[i–j]+j*bmax
12.             l[i]←j
13.  S[i]←S[i]+header
```

时间复杂度 $T(n)=O(n)$

$P = \langle 10, 12, 15, 255, 1, 2 \rangle.$

$S[1]=15,\ S[2]=19,\ S[3]=23,\ S[4]=42,\ S[5]=50$

$l[1]=1,\ l[2]=2,\ l[3]=3,\ l[4]=1,\ l[5]=2$

实例

10	12	15	255	1	2

$S[5]=50$　　$1\times 2+11$

10	12	15	255	1	2

$S[4]=42$　　$2\times 2+11$

10	12	15	255	1	2

$S[3]=23$　　$3\times 8+11$

10	12	15	255	1	2

$S[2]=19$　　$4\times 8+11$

10	12	15	255	1	2

$S[1]=15$　　$5\times 8+11$

10	12	15	255	1	2

$6\times 8+11$

追踪解

算法 Traceback(n,l)

输入：数组l

输出：数组C　　// $C[j]$是从后向前追踪的第 j段的长度

1. $j \leftarrow 1$　　// j为正在追踪的段数
2. while $n \neq 0$ do
3. 　$C[j] \leftarrow l[n]$
4. 　$n \leftarrow n - l[n]$
5. 　$j \leftarrow j+1$

时间复杂度：$O(n)$

3.3.5 最大子段和

问题：给定n 个整数（可以为负数）的序列

$(a_1, a_2, \ldots, a_n)$

求 $\max\{0, \max_{1\le i\le j\le n} \sum_{k=i}^{j} a_k\}$

实例： (-2, 11, -4, 13, -5, -2)

解：最大子段和 $a_2+a_3+a_4= 20$

算法1---顺序求和+比较

算法2---分治策略

算法3---动态规划

算法1 顺序求和+比较

算法 Enumerate

输入：数组$A[1..n]$

输出：*sum, first, last*

```
sum←0
for i←1 to n do            // i为当前和的首位置
    for j←i to n do        // j为当前和的末位置
        thissum←0          // thissum为A[i]到A[j]之和
        for k←i to j do
            thissum←thissum+A[k]
        if thissum > sum
        then sum←thissum
            first←i        // 记录最大和的首位置
            last←j         // 记录最大和的末位置
```

时间复杂度：$O(n^3)$

算法2 分治策略

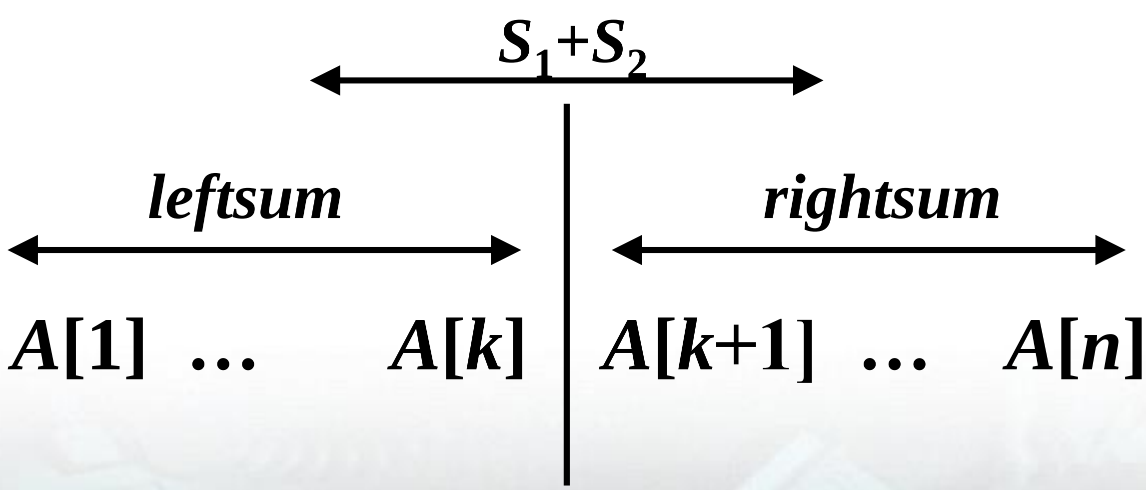
将序列分成左右两半，中间分点*center*

递归计算左段最大子段和 *leftsum*

递归计算右段最大子段和 *rightsum*

$a_{center} \rightarrow a_1$的最大和$S_1$，$a_{center+1} \rightarrow a_n$的最大和$S_2$

