第6讲 动态规划

汪小林

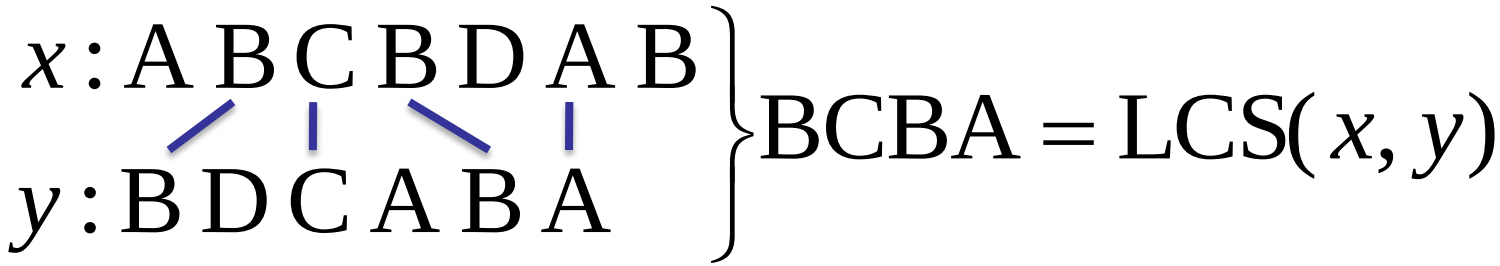
北京大学 信息科学技术学院

主要内容

- 最大公共子序列
  - 转化问题
  - 最优子结构
  - 重叠子问题
  - 还原解
- 矩阵链乘法
- 最优二叉搜索树
- 课堂练习

问题1：最长公共子序列（LCS）

- 给定两个序列$x[1..m]$和$y[1..n]$，找出一个同时出现在两个序列中的最长的子序列（可能不唯一）



LCS的蛮力算法

- 检查$x[1..m]$的每个子序列是否也是$y[1..n]$的子序列，记录下最长的子序列
- 运行时间分析
  - 检查x每个子序列是否为y的子序列 = $O(n)$
  - x的子序列共有2^m个（m个元素都有出现在子序列中或不出现在子序列中2中选择）
  - 最坏情形下运行时间 = $O(n2^m)$=指数时间

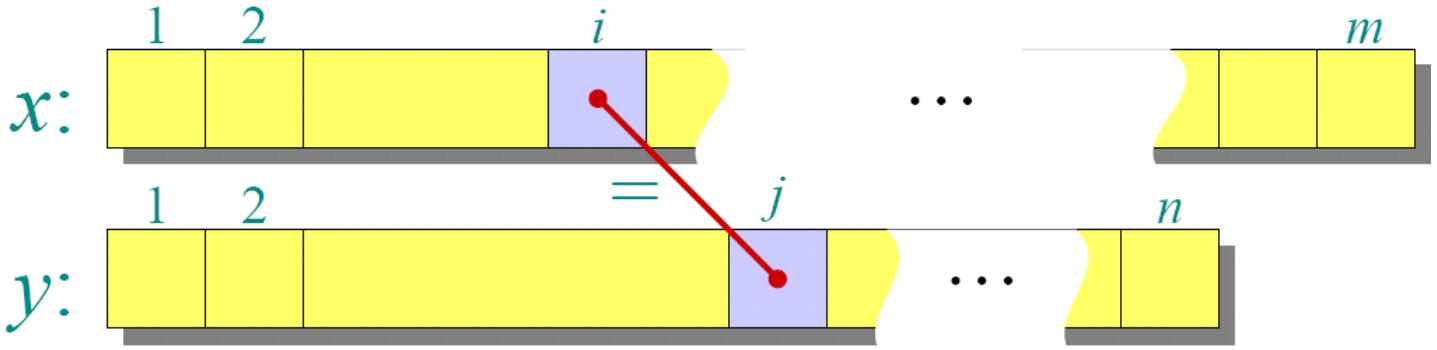
寻找一个更好的算法

- 简化问题
  - 先求解最长公共子序列的长度
  - 再把算法扩展为可计算出最长公共子序列
- 我们记$|s|$表示序列s的长度
- 策略：考虑x和y的前缀
  - 定义 $c[i,j]=|LCS(x[1..i],y[1..j])|$
  - 于是有 $c[m,n]=|LCS(x,y)|$
- 递推方程：

$$c[i,j]=\begin{cases} c[i-1,j-1]+1 & \text{if } x[i]=y[j] \\ \max\{c[i-1,j],c[i,j-1]\} & \text{else} \end{cases}$$

