



哈爾濱工業大學

海量数据计算研究中心

Massive Data Computing Lab @ HIT

# 算法设计与分析

## 第四讲 动态规划

哈尔滨工业大学

王宏志

[wangzh@hit.edu.cn](mailto:wangzh@hit.edu.cn)

<http://homepage.hit.edu.cn/pages/wang/>

# 本讲内容

4. 1 动态规划的原理

4. 2 矩阵乘法问题

4. 3 最长公共子序列问题

4. 4 0-1背包问题

4. 5 最优二分搜索树

# Why?

- 分治技术的问题
  - 子问题是相互独立的
  - 如果子问题不是相互独立的，分治方法将重复计算公共子问题，效率很低
- 优化问题
  - 给定一组约束条件和一个代价函数，在解空间中搜索具有最小或最大代价的优化解
  - 很多优化问题可分为多个子问题，子问题相互关联，子问题的解被重复使用

# What?

- 动态规划的特点
  - 把原始问题划分成一系列子问题
  - 求解每个子问题仅一次，并将其结果保存在一个表中，以后用到时直接存取，不重复计算，节省计算时间
  - 自底向上地计算
- 适用范围
  - 一类优化问题：可分为多个相关子问题，子问题的解被重复使用

# How?

- 使用动态规划的条件
  - 优化子结构
    - 当一个问题的优化解包含了子问题的优化解时，我们说这个问题具有优化子结构。
    - 缩小子问题集合，只需那些优化问题中包含的子问题，降低实现复杂性
    - 优化子结构使得我们能自下而上地完成求解过程
  - 重叠子问题
    - 在问题的求解过程中，很多子问题的解将被多次使用

- 动态规划算法的设计步骤
  - 分析优化解的结构
  - 递归地定义最优解的代价
  - 自底向上地计算优化解的代价保存之，并获取构造最优解的信息
  - 根据构造最优解的信息构造优化解



# 本讲内容

4. 1 动态规划的原理

4. 2 矩阵乘法问题

4. 3 最长公共子序列问题

4. 4 0-1背包问题

4. 5 最优二分搜索树

# 问题的定义

- 输入:  $\langle A_1, A_2, \dots, A_n \rangle$ ,  $A_i$ 是矩阵
- 输出: 计算 $A_1 \times A_2 \times \dots \times A_n$ 的最小代价方法

矩阵乘法的代价/复杂性: 乘法的次数

若 $A$ 是 $p \times q$ 矩阵,  $B$ 是 $q \times r$ 矩阵, 则 $A \times B$ 的代价是 $O(pqr)$

# 动机

- 矩阵链乘法的实现
  - 矩阵乘法满足结合率。
  - 计算一个矩阵链的乘法可有多种方法：

例如,  $(A_1 \times A_2 \times A_3 \times A_4)$

$$= (A_1 \times (A_2 \times (A_3 \times A_4)))$$

$$= ((A_1 \times A_2) \times (A_3 \times A_4))$$

....

$$= ((A_1 \times A_2) \times A_3) \times A_4)$$

- 矩阵链乘法的代价与计算顺序的关系
  - 设  $A_1=10\times 100$  矩阵,  $A_2=100\times 5$  矩阵,  $A_3=5\times 50$  矩阵

$$T((A_1 \times A_2) \times A_3) = 10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$$

$$T(A_1 \times (A_2 \times A_3)) = 100 \times 5 \times 50 + 10 \times 100 \times 50 = 750000$$

结论: 不同计算顺序有不同的代价

- 矩阵链乘法优化问题的解空间
  - 设 $p(n)$ =计算 $n$ 个矩阵乘积的方法数
  - $p(n)$ 的递归方程

如此之大的解空间是无法用枚举方法  
求出最优解的！

$$p(n) = \sum_{k=1}^{n-1} p(k)p(n-k) \quad \text{if } n > 1$$

$$\begin{aligned} p(n) &= C(n-1) = \text{Catalan数} = \frac{1}{n} \binom{2(n-1)}{n-1} \\ &= \Omega(4^n/n^{3/2}) \end{aligned}$$

# 下边开始设计求解矩阵链乘法问题 的动态规划算法

- 分析优化解的结构
- 递归地定义最优解的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优解的信息
- 根据构造最优解的信息构造优化解