$\max\{ leftsum, rightsum, S_1+S_2\}$



分治算法

算法 MaxSubSum(*A*, *left*, *right*）

输入：数组*A*，*left*，*right*分别是*A*的左、右边界
输出：*A*的最大子段和*sum*及其子段的前后边界

1. if $|A|=1$ then 输出元素值（当值为负时输出0）
2. $center \leftarrow \lfloor (left+right)/2 \rfloor$
3. $leftsum \leftarrow \mathrm{MaxSubSum}(A, left, center)$ //子问题A_1
4. $righsum \leftarrow \mathrm{MaxSubSum}(A, center+1, right)$ //子问题A_2
5. $S_1 \leftarrow$从$A[center]$向左的最大和 //从*center*向左的最大和
6. $S_2 \leftarrow$从$A[center+1]$向右的最大和 //从*center*+1向右的最大和
7. $sum \leftarrow S_1+S_2$
8. if $leftsum > sum$ then $sum \leftarrow leftsum$
9. if $rightsum > sum$ then $sum \leftarrow rightsum$

时间：$T(n)=2T(n/2)+O(n)$, $T(c)=O(1)$
$T(n)=O(n\log n)$

算法3：动态规划

令$C[i]$ 是 $A[1..i]$中必须包含元素$A[i]$的最大子段和

$$C[i] = \max_{1 \le k \le i} \{ \sum_{j=k}^{i} A[j] \}$$

$A[1]$ $A[2]$ $A[3]$ … $A[i\text{-}1]$ | $A[i]$ | … $A[n]$

最大子段和 $C[i\text{-}1]$

最大子段和 $C[i]$

递推方程: $C[i] = \max\{C[i-1]+A[i], A[i]\}$ $i=1,\ldots,n$

$C[0]=0$

解: $$\text{OPT}(A) = \max_{0 \le i \le n} \{C[i]\}$$

算法 MaxSum

算法3.10 MaxSum(A, n)

输入：数组A

输出：最大子段和sum, 子段的最后位置c

1. $sum \leftarrow 0$
2. $b \leftarrow 0$　　// b是前一个最大子段和
3. for $i \leftarrow 1$ to n do
4. 　if $b>0$
5. 　then $b \leftarrow b+A[i]$
6. 　else $b \leftarrow A[i]$
7. 　if $b>sum$
8. 　then $sum \leftarrow b$
9. 　　$c \leftarrow i$　　// 记录最大和的末项标号
10. return sum，c

时间复杂度$O(n)$；空间复杂度$O(n)$

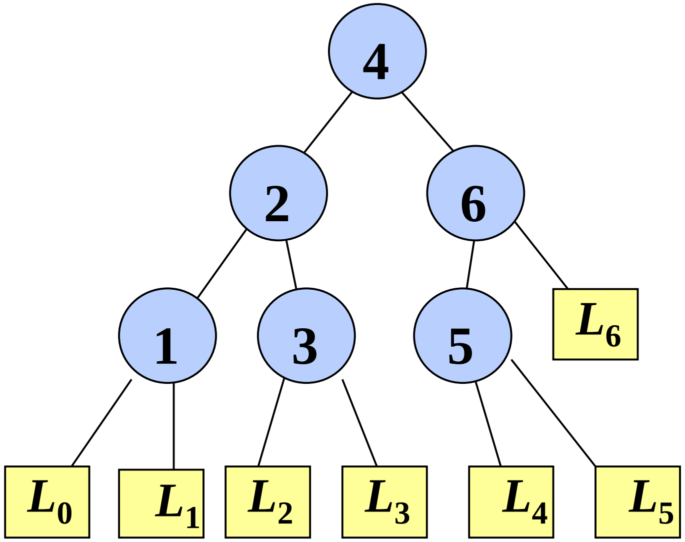
北京大学

3.3.6 最优二叉检索树

设集合 *S* 为排序的 *n* 个元素 $x_1<x_2<\ldots<x_n$，将这些元素存储在一棵二叉树的结点上，以查找 *x* 是否在这些数中. 如果 *x* 不在，确定 *x* 在那个空隙.