证明递推方程

- 证明：以$x[i]=y[j]$为例（其他情形的证明类似）



- 令$z[1..k]=LCS(x[1..i],y[1..j])$，其中$c[i,j]=k$。此时有$z[k]=x[i]=y[j]$，否则$z$可以扩展为更长的公共子序列。这样，$z[1..k-1]$就是$x[1..i-1]$和$y[1..j-1]$的公共子序列。

证明递推方程（续）

- 证明

  $z[1..k-1]=LCS(x[1..i-1], y[1..j-1])$

- 反正法
  - 假设w是$x[1..i-1]$和$y[1..j-1]$的一个更长的公共子序列，于是有$|w|>k-1$
  - 把w和$z[k]$组合在一起（$w||z[k]$）也是$x[1..i]$和$y[1..j]$的公共子序列，并且$|w||z[k]|>k$
  - 矛盾，故命题成立
- 于是

  $c[i,j]=k=(k-1)+1=c[i-1,j-1]+1$

动态规划的特证1

最优子结构
问题的最优解蕴含着
子问题的最优解

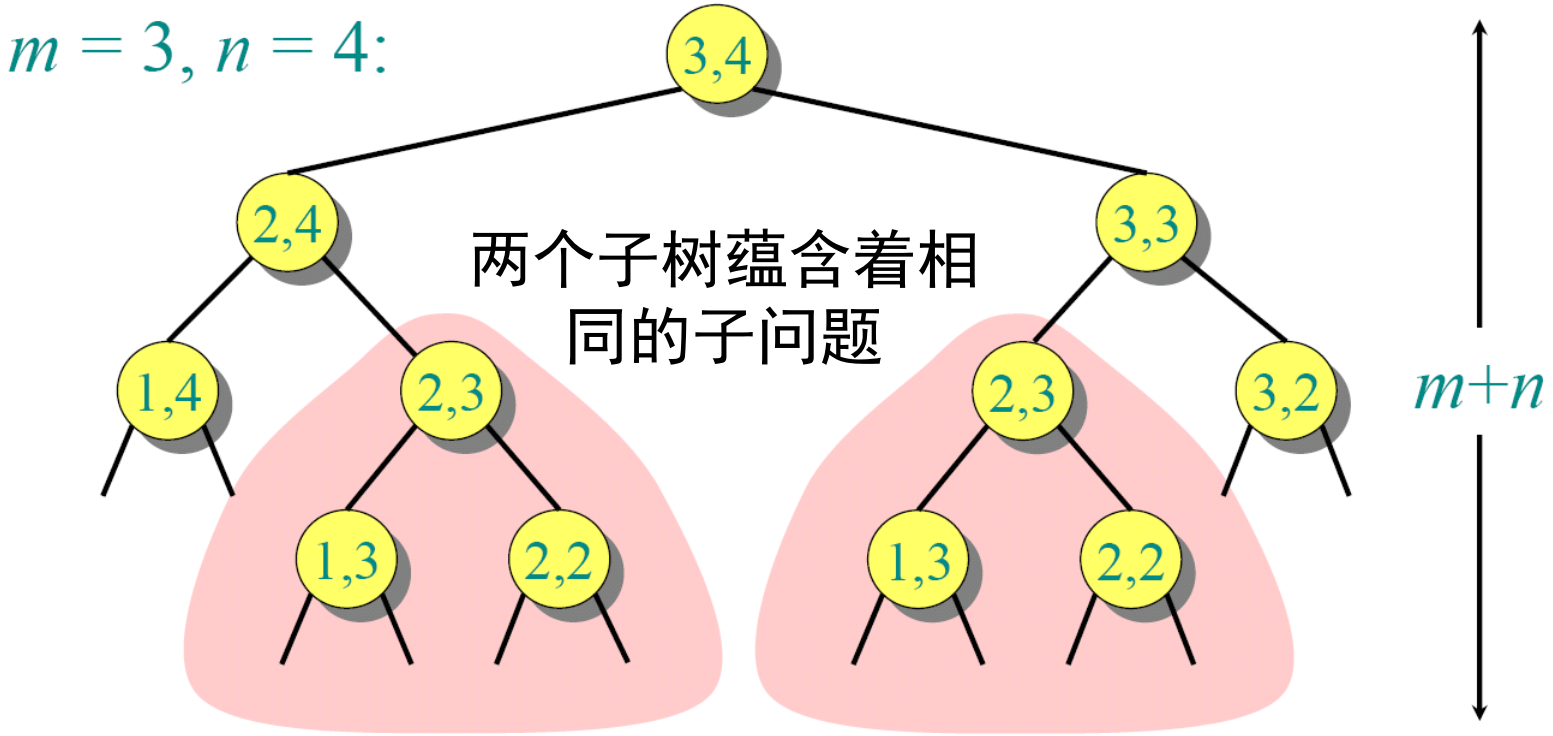
如果$z=LCS(x,y)$，则z的任意一个前缀都是x的某个前缀和y的某个前缀的最大公共子序列