# 分析优化解的结构

- 两个记号

- $A_{i:j} = A_i \times A_{i+1} \times \dots \times A_j$

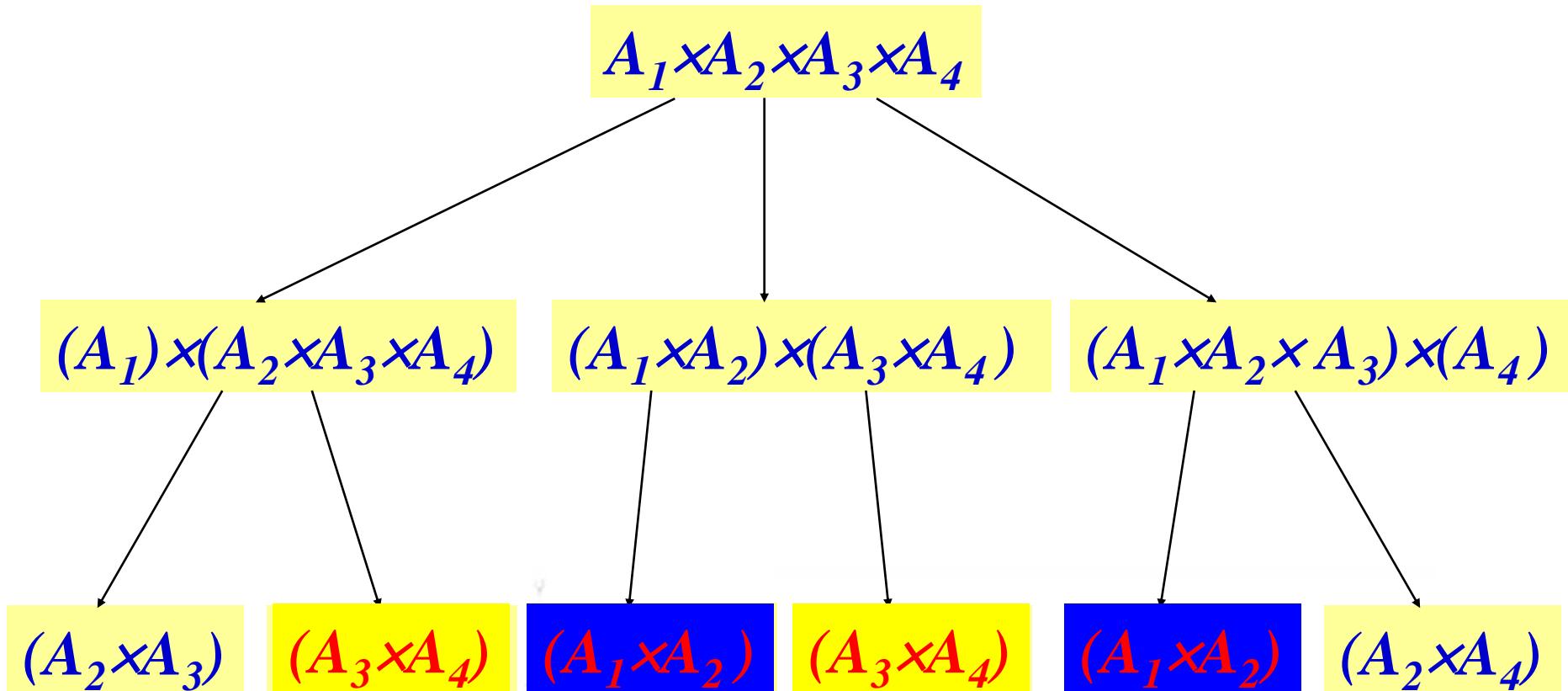
- $\text{cost}(A_{i:j})$  = 计算  $A_{i:j}$  的代价

- 优化解的结构

- 若计算  $A_{i:j}$  的优化顺序在  $k$  处断开矩阵链，  
具有优化子结构：

问题的优化解包括子问题优化解  
化解，对于子问题  $A_{k+1:n}$  的解必须是  
 $A_{k+1:n}$  的优化解

## • 子问题重叠性



具有子问题重叠性

# 递归地定义最优解的代价

- 假设

- $m[i, j]$  = 计算  $A_{i \sim j}$  的最小乘法数

- $m[1, n]$  = 计算  $A_{1 \sim n}$  的最小乘法数

- $A_1 \dots A_k A_{k+1} \dots A_n$  是优化解 ( $k$  实际上是不可预知)

- 代价方程

- $m[i, i] =$  计算  $A_{i \sim i}$  的最小乘法数 = 0

- $m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$

- 其中,  $p_{i-1} p_k p_j$  是计算  $A_{i \sim k} \times A_{k+1 \sim j}$  所需乘法数,  
 $A_{i \sim k}$  和  $A_{k+1 \sim j}$  分别是  $p_{i-1} \times p_k$  和  $p_k \times p_j$  矩阵.

考虑到所有的 $k$ , 优化解的代价方程为

$$m[i, j] = 0 \quad \text{if } i=j$$

$$m[i, j] = \min_{i \leq k < j} \{ m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \}$$

$$\text{if } i < j$$

# 自底向上计算优化解的代价

$$m[i, j] = \min_{i \leq k < j} \{ m[i, k] + m[k+1, j] + p_0 p_k p_5 \}$$

$m[1,1]$   $m[1,2]$   $m[1,3]$   $m[1,4]$   $m[1,5]$

$m[2,2]$   $m[2,3]$   $m[2,4]$   $m[2,5]$

$m[3,3]$   $m[3,4]$   $m[3,5]$

$m[4,4]$   $m[4,5]$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] \\ m[2,3] + m[4,4] \end{cases} \quad m[5,5]$$

$$m[i, j] = \min_{i \leq k < j} \{ m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \}$$

~~$m[1,1]$   $m[1,2]$   $m[1,3]$   $m[1,4]$   $m[1,5]$~~

~~$m[2,2]$   $m[2,3]$   $m[2,4]$   $m[2,5]$~~

~~$m[3,3]$   $m[3,4]$   $m[3,5]$~~

~~$m[4,4]$   $m[4,5]$~~

~~$m[5,5]$~~



# Matrix-Chain-Order( $p$ )

$n=\text{length}(p)-1;$

**FOR**  $i=1$  TO  $n$  DO

$m[i, i]=0;$

**FOR**  $l=2$  TO  $n$  DO /\* 计算第  $l$  对角线 \*/

**FOR**  $i=1$  TO  $n-l+1$  DO

$j=i+l-1;$

$m[i, j]=\infty;$

**FOR**  $k \leftarrow i$  To  $j-1$  DO /\* 计算  $m[i, j]$  \*/

$q=m[i, k]+m[k+1, j]+p_{i-1}p_kp_j$

**IF**  $q < m[i, j]$  **THEN**  $m[i, j]=q;$

**Return**  $m.$

# 获取构造最优解的信息

**Matrix-Chain-Order( $p$ )**

$n = \text{length}(p) - 1;$

**FOR**  $i=1$  TO  $n$  DO

$m[i, i] = 0;$

**FOR**  $l=2$  TO  $n$  DO

**FOR**  $i=1$  TO  $n-l+1$  DO

$j=i+l-1;$

$m[i, j] = \infty;$

**FOR**  $k \leftarrow i$  To  $j-1$  DO

$q = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$

            IF  $q < m[i, j]$  THEN  $m[i, j] = q,$

$s[i, j] = k;$

**Return**  $m$  and  $s.$

$S[i, j]$ 记录 $A_i A_{i+1} \dots A_j$ 的最优划分处在 $A_k$ 与 $A_{k+1}$ 之间

# 构造最优解

**Print-Optimal-Parens( $s, i$ ,**

IF  $j=i$

THEN Print “A” $i$ ;