检索方法:

1. 初始，*x*与根元素比较;
2. *x*<根元素，递归进入左子树;
3. *x*>根元素，递归进入右子树;
4. *x*=根元素，算法停止，输出*x*;
5. *x*到达叶结点，算法停止，输出*x*不在数组中.



S=< 1, 2, 3, 4, 5, 6>

存取概率不等情况

空隙： $(-\infty, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, +\infty)$,

$x_0 = -\infty, \quad x_{n+1} = \infty$

给定序列 $S = <x_1, x_2, \ldots, x_n>$,

x 在 x_i 的概率为b_i， x 在 (x_i, x_{i+1}) 的概率为a_i，

S的存取概率分布如下：

$$P = <a_0, b_1, a_1, b_2, a_2, \ldots, b_n, a_n>$$

实例

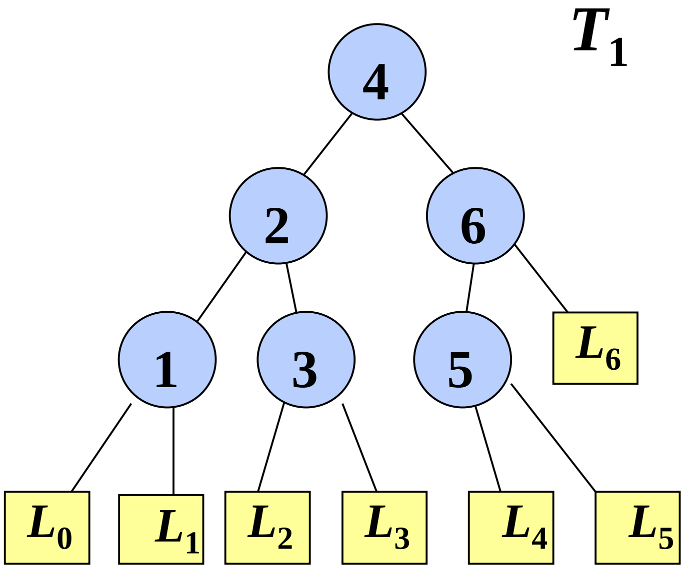
S=<1,2,3,4,5,6>

P=<0.04,0.1,0.01,0.2,0.05,0.2,0.02,0.1,0.02,0.1,0.07, 0.05,0.04>

实例

S = < 1, 2, 3, 4, 5, 6 >

P = < 0.04, 0.1, 0.01, 0.2, 0.05, 0.2,
0.02, 0.1, 0.02, 0.1, 0.07, 0.05,
0.04 >

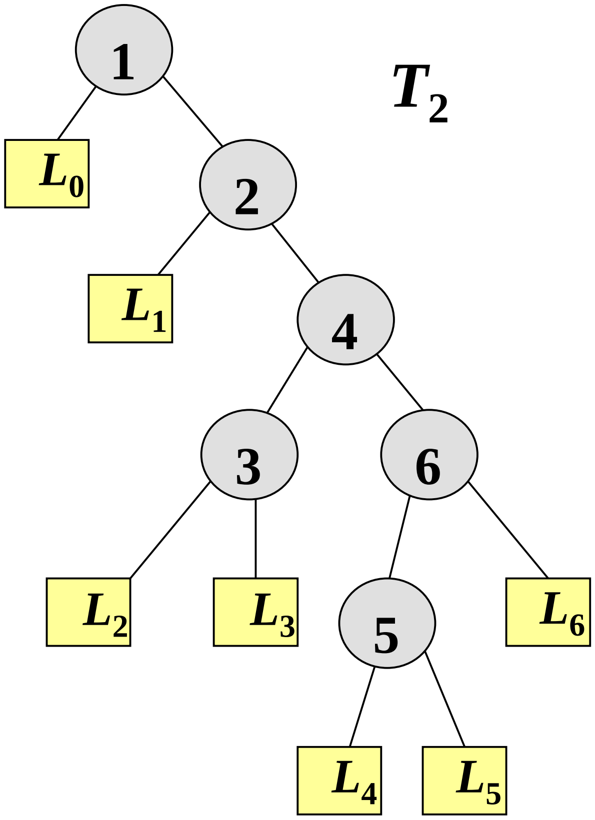


$$m(T_1)=[1*0.1+2*(0.2+0.05)+3*(0.1+0.2+0.1)]$$
$$+[3*(0.04+0.01+0.05+0.02+0.02+0.07)+2*0.04]$$
$$= 1.8+0.71=2.51$$

实例

$S = <1, 2, 3, 4, 5, 6>$

$P = <0.04, 0.1, 0.01, 0.2, 0.05, 0.2, 0.02, 0.1, 0.02, 0.1, 0.07, 0.05, 0.04>$



$$
\begin{aligned}
m(T_2) = & [\, 1*0.1+2*0.2 + 3*0.1 \\
& + 4*(0.2+0.05) + 5*0.1 \,] \\
& + [\, 1*0.04 + 2*0.01 \\
& + 4*(0.05 + 0.02 + 0.04) \\
& + 5*(0.02 + 0.07)] \\
= & \ 2.3 + 0.95 = 3.25
\end{aligned}
$$

问题

数据集 $S=<x_1,x_2,\ldots,x_n>$