LCS的递归算法

1. **LCS(*x*, *y*, *i*, *j*)**
2. **if *x*[*i*] = *y*[*j*]**
3. **then *c*[*i*, *j*] ← LCS(*x*, *y*, *i*–1, *j*–1) + 1**
4. **else *c*[*i*, *j*] ← max{LCS(*x*, *y*, *i*–1, *j*),LCS(*x*, *y*, *i*, *j*–1)}

- 最坏情形发生在当$x[i] \neq y[j]$时，此时需要计算两个子问题的结果（每个子问题仅比原问题的一个参数减小了1），再求最大值

递归树



高度为$m+n$的树会推导出指数多个子问题，但很多子问题都是曾经计算过了的。

动态规划的特征2

重叠子问题
递归中求解的为数不多的
子问题被多次重复的计算

关于x和y的最大公共子序列长度问题
的子问题仅有mn个

备忘录式的递归算法

- 备忘录：结算完一个子问题的解后，把这个解存放在一个备忘录表中，再次求解同样的子问题时，直接查表返回结果。

1. **LCS(*x*, *y*, *i*, *j*)**
2. **if *c*[*i*,*j*] = NIL**
3. **then if *x*[*i*] = *y*[*j*]**
4. **then *c*[*i*, *j*] ← LCS(*x*, *y*, *i*–1, *j*–1) + 1**
5. **else *c*[*i*, *j*] ← max{LCS(*x*, *y*, *i*–1, *j*),LCS(*x*, *y*, *i*, *j*–1)}**
6. **else return *c*[*i*, *j*]**

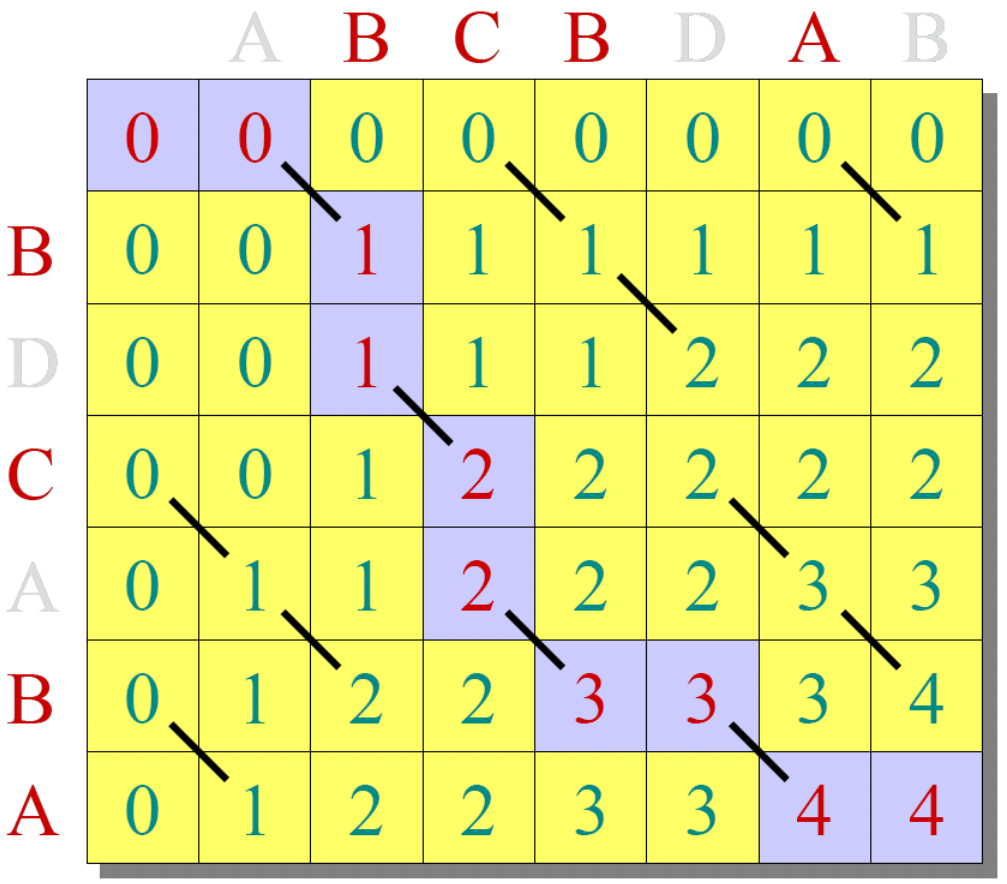
- 算法的运行时间$\Theta(mn)$，算法的空间开销$\Theta(mn)$