ELSE Print “(”

**Print-Optimal-Parens( $s, i, s[i, j]$ )**

**Print-Optimal-Parens( $s, s[i, j]+1, j$ )**

Print “)”

$S[i, j]$ 记录  $A_i \dots A_j$  的最优划分处；

$S[i, S[i, j]]$  记录  $A_i \dots A_{s[i, j]}$  的最优划分处；

$S[S[i, j]+1, j]$  记录  $A_{s[i, j]+1} \dots A_j$  的最优划分处。

调用 **Print-Optimal-Parens( $s, 1, n$ )**

即可输出  $A_{1 \sim n}$  的优化计算顺序

# 算法复杂性

- 时间复杂性
  - 计算代价的时间
    - $(l, i, k)$ 三层循环, 每层至多 $n-1$ 步
    - $O(n^3)$
  - 构造最优解的时间:  $O(n)$
  - 总时间复杂性为:  $O(n^3)$
- 空间复杂性
  - 使用数组 $m$ 和 $S$
  - 需要空间 $O(n^2)$

# 本讲内容

- 4. 1 动态规划的原理
- 4. 2 矩阵乘法问题
- 4. 3 最长公共子序列问题
- 4. 4 0-1背包问题
- 4. 5 最优二分搜索树

# 问题的定义

- 子序列
  - $X=(A, B, C, B, D, B)$
  - $Z=(B, C, D, B)$ 是 $X$ 的子序列
  - $W=(B, D, A)$ 不是 $X$ 的子序列
- 公共子序列
  - $Z$ 是序列 $X$ 与 $Y$ 的公共子序列如果 $Z$ 是 $X$ 的子序也是 $Y$ 的子序列。



# 最长公共子序列（LCS）问题

输入：  $X = (x_1, x_2, \dots, x_n)$ ,  $Y = (y_1, y_2, \dots, y_m)$

输出：  $Z = X$  与  $Y$  的最长公共子序列

# 最长公共子序列结构分析

- 第*i*前缀

- 设 $X=(x_1, x_2, \dots, x_n)$ 是一个序列， $X$ 的第*i*前缀 $X_i$ 是一个序列，定义为 $X_i=(x_1, \dots, x_i)$

例.  $X=(A, B, D, C, A)$ ,  $X_1=(A)$ ,  $X_2=(A, B)$ ,  $X_3=(A, B, D)$



- 优化子结构

**定理1（优化子结构）** 设 $X=(x_1, \dots, x_m)$ 、  
 $Y=(y_1, \dots, y_n)$  是两个序列， $Z=(z_1, \dots, z_k)$ 是 $X$ 与 $Y$ 的  
 $LCS$ ， 我们有：

- (1) 如果 $x_m=y_n$ , 则 $z_k=x_m=y_n$ ,  $Z_{k-1}$ 是 $X_{m-1}$ 和 $Y_{n-1}$ 的  
 $LCS$ ， 即，  $LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + <x_m=y_n>$ .
- (2) 如果 $x_m \neq y_n$ , 且 $z_k \neq x_m$ , 则 $Z$ 是 $X_{m-1}$ 和 $Y$ 的  
 $LCS$ ， 即  $LCS_{XY} = LCS_{X_{m-1}Y}$
- (3) 如果 $x_m \neq y_n$ , 且 $z_k \neq y_n$ , 则 $Z$ 是 $X$ 与 $Y_{n-1}$ 的 $LCS$ ，  
即  $LCS_{XY} = LCS_{XY_{n-1}}$

## 证明：

(1).  $X = \langle x_1, \dots, x_{m-1}, \textcolor{red}{x_m} \rangle$ ,  $Y = \langle y_1, \dots, y_{n-1}, \textcolor{red}{x_m} \rangle$ , 则

$$LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + \langle x_m = y_n \rangle.$$

设  $z_k \neq x_m$ , 则可加  $x_m = y_n$  到 Z, 得到一个长为  $k+1$  的 X 与 Y 的公共序列, 与 Z 是 X 和 Y 的 LCS 矛盾。于是  $z_k = x_m = y_n$ 。

现在证明  $Z_{k-1}$  是  $X_{m-1}$  与  $Y_{n-1}$  的 LCS。显然  $Z_{k-1}$  是  $X_{m-1}$  与  $Y_{n-1}$  的公共序列。我们需要证明  $Z_{k-1}$  是 LCS。

设不然, 则存在  $X_{m-1}$  与  $Y_{n-1}$  的公共子序列 W, W 的长大于  $k-1$ 。增加  $x_m = y_n$  到 W, 我们得到一个长大于 k 的 X 与 Y 的公共序列, 与 Z 是 LCS 矛盾。于是,  $Z_{k-1}$  是  $X_{m-1}$  与  $Y_{n-1}$  的 LCS。

(2)  $X = \langle x_1, \dots, x_{m-1}, x_m \rangle$ ,  $Y = \langle y_1, \dots, y_{n-1}, y_n \rangle$ ,  
 $x_m \neq y_n$ ,  $z_k \neq x_m$ , 则  $LCS_{XY} = LCS_{X_{m-1}Y}$

由于  $z_k \neq x_m$ ,  $Z$  是  $X_{m-1}$  与  $Y$  的公共子序列。我们来证  $Z$  是  $X_{m-1}$  与  $Y$  的  $LCS$ 。设  $X_{m-1}$  与  $Y$  有一个公共子序列  $W$ ,  $W$  的长大于  $k$ , 则  $W$  也是  $X$  与  $Y$  的公共子序列, 与  $Z$  是  $LCS$  矛盾。