存取概率分布 $P=<a_0, b_1, a_1, b_2, \ldots, a_i, b_{i+1}, \ldots, b_n, a_n>$

结点 x_i 在T 中的深度是 $d(x_i)$, $i=1,2,\ldots,n$,

空隙 L_j 的深度为 $d(L_j)$, $j=0,1,\ldots,n$,

平均比较次数为:

$$t=\sum_{i=1}^{n} b_i(1+d(x_i))+\sum_{j=0}^{n} a_j d(L_j)$$

问题：给定数据集 S 和相关存取概率分布 P，求一棵最优的(即平均比较次数最少的) 二分检索树.

算法设计:子问题划分

$S[i,j] = \langle x_i, x_{i+1}, \ldots, x_j \rangle$ 是S 以 i 和 j 作为边界的子数据集

$P[i,j] = \langle a_{i-1}, b_i, a_i, b_{i+1}, \ldots, b_j, a_j \rangle$是对应$S[i,j]$存取概率分布

例：$S=\langle A, B, C, D, E \rangle$

$P=\langle 0.04, \underline{0.1}, 0.02, \underline{0.3}, 0.02, \underline{0.1}, 0.05, \underline{0.2}, 0.06, \underline{0.1}, 0.01 \rangle$

$S[2,4]=\langle B, C, D \rangle$

$P[2,4]=\langle 0.02, \underline{0.3}, 0.02, \underline{0.1}, 0.05, \underline{0.2}, 0.06 \rangle$

子问题划分：以 x_k 作为根，划分成两个子问题：

$S[i,k-1], P[i,k-1]$

$S[k+1,j], P[k+1,j]$

例：以B为根，划分成以下子问题：

$S[1,1]=\langle A \rangle$，$P[1,1]=\langle 0.04, \underline{0.1}, 0.02 \rangle$

$S[3,5]=\langle C,D,E \rangle$, $P[3,5]=\langle 0.02, \underline{0.1}, 0.05, \underline{0.2}, 0.06, \underline{0.1}, 0.01 \rangle$

北京大学

递推方程

设 $m[i,j]$ 是相对于输入 $S[i,j]$ 和 $P[i,j]$ 的最优二叉搜索树的平均比较次数，令

$$w[i,j]=\sum_{p=i-1}^{j}a_p+\sum_{q=i}^{j}b_q$$

是 $P[i,j]$ 中所有概率（包括数据元素与空隙）之和

递推方程：

