HIT
CS&E

第四章
动态规划算法的设计与分析原理

程思瑶
计算机科学与技术学院

提要

4.1 动态规划算法的要素

4.2 最长公共子串发现算法

4.3 矩阵链乘法

4.4 凸多边形的最优三角抛分

4.5 0/1背包问题

4.6 最优二叉搜索树

参考资料

Introduction to Algorithms

第15章

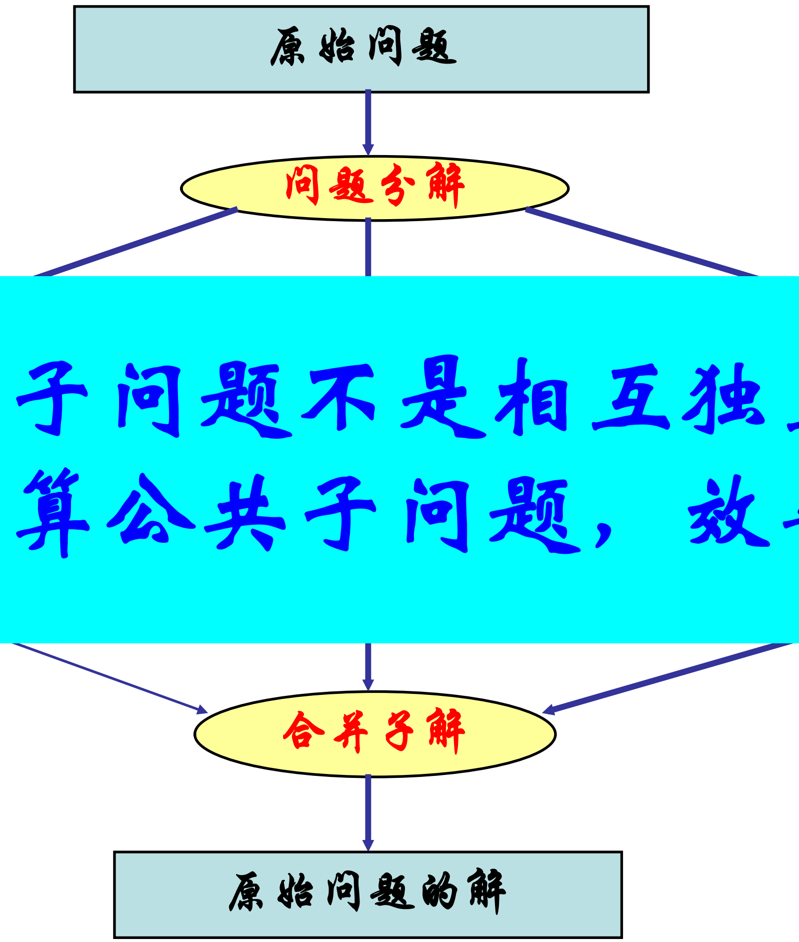
15.2, 15.3, 15.4, 15.5

4.1 动态规划算法的要素

Why?
What?
How?

研究动机(Why?)

- Divide-and-Conquer方法的问题



问题：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

研究动机(Why?)

一个例子：计算矩阵乘法$A_1 \times A_2 \times A_3 \times A_4$所需最小乘法的个数

$A_1 \times A_2 \times A_3 \times A_4$

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

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

可以考虑利用空间换时间——动态规划

适用范围及思想(What?)

- 适用范围

  ✓优化问题：给定一个代价函数，在解空间中搜索具有最小或最大代价的解

  ✓优化子结构(Optimal Substructure)：一个问题的优化解包含了子问题的优化解

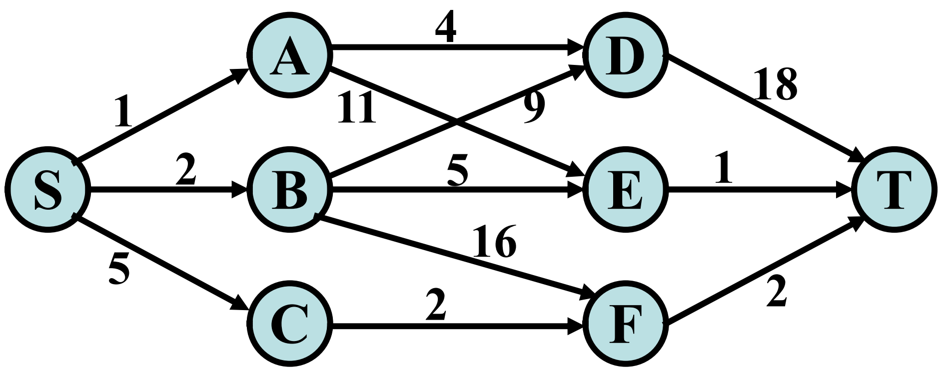
  ✓重叠子问题(Subteties)：问题的求解过程中，很多子问题的解将被多次使用

适用范围及思想(What?)