(3) 同(2)可证。

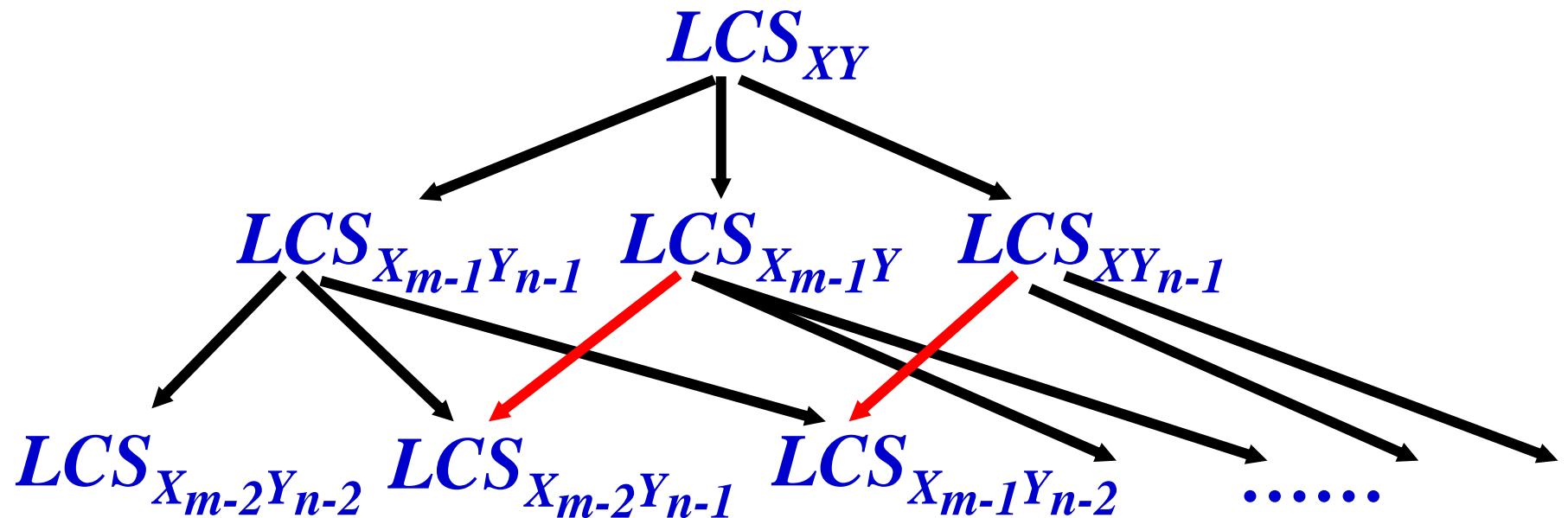
$X$ 和 $Y$ 的LCS的优化解结构为

$$LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + \langle x_m = y_n \rangle \quad \text{if } x_m = y_n$$

$$LCS_{XY} = LCS_{X_{m-1}Y} \quad \text{if } x_m \neq y_n, z_k \neq x_m$$

$$LCS_{XY} = LCS_{XY_{n-1}} \quad \text{if } x_m \neq y_n, z_k \neq y_n$$

- 子问题重叠性



**LCS问题具有子问题重叠性**

# 建立LCS长度的递归方程

- $C[i, j]$  =  $X_i$ 与 $Y_j$  的LCS的长度
- LCS长度的递归方程

$$C[i, j] = 0$$

if  $i=0$  或  $j=0$

$$C[i, j] = C[i-1, j-1] + 1$$

if  $i, j>0$  且  $x_i = y_j$

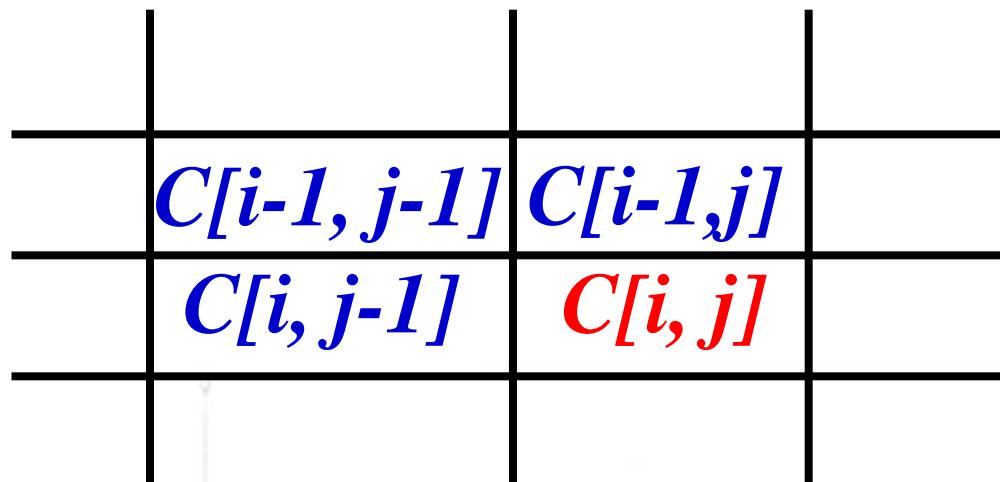
$$C[i, j] = \text{Max}(C[i, j-1], C[i-1, j])$$

if  $i, j>0$  且  $x_i \neq y_j$



# 自底向上计算LCS的长度

- 基本思想



- 计算过程

$C[0,0]$	$C[0,1]$	$C[0,2]$	$C[0,3]$	$C[0,4]$
$C[1,0]$	$C[1,1]$	$C[1,2]$	$C[1,3]$	$C[1,4]$
$C[2,0]$	$C[2,1]$	$C[2,2]$	$C[2,3]$	$C[2,4]$
$C[3,0]$	$C[3,1]$	$C[3,2]$	$C[3,3]$	$C[3,4]$