$$m[i,j]=\min_{i\le k\le j}\{m[i,k-1]+m[k+1,j]+w[i,j]\},\quad 1\le i\le j\le n$$

$$m[i,i-1]=0,\quad i=1,2,\ldots,n$$

证明

$m[i,j]_k$：根为 x_k 时的二分检索树平均比较次数的最小值

$$
\begin{aligned}
& m[i,j]_k \\
&= (m[i,k-1]+w[i,k-1]) + (m[k+1,j]+w[k+1,j]) + 1\times b_k \\
&= (m[i,k-1]+m[k+1,j]) + (w[i,k-1]+b_k+w[k+1,j]) \\
&= (m[i,k-1]+m[k+1,j]) + (\sum_{p=i-1}^{k-1} a_p + \sum_{q=i}^{k-1} b_q) + b_k + (\sum_{p=k}^{j} a_p + \sum_{q=k+1}^{j} b_q) \\
&= (m[i,k-1]+m[k+1,j]) + \sum_{p=i-1}^{j} a_p + \sum_{q=i}^{j} b_q \\
&= m[i,k-1]+m[k+1,j]+w[i,j]
\end{aligned}
$$

平均比较次数：在所有 k 的情况下 $m[i,j]_k$ 的最小值，

$m[i,j]=\min\{m[i,j]_k \mid i\le k\le j\}$

实例

$$m[i,j]=\min_{i\le k\le j}\{m[i,k-1]+m[k+1,j]+w[i,j]\}\quad 1\le i\le j\le n$$

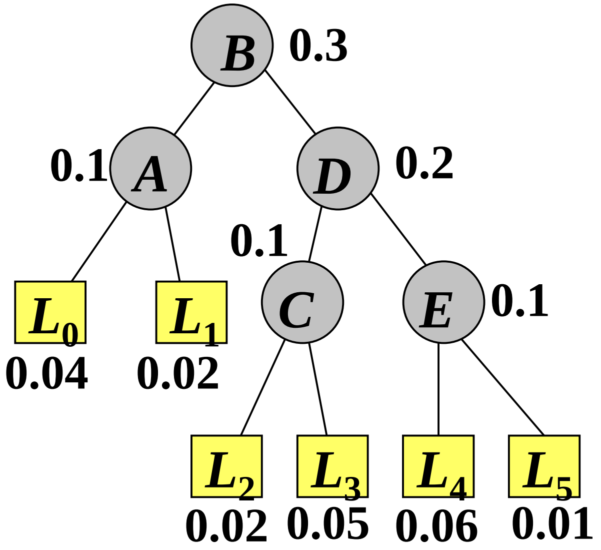
$$m[i,i-1]=0$$

$$m[1,1]=0.16,\quad m[3,5]=0.88$$

$$m[1,5]=1+\min_{k=2,3,4}\{m[1,k-1]+m[k+1,5]\}$$

$$=1+\{m[1,1]+m[3,5]\}$$

$$=1+\{0.16+0.88\}=2.04$$



复杂性估计：

$T(n)=O(n^3)\qquad S(n)=O(n^2)$

小结

(1) 引入参数来界定子问题的边界.

(2) 判断该优化问题是否满足优化原则.

(3)注意子问题的重叠程度.

(4) 给出带边界参数的优化函数定义与优化函数的递推关系 考虑标记函数. 找到递推关系的初值.

(5) 采用自底向上的实现技术，从最小的子问题开始迭代计算，计算中用备忘录保留优化函数和标记函数的值.

(6)动态规划算法的时间复杂度是对所有子问题(备忘录)的计算工作量求和(可能需要追踪解的工作量)

(7)动态规划算法一般使用较多的存储空间，这往往成为限制动态规划算法使用的瓶颈因素.