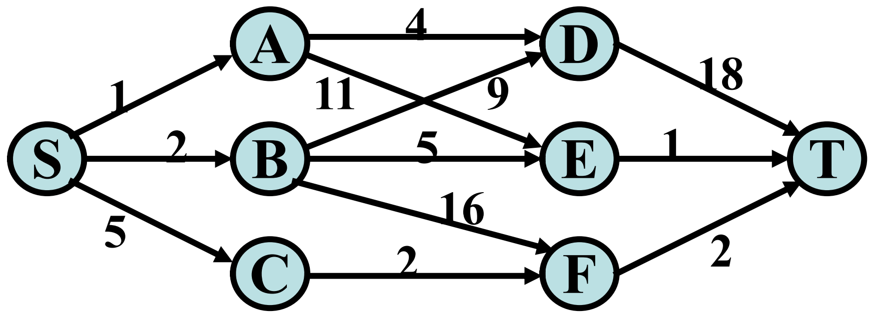
- 动态规划算法的主要思想
  - ✓将原始问题划分成一系列子问题
  - ✓求解每个子问题仅一次，并将其结果保存在一个表中，以后用到时直接存取，不重复计算，节省计算时间
  - ✓自底向上地计算

- 动态规划算法的特点
  - ✓利用空间换时间

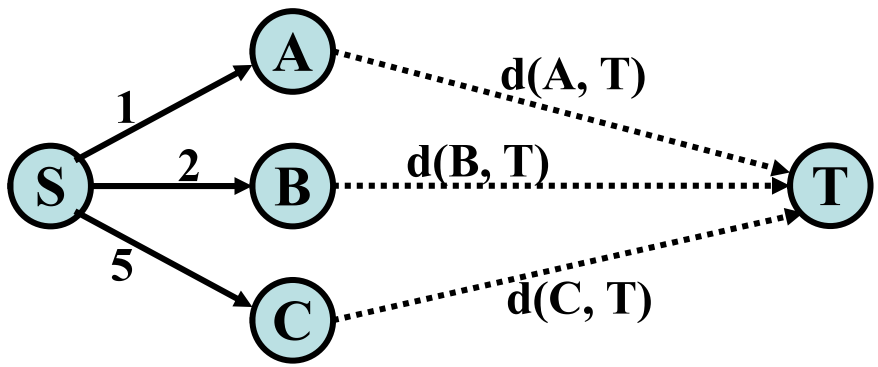
- **Dynamic Programming的实例**
  - **最短路径问题**