动态规划算法伪码

```
LCS-LENGTH(X, Y)
1 m ← length[X]
2 n ← length[Y]
3 for i ← 1 to m
4     do c[i, 0] ← 0
5 for j ← 0 to n
6     do c[0, j] ← 0
7 for i ← 1 to m
8     do for j ← 1 to n
9            do if x_i = y_j
10                 then c[i, j] ← c[i - 1, j - 1] + 1
11                 else if c[i - 1, j] ≥ c[i, j - 1]
12                        then c[i, j] ← c[i - 1, j]
13                        else c[i, j] ← c[i, j - 1]
14 return c[m, n]
```

动态规划算法

- 自底向上递推求解
- 运行时间为$\Theta(mn)$
- 反向追踪还原LCS
- 空间开销为$\Theta(mn)$
- 思考问题？
  空间开销为
  $\Theta(\min(m,n))$
  的算法？
- 如何反向追踪？



		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4

练习与思考

15.4-4 说明如何仅用表中$2\min(m,n)$项以及$O(1)$的额外空间来计算一个LCS的长度；进一步说明，如何用$\min(m,n)$项以及$O(1)$的额外空间做到这一点。

问题2：矩阵乘法

- 设$A_1, A_2, \ldots, A_n$为矩阵序列，其中A_i为$P_{i-1} \times P_i$阶矩阵，$i=1, 2, \ldots, n$。确定乘法顺序使元素相乘的总次数最少。
- 输入：向量$P = <P_0, P_1, \ldots, P_n>$
- 实例：$P = <10, 100, 5, 50>$

  $A_1: 10 \times 100, A_2: 100 \times 5, A_3: 5 \times 50$
- 乘法次序

  $(A_1A_2)A_3: 10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$

  $A_1(A_2A_3): 10 \times 100 \times 50 + 100 \times 5 \times 50 = 75000$
- 一般算法: 加括号的方法有 $\frac{1}{n+1}\binom{2n}{n}$种(Catalan数)

递推方程

- 输入为$P=<P_0, P_1, \dots, P_n>$，$A_{i..j}$ 表示乘积$A_iA_{i+1}..A_j$的结果，其最后一次相乘是$A_{i..j}= A_{i..k}\ A_{k+1..j}$
- $m[i,j]$表示得到$A_{i..j}$的最少的相乘次数，则
- 递推方程

$$m[i,j]=\begin{cases}0 & i=j\\ \min\{m[i,k]+m[k+1,j]+P_{i-1}P_kP_j\} & i<j\end{cases}$$

- 为了确定加括号的次序，设计表$s[i,j]$，记录求得最优时最后一次运算的位置

算法1：递归算法

RecurMatrixChain(*p*,*i*,*j*)

1. $m[i,j] \leftarrow \infty$

2. $s[i,j] \leftarrow i$

3. for $k \leftarrow i$ **to** $j-1$ **do**

4. $q \leftarrow$ **RecurMatrixChain(*p*,*i*,*k*)**
+RecurMatrixChain(*p*,*k*+1,*j*)+ $p_{i-1}p_kp_j$