- 计算LCS长度的算法
  - 数据结构

**$C[0:m,0:n]$ :**  $C[i,j]$ 是 $X_i$ 与 $Y_j$ 的LCS的长度

**$B[1:m,1:n]$ :**  $B[i,j]$ 是指针，指向计算 $C[i,j]$ 时所选择的子问题的优化解所对应的C表的表项

## LCS-length( $X, Y$ )

$m \leftarrow \text{length}(X); n \leftarrow \text{length}(Y);$

**For**  $i \leftarrow 1$  **To**  $m$  **Do**  $C[i,0] \leftarrow 0;$

**For**  $j \leftarrow 1$  **To**  $n$  **Do**  $C[0,j] \leftarrow 0;$

**For**  $i \leftarrow 1$  **To**  $m$  **Do**

**For**  $j \leftarrow 1$  **To**  $n$  **Do**

**If**  $x_i = y_j$

**Then**  $C[i,j] \leftarrow C[i-1,j-1] + 1;$   $B[i,j] \leftarrow “\nwarrow”;$

**Else If**  $C[i-1,j] \geq C[i,j-1]$

**Then**  $C[i,j] \leftarrow C[i-1,j];$   $B[i,j] \leftarrow “\uparrow”;$

**Else**  $C[i,j] \leftarrow C[i,j-1];$   $B[i,j] \leftarrow “\leftarrow”;$

**Return**  $C$  and  $B.$

# 构造优化解

- 基本思想
  - 从 $B[m, n]$ 开始按指针搜索
  - 若 $B[i, j]=“↖”$ ， 则 $x_i=y_j$ 是LCS的一个元素
  - 如此找到的 “LCS”是X与Y的LCS

**Print-LCS( $B, X, i, j$ )**

IF  $i=0$  or  $j=0$  THEN Return;

IF  $B[i, j]=“↖”$

THEN Print-LCS( $B, X, i-1, j-1$ );

    Print  $x_i$ ;

ELSE If  $B[i, j]=“↑”$

        THEN Print-LCS( $B, X, i-1, j$ );

        ELSE Print-LCS( $B, X, i, j-1$ ).

**Print-LCS(B, X, length(X), length(Y))**

可打印出 $X$ 与 $Y$ 的*LCS*。

# 算法复杂性

- 时间复杂性
  - 计算代价的时间
    - $(i, j)$ 两层循环,  $i$ 循环 $m$ 步,  $j$ 循环 $n$ 步
    - $O(mn)$
  - 构造最优解的时间:  $O(m+n)$
  - 总时间复杂性为:  $O(mn)$
- 空降复杂性
  - 使用数组 $C$ 和 $B$
  - 需要空间 $O(mn)$

# 本讲内容

4. 1 动态规划的原理

4. 2 矩阵乘法问题

4. 3 最长公共子序列问题

4. 4 0-1背包问题

4. 5 最优二分搜索树

## 问题的定义

给定 $n$ 种物品和一个背包，物品 $i$ 的重量是 $w_i$ ，价值 $v_i$ ，背包容量为 $C$ ，问如何选择装入背包的物品，使装入背包中的物品的总价值最大？

对于每种物品只能选择完全装入或不装入，一个物品至多装入一次。

- 输入:  $C > 0, w_i > 0, v_i > 0, 1 \leq i \leq n$
- 输出:  $(x_1, x_2, \dots, x_n)$ ,  $x_i \in \{0, 1\}$ , 满足

$$\sum_{1 \leq i \leq n} w_i x_i \leq C, \quad \sum_{1 \leq i \leq n} v_i x_i \text{ 最大}$$

## 等价的整数规划问题

$$\max \sum_{1 \leq i \leq n} v_i x_i$$

$$\sum_{1 \leq i \leq n} w_i x_i \leq C$$

$$x_i \in \{0, 1\}, \quad 1 \leq i \leq n$$

$\{w_1, \dots, w_k\}$

if  $2^{k+1} > C$ ,  $w_1 = w_2 + w_3 + \dots$   
 $w_0 = w_2$ .

if  $2^{k+1} \leq C$  不一定

$$x_1, x_2, \dots, x_n$$

$$\max \sum_{1 \leq i \leq n} v_i x_i$$

$$\sum_{1 \leq i \leq n} w_i x_i \leq C - w_0 x_1$$

$$\text{cost}_{1,c} = \max \begin{cases} \text{cost}_{2,c-w_0+x_1} + v_1 \\ \text{cost}_{2,c} \end{cases}$$

# 优化解结构的分析

**定理 (优化子结构)** 如果  $(y_1, y_2, \dots, y_n)$  是 0-1 背包问题的优化解，则  $(y_2, \dots, y_n)$  是如下子问题的优化解：

$$\begin{aligned} & \max \sum_{2 \leq i \leq n} v_i x_i \\ & \sum_{2 \leq i \leq n} w_i x_i \leq C - w_1 y_1 \\ & x_i \in \{0, 1\}, \quad 2 \leq i \leq n \end{aligned}$$

**证明：**如果  $(y_2, \dots, y_n)$  不是子问题优化解，则存在  $(z_2, \dots, z_n)$  是子问题更优的解。于是， $(y_1, z_2, \dots, z_n)$  是原问题比  $(y_1, y_2, \dots, y_n)$  更优解，矛盾。

# 建立优化解代价的递归方程

- 设子问题

$$\begin{aligned} & \max \sum_{i \leq k \leq n} v_k x_k \\ & \sum_{i \leq k \leq n} w_k x_k \leq j \\ & x_k \in \{0, 1\}, \quad i \leq k \leq n \end{aligned}$$

的最优解代价为  $m(i, j)$ .

即  $m(i, j)$  是背包容量为  $j$ , 可选物品为  $i, i+1, \dots, n$  时问题最优解的代价.