  - **Dynamic Programming求解过程**

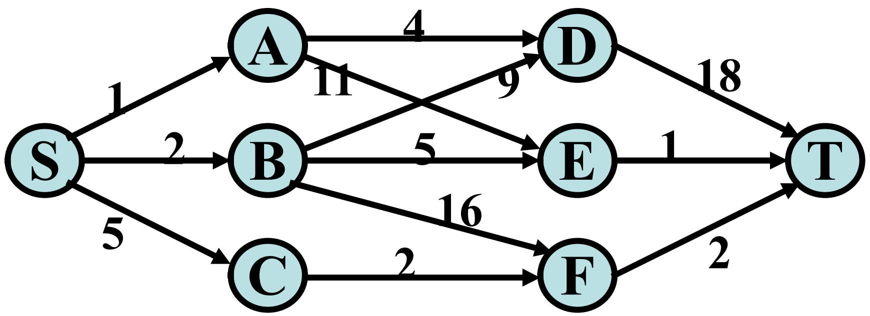


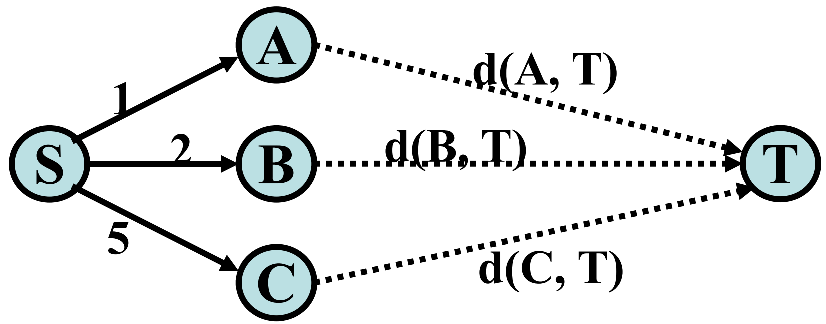
1：划分求d(S,T)问题为三个子问题



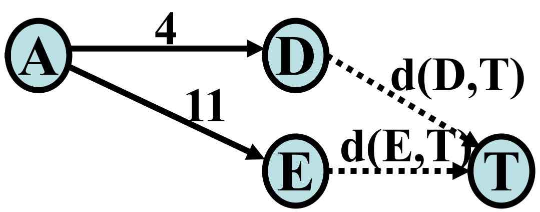
从S到T的最短路径长度为：

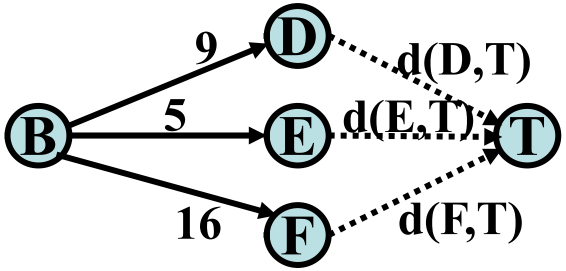
$d(S,T)=\min\{1+d(A,T),\ 2+d(B,T),\ 5+d(C,T)\}$