5. if $q < m[i,j]$

6. then $m[i,j] \leftarrow q$

7. $s[i,j] \leftarrow k$

8. return $m[i,j]$

递归算法运行时间

- 运行时间满足递推关系

$$T(n) \geq \begin{cases} \Theta(1) & n = 1 \\ \sum_{k=1}^{n-1} [T(k) + T(n-k) + \Theta(1)] & n > 1 \end{cases}$$

$$T(n) \geq \Theta(n) + \sum_{k=1}^{n-1} T(k) + \sum_{k=1}^{n-1} T(n-k)$$

$$= \Theta(n) + 2\sum_{k=1}^{n-1} T(k), n > 1$$

递归算法运行时间

- 猜测$T(n)=\Omega(2^n)$
- 只需用数学归纳法证明$T(n)\geq 2^{n-1}$
- 当$n=1$时，显然能成立
- 假设对于任何小于n的k命题都成立，则

$$T(n)\geq \Theta(n)+2\sum_{k=1}^{n-1}T(k)\geq \Theta(n)+2\sum_{k=1}^{n-1}2^{k-1}$$

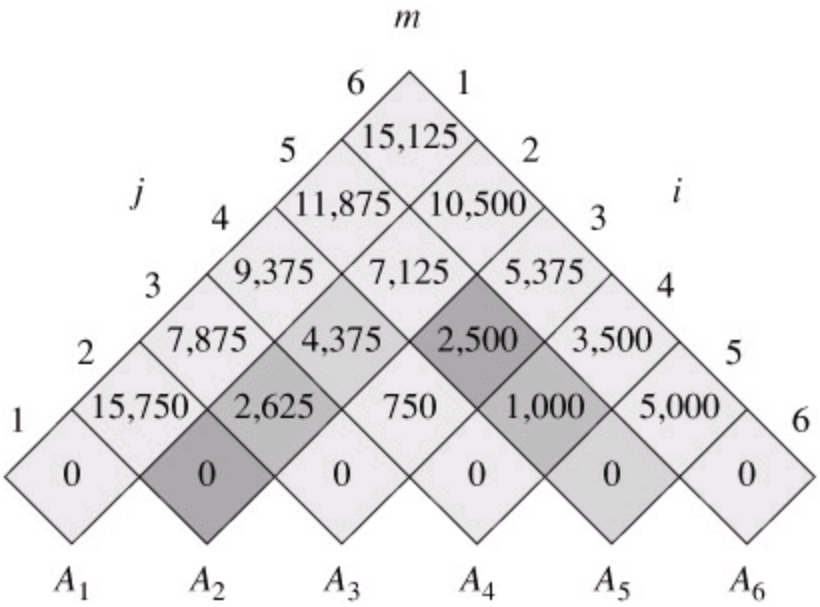
$$=\Theta(n)+2\left(2^{n-1}-1\right)\geq 2^{n-1}$$

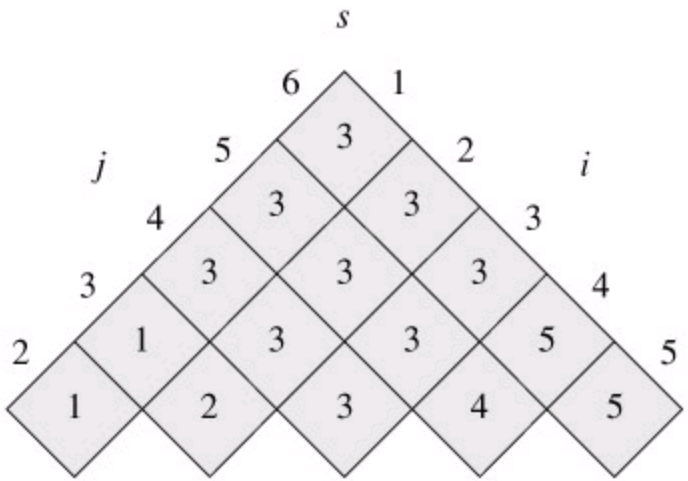
算法2：非递归算法

$T(n)=O(n^3)$

```
MATRIX-CHAIN-ORDER(p)
1 n ← length[p] - 1
2 for i ← 1 to n
3     do m[i, i] ← 0
4 for l ← 2 to n        ▷l 是子矩阵链长度.
5     do for i ← 1 to n - l + 1
6            do j ← i + l - 1
7               m[i, j] ← ∞
8               for k ← i to j - 1
9                   do q ← m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
10                     if q < m[i, j]
11                        then m[i, j] ← q
12                             s[i, j] ← k
13 return m and s
```