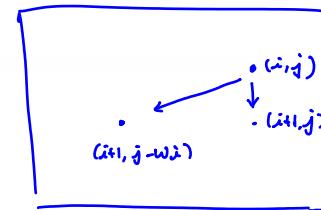
# 递归方程

$$m(i, j) = m(i+1, j) \quad 0 \leq j < w_i$$

$$m(i, j) = \max\{m(i+1, j), m(i+1, j-w_i) + v_i\} \quad j \geq w_i$$

$$m(n, j) = 0 \quad 0 \leq j < w_n$$

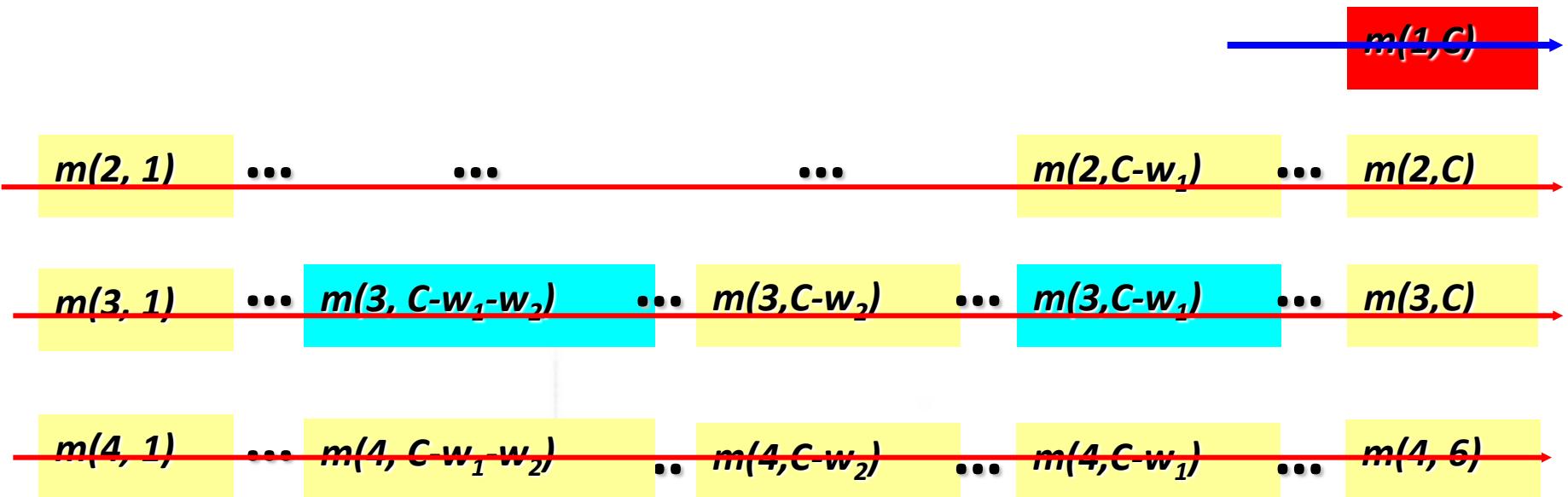
$$m(n, j) = v_n \quad j \geq w_n$$



# 自底向上计算优化解的代价

计算 $m(i, j)$ 需要  
 $m(i+1, j-w_i)$ 和 $m(i+1, j)$

令 $w_i=$ 整数,  $n=4$



# • 算法

$O(\Theta)$

For  $j=0$  To  $\min(w_n-1, C)$  Do

$m[n, j] = 0;$

For  $j=w_n$  To  $C$  Do

$m[n, j] = v_n;$

For  $i=n-1$  To  $2$  Do

For  $\tilde{j}=0$  To  $\min(w_i-1, \tilde{C})$  Do

$m[i, j] = m[i+1, j];$

For  $j=w_i$  To  $\tilde{C}$  Do

$m[i, j] = \max\{m[i+1, j], m[i+1, j-w_i] + v_i\};$

If  $C < w_1$

시간복잡도:  $O(nC)$

공간복잡도:  $O(nC)$

Then  $m[1, C] = m[2, C];$

Else  $m[1, C] = \max\{m[2, C], m[2, C-w_1] + v_1\};$

		C=10													
		v <sub>i</sub>		w <sub>i</sub>											
1	4	6	2	0	0	0	6	6	6	6	6	11	..		
2	6	4	3	0	0	0	5	5	5	5	5	11	..		
3	5	5	3	0	0	0	5	5	5	5	5	11	..		
4	3	7	4	0	0	0	0	0	3	3	3	3	3		
			1	2	3	4	5	6	7	8	9	10			

# 构造优化解

1.  $m(1, C)$ 是最优解代价值，相应解计算如下：

If  $m(1, C)=m(2, C)$

Then  $x_1=0$

Else  $x_1=1;$

2. 如果  $x_1=0$ , 由  $m(2, C)$ 继续构造最优解；

3. 如果  $x_1=1$ , 由  $m(2, C-w_1)$ 继续构造最优解.

$$S = \{k_1, \dots, k_n\}$$

$$S' \subseteq S, S' = \{k'_1, \dots, k'_m\}$$

$$\sum_{k_i \in S'} k'_i = \frac{1}{2} \sum_{k_i \in S} k_i$$

# 本讲内容

4. 1 动态规划的原理

4. 2 矩阵乘法问题

4. 3 最长公共子序列问题

4. 4 0-1背包问题

4. 5 最优二分搜索树

# 问题的定义

- 二叉搜索树  $T$

- 搜索树的期望代价

$$E(T) = \sum_{i=1}^n (DEP_T(k_i) + 1)p_i + \sum_{j=0}^n (DEP_T(d_j) + 1)q_j$$

$$E(T) = \min_{1 \leq i \leq n} \{ E(T_{i+1:n}) + E(T_{i+1:n}) + \sum p_i + \sum q_j \}$$