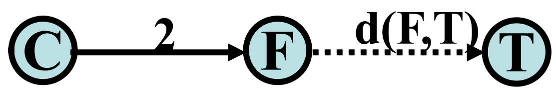




2：划分d(A,T)、d(B,T)、d(C,T)为6个子问题







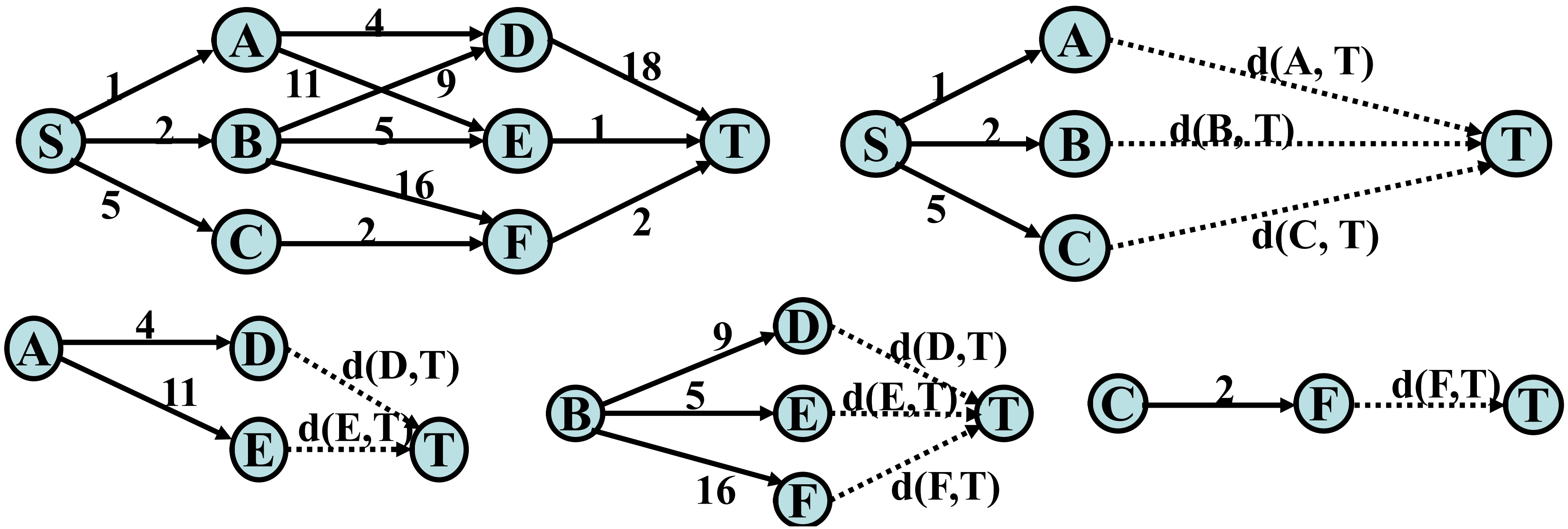
3: 求解最小子问题

d(A,T)=min{4+d(D,T), 11+d(E,T)}=min{22, 12}=12   <A,E,T>

d(B,T)=min{9+d(D,T), 5+d(E,T), 16+d(F,T)}=min{27, 6, 18}=6

<B,E,T>

d(C,T)=min{2+d(F,T)}=4   <C,F,T>



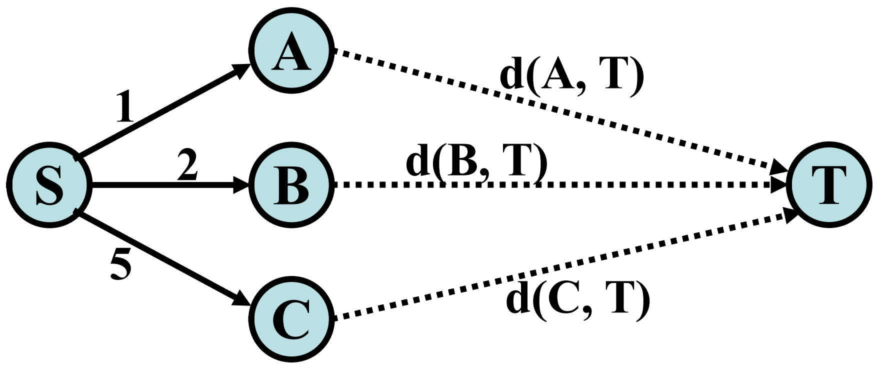
d(A,T)=22 <A,D,T>, d(B,T)=6 <B,E,T>, d(C,T)=4 <C,F,T>

4：最后确定从S到T的最短路径

d(S,T)=min{1+d(A,T), 2+d(B,T), 5+d(C,T)}=min{23, 8, 9}=8

<S, B, E, T>

- Dynamic Programming算法的设计步骤
  - 分析优化解的结构：划分子问题、优化子结构、子问题重叠性



  - 递归地定义最优解的代价

    $d(S,T)=\min\{1+d(A,T), 2+d(B,T), 5+d(C,T)\}$
  - 递归地划分问题，直至不可划分
  - 自底向上求解各个子问题：
    - 计算优化解代价并保存之
    - 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

4.2 最长公共子串发现算法

- 问题定义
- 问题求解
  - 优化解的结构分析
  - 建立优化解的代价递归方程
  - 递归地划分子问题
  - 自底向上计算优化解的代价
    记录优化解的构造信息
  - 构造优化解

问题的定义

- 子序列
  - $X=(A, B, C, B, D, B)$
  - $W=(B, D, A)$是X的子序列?
  - $Z=(B, C, D, B)$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的子序列也是Y的子序列。

最长公共子序列（*LCS*）问题

输入：$X = (x_1,x_2,\ldots,x_m)$，$Y = (y_1,y_2,\ldots y_n)$
输出：X与Y的最长公共子序列
$Z=(z_1, z_2, \ldots, z_k)$

最长公共子序列结构分析

- 第i前缀
  - 设$X=(x_1, x_2, \ldots, x_n)$是一个序列
    则$X_i=(x_1, \ldots, x_i)$是X的第i前缀

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

$$X = (x_1, x_2, \ldots, x_m)$$
$$Y = (y_1, y_2, \ldots y_n)$$

- 优化子结构的猜想

X和Y的LCS为$LCS_{XY}=(z_1, \ldots, z_k)$

If $x_m=y_n$

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

If $x_m \neq y_n$,

$$\left.\begin{array}{ll} z_k \neq x_m & LCS_{XY}=LCS_{X_{m-1}Y} \\ z_k \neq y_n & LCS_{XY}=LCS_{XY_{n-1}} \end{array}\right\} \rightarrow LCS_{XY}=\max\{LCS_{X_{m-1}Y}, LCS_{XY_{n-1}}\}$$

- 优化子结构

定理1（优化子结构）设$X=(x_1, ..., x_m)$、$Y=(y_1, ..., y_n)$是两个序列，$LCS_{XY}=(z_1, ..., z_k)$是X与Y的LCS，我们有：

(1) 如果$x_m=y_n$, 则$z_k=x_m=y_n$, $LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + <x_m=y_n>$,

$LCS_{X_{m-1}Y_{n-1}}$是X_{m-1}和Y_{n-1}的LCS.