矩阵链乘法的动态规划例子





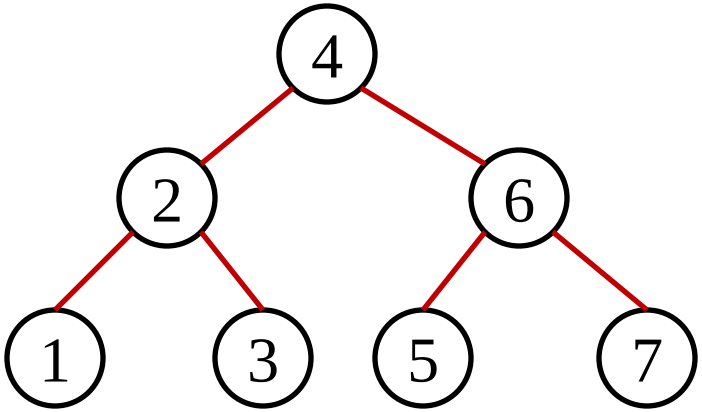
A_1 30×35
A_2 35×15
A_3 15×5
A_4 5×10
A_5 10×20
A_6 20×25

最优加括号的情形

$((A_1 (A_2 A_3)) ((A_4 A_5)A_6))$

问题3：最优二叉搜索树

- 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



- S={1, 2, 3, 4, 5, 6, 7}, 等概分布下

存取概率不等的情况

- 空隙：

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

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

- 给定序列$S = <x_1, x_2, \ldots, x_n>$,

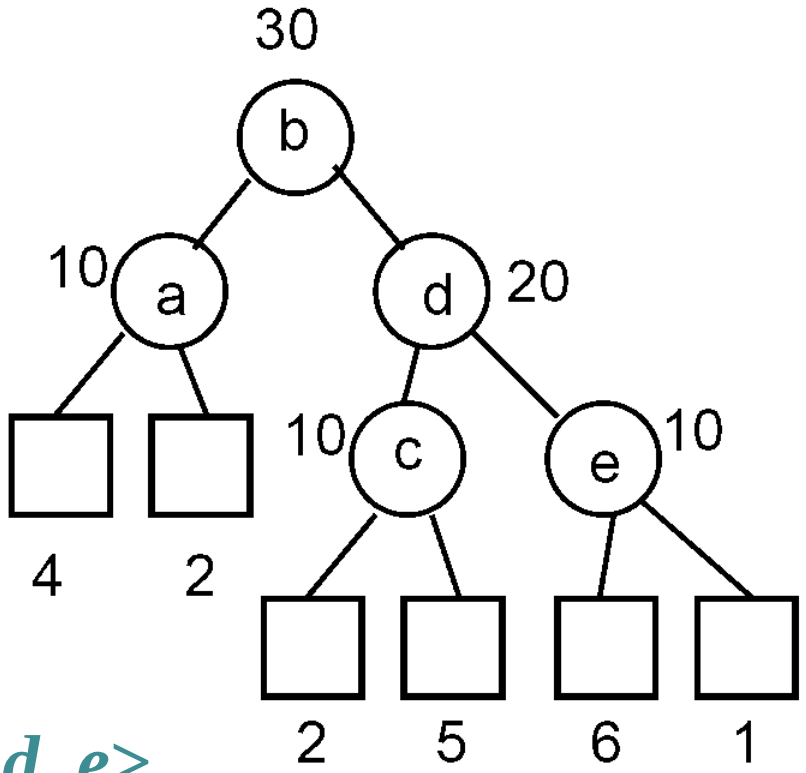
  x在x_i 的概率为b_i ,

  x在(x_i , x_{i+1}) 的概率为a_i ,

- 得到存取概率分布如下：

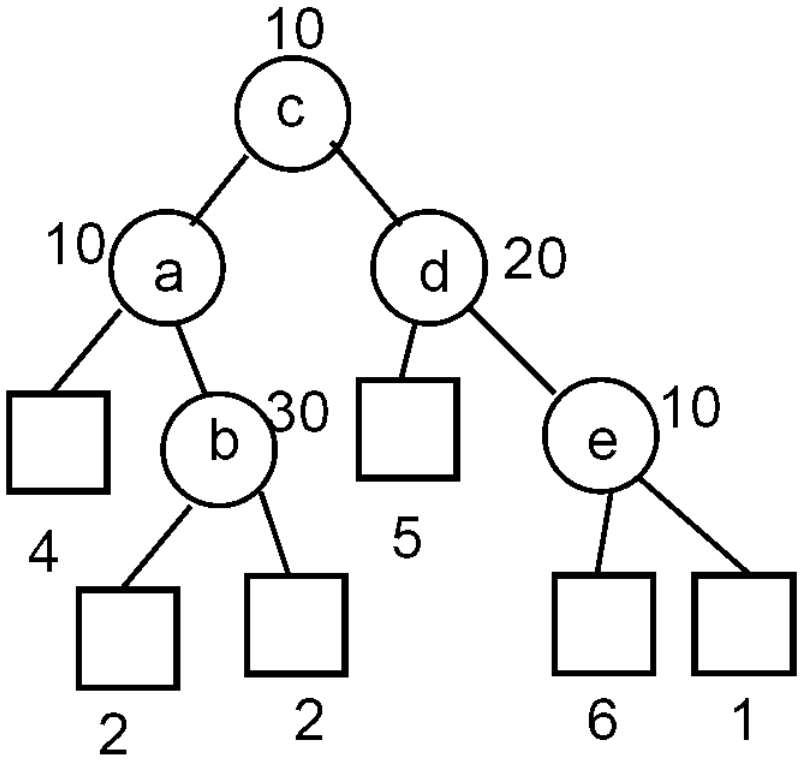
  $C=(a_0, b_1, a_1, b_2, a_2, \cdots, b_n, a_n)$

实例1



- $S = <a, b, c, d, e>$