$a_n$  对应区间  
 $(k_n, +\infty)$

- 附加信息

- 搜索  $k_i$  的概率为  $p_i$

- 搜索  $d_i$  的概率为  $q_i$

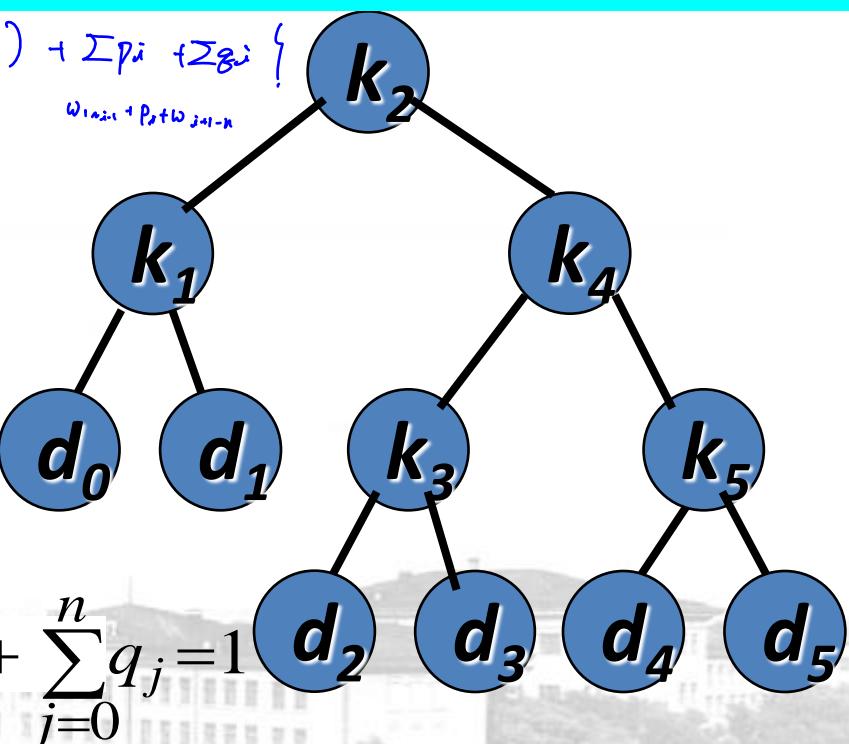
$$\sum_{i=1}^n p_i + \sum_{j=0}^n q_j = 1$$

$k_1 \dots k_n$

$\frac{k_1 \dots k_{i-1}}{T_1} \overset{k_i}{\wedge} \frac{k_{i+1} \dots k_n}{T_2}$

$$E(T_i) = \sum_{l=1}^{i-1} (\text{dep}_{T_i}(k_l) + 1)p_l$$

$$+ \sum_{l=0}^{n-i} (\text{dep}_{T_i}(d_l) + 1)q_l$$



## • 问题的定义

输入：  $k=\{k_1, k_2, \dots, k_n\}$ ,  $k_1 < k_2 < \dots < k_n$ ,  
 $P=\{p_1, p_2, \dots, p_n\}$ ,  $p_i$  为搜索  $k_i$  的概率  $Q=\{q_0, q_1, \dots, q_n\}$ ,  $q_i$  为搜索值  $d_i$  的概率

输出：  $K$  的二叉搜索树  $T$ ,  $E(T)$  最小

# 优化二叉搜索树结构的分析

- 优化子结构

**定理.** 如果优化二叉搜索树 $T$ 具有包含关键字集合 $\{k_i, k_{i+1}, \dots, k_j\}$ 子树 $T'$ , 则 $T'$ 必是关于关键字集合 $\{k_i, k_{i+1}, \dots, k_j\}$ 子问题的优化解.

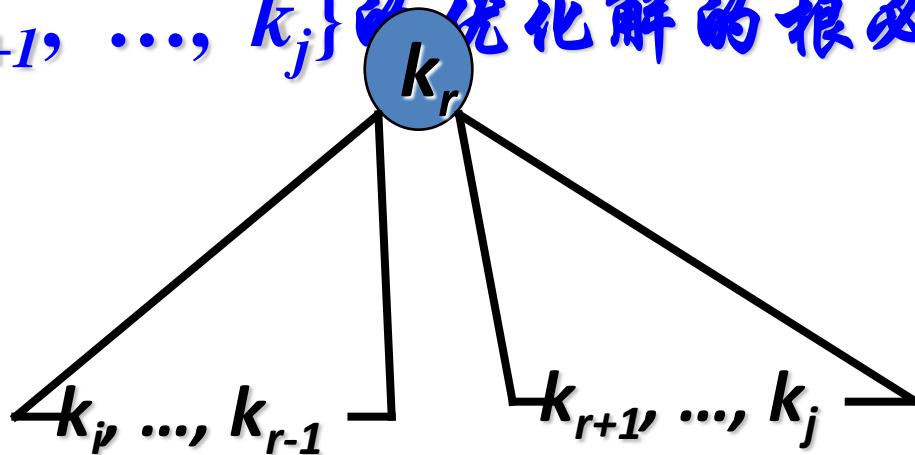
**证明:** 若不然, 必有关关键字集 $\{k_i, k_{i+1}, \dots, k_j\}$ 子树 $T''$ ,  $T''$ 的期望搜索代价低于 $T'$ .

用 $T''$ 替换 $T$ 中的 $T'$ , 可以得到一个期望搜索代价比 $T$ 小的原始问题的二叉搜索树,  
与 $T$ 是最优解矛盾.