(2) 如果$x_m \neq y_n$，且$z_k \neq x_m$，则LCS_{XY}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LCS_{XY}是X与Y_{n-1}的LCS，即

$$LCS_{XY} = LCS_{XY_{n-1}}$$

证明：

(1). $X=<x_1, \ldots, x_{m-1}, x_m>$, $Y=<y_1, \ldots, y_{n-1}, x_m>$，则

$$z_k=x_m=y_n \text{ 且 } LCS_{XY}=LCS_{X_{m-1}Y_{n-1}}+<x_m=y_n>.$$

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，$z_k=x_m=y_n$。

设存在X_{m-1}与Y_{n-1}的非最长公共子序列Z_{k-1}，使得

$$LCS_{XY}=Z_{k-1}+<x_m=y_n>,$$

则由于$|Z_{k-1}|<|LCS_{X_{m-1}Y_{n-1}}|$,

$|LCS_{XY}=Z_{k-1}+<x_m=y_n>|<|LCS_{X_{m-1}Y_{n-1}}+<x_m=y_n>|$,

与LCS_{XY}是LCS矛盾。

(2) $X=<x_1, \ldots, x_{m-1}, x_m>$, $Y=<y_1, \ldots, y_{n-1}, y_n>$，

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

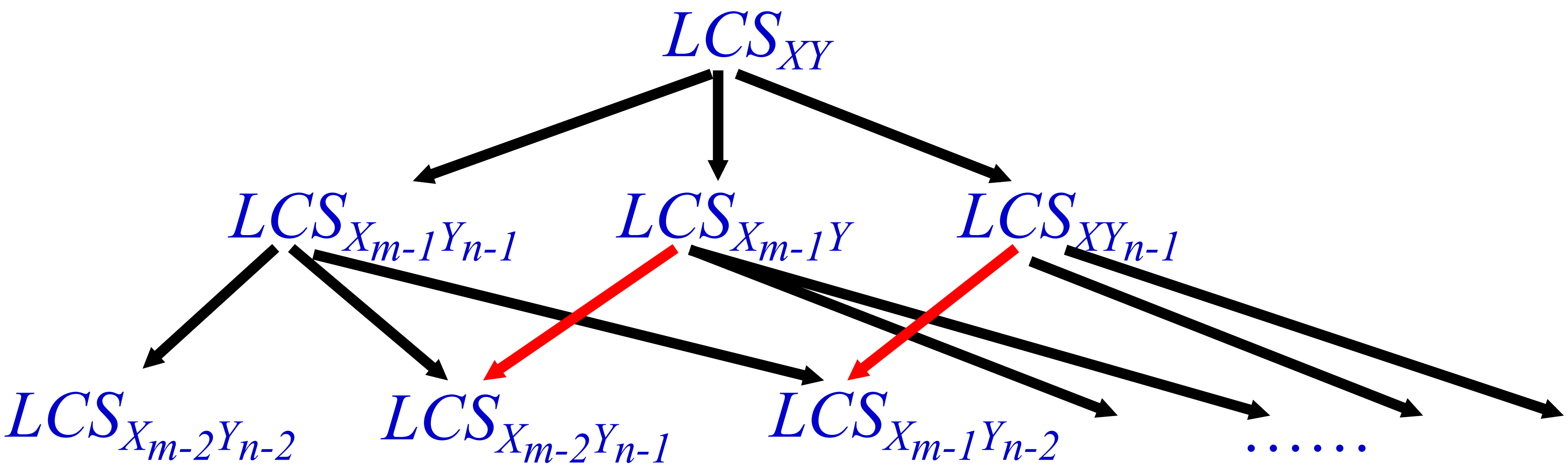
由于$z_k \neq x_m$，$Z=LCS_{XY}$是X_{m-1}与Y的公共子序列。

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) 证明同(2)。

- 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

$$C[i, j] = 0 \qquad \text{if } i=0 \text{ 或 } j=0$$

$$C[i, j] = C[i-1, j-1] + 1 \qquad \text{if } i, j>0 \text{ and } x_i = y_j$$

$$C[i, j] = Max(C[i, j-1], C[i-1, j]) \qquad \text{if } i, j>0 \text{ and } x_i \neq y_j$$