- C = (4, 10, 2, 30, 2, 10, 5, 20, 6, 10, 1) /100
- p = [0.3×1+(0.1+ 0.2)×2 +(0.1+ 0.1)×3]
  + [(0.04 + 0.02)×2+ (0.02+0.05+0.06+0.01)×3]
  = 2.04

实例2



- $S = <a, b, c, d, e>$
- C = (4, 10, 2, 30, 2, 10, 5, 20, 6, 10, 1) /100
- p = [0.1×1+ (0.1+0.2)×2+(0.1+0.3)×3]
  + [(0.04 + 0.05)×2+ (0.02+0.02+0.06+0.01)×3]
  = 2.41

最优二叉搜索树

- 设x_i 的结点深度为c_i，(x_j, x_{j+1}) 空隙结点（用叶结点表示）的深度为d_j，则平均比较次数为：

$$p = \sum_{i=1}^{n} b_i (1 + c_i) + \sum_{j=0}^{n} a_j d_j$$

- 问题：给定集合S和存取概率分布C，求一棵平均查找次数（平均路长）p 最小的二叉搜索树，即最优二叉搜索树

子问题

设T_{ij}为$\{x_i, x_{i+1}, \cdots, x_j\}$存取概率(条件概率)分布为

$$\left(\overline{a_{i-1}}, b_i, \overline{a_i}, \cdots, b_j, \overline{a_j}\right)$$