- 用优化子结构从子问题优化解构造优化解

$K = \{k_i, k_{i+1}, \dots, k_j\}$  的优化解的根必为  $K$  中某个

$k_r$

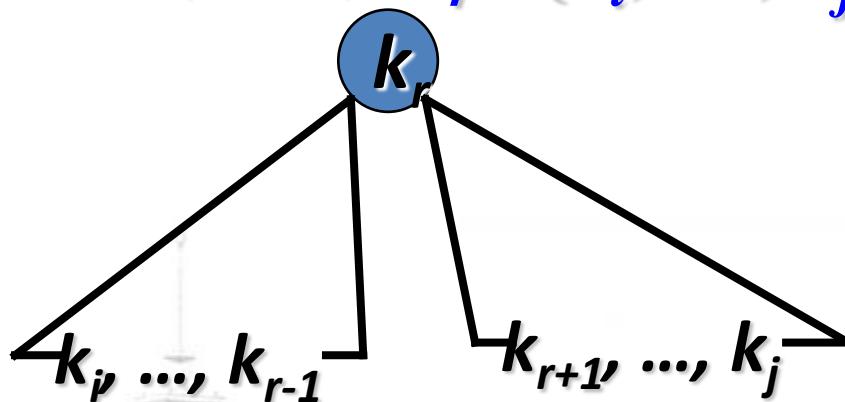


只要对于每个  $k_r \in K$ , 确定  $\{k_i, \dots, k_{r-1}\}$  和  $\{k_{r+1}, \dots, k_j\}$  的优化解, 我们就可以求出  $K$  的优化解.

如果  $r=i$ , 左子树  $\{k_i, \dots, k_{i-1}\}$  仅包含  $d_{i-1}$   
如果  $r=j$ , 右子树  $\{k_{r+1}, \dots, k_j\}$  仅包含  $d_j$

# 建立优化解的搜索代价递归方程

- 令  $E(i, j)$  为  $\{k_i, \dots, k_j\}$  的优化解  $T_{ij}$  的期望搜索代价
  - 当  $j=i-1$  时,  $T_{ij}$  中只有叶结点  $d_{i-1}$ ,  $E(i, i-1) = q_{i-1}$
  - 当  $j \geq i$  时, 这样一个  $k_r \in \{k_i, \dots, k_j\}$ :



当把左右优化子树放进  $T_{ij}$  时, 每个结点的深度增加1

$$E(i, j) = P_r + E(\text{左子树}) + W(i, r-1) + E(\text{右子树}) + W(r+1, j)$$

## • 计算 $W(i, r-1)$ 和 $W(r+1, j)$

由  $E(LT+1) = \sum_{l=i}^{r-1} (DEP_{\not\in}(k_l) + 2)p_l + \sum_{l=i-1}^{r-1} (DEP_{\not\in}(d_l) + 2)q_l$

$$E(LT) = \sum_{l=i}^{r-1} (DEP_{\not\in}(k_l) + 1)p_l + \sum_{l=i-1}^{r-1} (DEP_{\not\in}(d_l) + 1)q_l$$

**$E(i, j) = E(i, r-1) + E(r+1, j) + W(i, j)$**

即  $W(i, r-1) = E(LT+1) - E(LT) - \sum_{l=i}^{r-1} p_l + \sum_{l=i-1}^{r-1} q_l$

同理,  $W(r+1, j) = \sum_{l=i}^j p_l + \sum_{l=i-1}^j q_l$

**$W(i, j) = W(i, r-1) + W(r+1, j) + P_r$**

总之

$$E(i, j) = q_{i-1} \quad \text{If } j = i-1$$

$$E(i, j) = \min_{i \leq r \leq j} \{E(i, r-1) + E(r+1, j) + W(i, j)\} \quad \text{if } j \geq i$$

# 自下而上计算优化解的搜索代价

- $E(i, j) = \min_{i \leq r \leq j} \{E(i, r-1) + E(r+1, j) + W(i, j)\}$

$$q_0 = E(1,0) \quad E(1,1) \quad E(1,2) \quad E(1,3) \quad \boxed{E(1,4)}$$

$$q_1 = E(2,1) \quad E(2,2) \quad E(2,3) \quad E(2,4)$$

$$q_2 = E(3,2) \quad E(3,3) \quad E(3,4)$$

$$q_3 = E(4,3) \quad E(4,4)$$

$$q_4 = E(5,4)$$



- $W(i, i-1) = q_{i-1}, \quad W(i, j) = W(i, j-1) + p_j$   
+  $q_j$

$$\cancel{q_0 = W(1,0) \ W(1,1) \ W(1,2) \ W(1,3) \ W(1,4)}$$

$$\cancel{q_1 = W(2,1) \ W(2,2) \ W(2,3) \ W(2,4)}$$

$$\cancel{q_2 = W(3,2) \ W(3,3) \ W(3,4)}$$

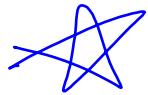
$$\cancel{q_3 = W(4,3) \ W(4,4)}$$

$$\cancel{q_4 = W(5,4)}$$

## • 算法

### • 数据结构

- $M[1:n+1; 0:n]$ : 存储优化解搜索代价
- $W[1: n+1; 0:n]$ : 存储代价增量
- $Root[1:n; 1:n]$ :  $root(i, j)$ 记录子问题  $\{k_i, \dots, k_j\}$  优化解的根



## Optimal-BST( $p, q, n$ )

For  $i=1$  To  $n+1$  Do

$$E(i, i-1) = q_{i-1};$$

$$W(i, i-1) = q_{i-1};$$

For  $l=1$  To  $n$  Do

For  $i=1$  To  $n-l+1$  Do

$$j=i+l-1;$$

$$E(i, j)=\infty;$$

$$W(i, j)=W(i, j-1)+p_j+q_j;$$

For  $r=i$  To  $j$  Do

$$t=E(i, r-1)+E(r+1, j)+W(i, j);$$

If  $t < E(i, j)$

Then  $E(i, j)=t$ ;  $Root(i, j)=r$ ;

Return  $E$  and  $Root$

# 思考：优化解的构造 算法