递归划分与自底向上求解

- 基本思想

$$C[i, j] = 0 \text{ , if } i=0 \text{ 或 } j=0$$

$$C[i, j] = C[i-1, j-1] + 1 \quad \text{if } i, j>0 \text{ and } x_i = y_j$$

$$C[i, j] = Max(C[i, j-1], C[i-1, j]) \text{ if } i, j>0 \text{ and } x_i \neq y_j$$

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

自底向上计算优化解代价

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

- 递归划分子问题与自底向上求解过程

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

- 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

  $B[1:m,1:n]$: $B[i,j]$记录优化解的信息

记录优化解信息

- $C[i, j] = 0, \quad i=0$ 或 $j=0$
- $C[i, j] = C[i-1, j-1] + 1, \quad x_i = y_j$
- $C[i, j] = Max(C[i, j-1], C[i-1, j]), \ x_i \neq y_j$

		y_j	B	D	C	A	B	A
$i=0$	x_i	0	0	0	0	0	0	0
	A	0	↑0	↑0	↑0	↖1	←1	↖1
	B	0	↖1	←1	←1	↑1	↖2	←2
	C	0	↑1	↑1	↖2	←2	↑2	↑2
	B	0	↖1	↑1	↑2	↑2	↖3	←3
	D	0	↑1	↖2	↑2	↑2	↑3	↑3
	A	0	↑1	↑2	↑2	↖3	↑3	↖4
	B	0	↖1	↑2	↑2	↑3	↖4	↑4

$j=0$

```
LCS-length(X, Y)
m←length(X);  n←length(Y);
For  i←0  To  m  Do
        C[i,0]←0;
For  j←1  To  n   Do
        C[0,j]←0;
For  i←1  To  m  Do
    For  j←1  To  n  Do
        If  x_i = y_j
        Then C[i,j]←C[i-1,j-1]+1;
                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]←"←";
  Return C and B.
```

- $C[i, j] = 0, \ i=0$ 或 $j=0$
- $C[i, j] = C[i\text{-}1, j\text{-}1] + 1, \ x_i = y_j$
- $C[i, j] = Max(C[i, j\text{-}1], C[i\text{-}1, j]), \ x_i \neq y_j$

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

	y_j	B	D	C	A	B	A
x_i	0	0	0	0	0	0	0
A	0	↑0	↑0	↑0	↖1	←1	↖1
B	0	↖1	←1	←1	↑1	↖2	←2
C	0	↑1	↑1	↖2	←2	↑2	↑2
B	0	↖1	↑1	↑2	↑2	↖3	←3
D	0	↑1	↖2	↑2	↑2	↑3	↑3
A	0	↑1	↑2	↑2	↖3	↑3	↖4
B	0	↖1	↑2	↑2	↑3	↖4	↑4

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*的*Inverse*

	y_j	B	D	C	A	B	A
x_i	0	0	0	0	0	0	0
A	0	↑0	↑0	↑0	↖1	←1	↖1
B	0	↖1	←1	←1	↑1	↖2	←2
C	0	↑1	↑1	↖2	←2	↑2	↑2
B	0	↖1	↑1	↑2	↑2	↖3	←3
D	0	↑1	↖2	↑2	↑2	↑3	↑3
A	0	↑1	↑2	↑2	↖3	↑3	↖4
B	0	↖1	↑2	↑2	↑3	↖4	↑4

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, n, m)
可打印出X与Y的LCS
n=length(X)
m=length(Y)

```
LCS-length(X, Y)
m←length(X); n←length(Y);
For i←0 To m Do
        C[i,0]←0;
For j←1 To n Do
        C[0,j]←0;
For i←1 To m Do
    For j←1 To n Do
        If x_i = y_j
        Then C[i,j]←C[i-1,j-1]+1;
              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]←"←";
  Return C and B.
```

x_i \ y_j		B	D	C	A	B	A
	0	0	0	0	0	0	0
A	0	↑0	↑0	↑0	↖1	←1	↖1
B	0	↖1	←1	←1	↑1	↖2	←2
C	0	↑1	↑1	↖2	←2	↑2	↑2
B	0	↖1	↑1	↑2	↑2	↖3	←3
D	0	↑1	↖2	↑2	↑2	↑3	↑3
A	0	↑1	↑2	↑2	↖3	↑3	↖4
B	0	↖1	↑2	↑2	↑3	↖4	↑4

算法复杂性

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

4.3 矩阵链乘法

问题定义

- 输入：$<A_1, A_2, ..., A_n>$，A_i是$p_{i-1} \times p_i$矩阵
- 输出：计算$A_1 \times A_2 \times ... \times A_n$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$O(pqr)$

问题定义(实例)

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

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

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

....