其中

$$\overline{b_k} = b_k / w_{ij}, i \le k \le j$$

$$\overline{a_h} = a_h / w_{ij}, i-1 \le h \le j$$

$$w_{ij} = a_{i-1} + b_i + a_i + \cdots + b_j + a_j$$

的最优二叉搜索树，其平均路长为p_{ij}

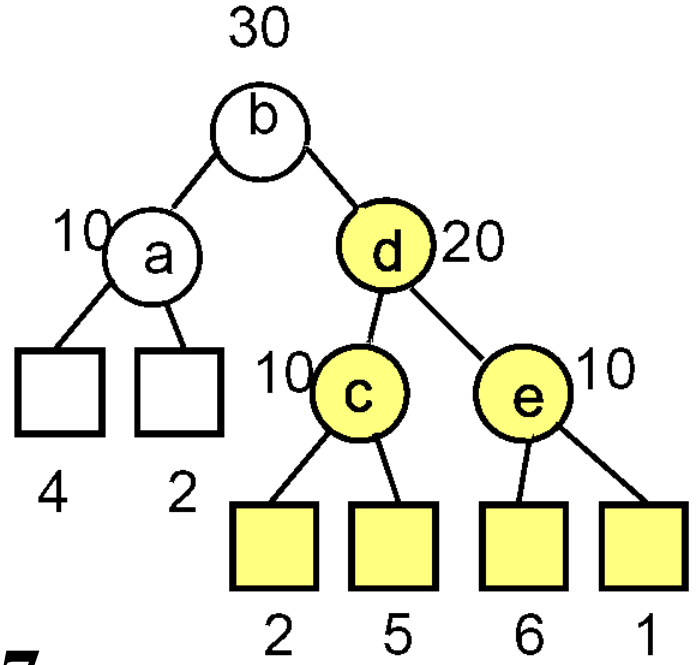
$$m[i, j] = w_{ij} p_{ij}$$

递推方程

$$T_{35}=\{x_3, x_4, x_5\}=\{c,d,e\}$$

$$C=\left(\overline{a_2},\overline{b_3},\overline{a_3},\overline{b_4},\overline{a_4},\overline{b_5},\overline{a_5}\right)$$

$$=\left(\frac{2}{54},\frac{10}{54},\frac{5}{54},\frac{20}{54},\frac{6}{54},\frac{10}{54},\frac{1}{54}\right)$$



$$P(3,5)=1+\left\{1\cdot\frac{17}{54}+1\cdot\frac{17}{54}\right\}=1+\frac{17}{27}\approx 1.63, k=4$$

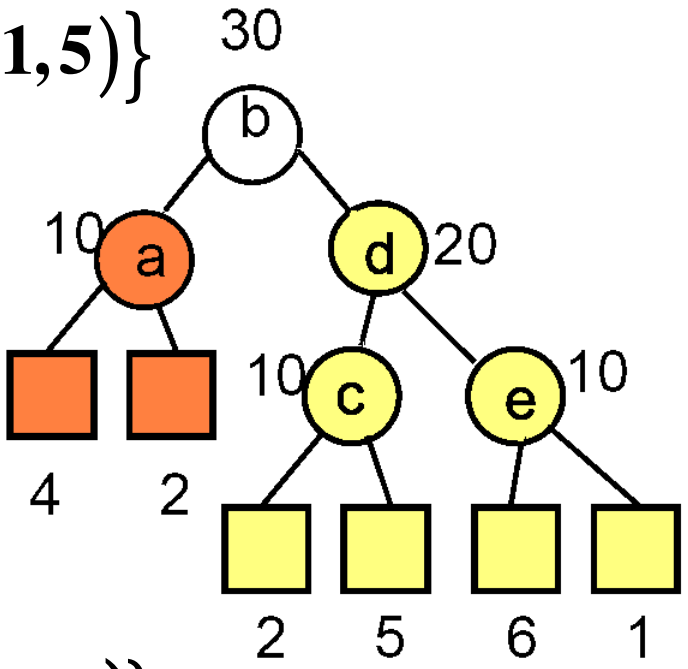
$$P(3,5)=\frac{20}{54}\cdot 1+\frac{34}{54}\cdot 2\approx 0.37+1.26=1.63$$

递推方程

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

$$=1+\{m(1,1)+m(3,5)\}\quad k=2$$

$$\approx 1+\left\{1\cdot\frac{16}{100}+1.63\cdot\frac{54}{100}\right\}=2.04$$



$$m(i,\ j)=1+\min_{i<k<j}\{m(i,k-1)+m(k+1,j)\}\quad i<j$$

$$m(i,\ i)=w_{ii}\quad 1\le i\le n$$

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

练习与思考

15.4-5 如何使用$O(n^2)$时间解决最优二叉搜索树问题

提示：$m(i, j)$与$m(i, j-1)$, $m(i+1, j)$之间的联系

动态规划方法总结

动态规划算法的设计步骤

- 将问题表示成多步判断
- 确定是否满足优化原则——必要条件（最优子结构，子问题独立）
- 确定子问题的重叠性——估计算法效率
- 列出关于优化函数的递推方程(或不等式)和边界条件
- 自底向上计算子问题的优化函数值----非递归的算法
- 备忘录方法记录中间结果
- 标记函数追踪问题的解

动态规划算法的问题

- 时间复杂性改进依赖于子问题的重叠程度
- 空间复杂性较高

总结

- 最优子结构
  - 子问题的独立性
- 重叠子问题
- 递推的步骤

练习与思考

15.4-5 请给出一个$O(n^2)$时间的算法，使之能找到一个n个数字序列中的最大单调递增子序列。