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

问题定义(实例)

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

T1:(A1×A2)×A3 =10×100×5+10×5×50 =7500

T2:A1×(A2×A3) =100×5×50+10×100×50 =75000

结论：不同计算顺序有不同的代价

HIT
CS&E

问题定义

- 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

$$(A_1 \times \ldots \times A_k) \times (A_{k+1} \times \ldots \times A_{n-1} \times A_n)$$

$p(n)=1$ if $n=1$

如此之大的解空间是无法用枚举方法求出最优解的！

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

问题定义

- 该问题的特点

$A_1 \times A_2 \times A_3 \times A_4$

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

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

具有子问题重叠性

可以考虑应用动态规划算法求解

求解矩阵链乘法问题的Dynamic Programming算法分为如下几步

- 分析优化解的结构
- 递归地定义最优解的代价
- 递归地划分子问题，直至不可分
- 自底向上地求解各个子问题
  - 计算优化解的代价并保存之
  - 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

具有优化子结构：
问题的优化解包括子问题优化解

- 两个记号
  - $A_{i\sim j}=A_i \times A_{i+1} \times \dots \times A_j$
  - $cost(A_{i\sim j})$=计算$A_{i\sim j}$的代价
- 优化解的结构

定理. 若计算$A_{1\sim n}$的优化顺序在k处断开矩阵链, 即 $A_{1\sim n}=A_{1\sim k} \times A_{k+1\sim n}$，则在$A_{1\sim n}$的优化顺序中，对应于子问题$A_{1\sim k}$的解必为$A_{1\sim k}$的优化解,对应于子问题$A_{k+1\sim n}$的解必为$A_{k+1\sim n}$的优化解.

递归定义最优解代价

- 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
- 优化解$(A_i \ldots A_k)(A_{k+1} \ldots A_j)$的代价方程

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

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

  其中，A_i是$p_{i-1}\times p_i$矩阵，

  $A_{i\sim k}$和$A_{k+1\sim j}$分别是$p_{i-1}\times p_k$和$p_k\times p_j$矩阵，

  $p_{i-1}p_kp_j$是计算$A_{i\sim k}\times A_{k+1\sim j}$所需乘法数.

递归定义最优解代价

$$(A_i \ldots A_k)(A_{k+1} \ldots A_j)$$

考虑到所有的k，优化解的代价方程为

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

$m[i, j]= \min_{i \le k < j}\{ m[i, k]+m[k+1, j]+p_{i-1}p_kp_j\}$

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

递归地划分子问题

$$m[i,j] = \min_{i \le k < j}\{ m[i,k] + m[k+1,j] + p_{i-1}p_kp_j \}$$

m[i,i]	*m[i,i+1]*	……	*m[i,j-1]*	*m[i,j]*
				m[i+1,j]
				m[i+2,j]
				……
				m[j, j]

递归地划分子问题

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

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]

自底向上计算优化解的代价

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

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

HIT
CS&E

自底向上计算优化解的代价

Matrix-Chain-Order(*n*)

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

  $m[i, i]=0;$

FOR $l=2$ **TO** n **DO** /* 计算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$m[i,j]$ */

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

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

Return m.

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]

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

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

获取构造最优解的信息

$$m[i,j]=\min_{i\le k<j}\{\, m[i,k]+m[k+1,j]+p_{i-1}p_kp_j \,\}$$

- **Matrix-Chain-Order(*p*)**
- ***n*=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]=∞*;**
- **   FOR *k←i* To *j-1* DO**
- ** $q = m[i,k]+m[k+1,j]+p_{i-1}\times p_k\times p_j$**
- **   IF *q<m[i, j]* THEN *m[i,j]=q*，*s[i,j]=k*;**
- **Return *m* and *s*.**

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

时间复杂性：$O(n^3)$

构造最优解

$S[i, j]$记录$A_i \ldots A_j$的最优划分处；

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

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

Print-Optimal-Parens(*s*, *i*, *j*)

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*)

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$O(n^2)$

4.4 最优三角抛分

- 问题定义
- 问题求解
  - 优化解的结构分析
  - 建立优化解代价的递归方程
  - 递归地划分子问题
  - 自底向上计算优化解的代价
    记录优化解的构造信息
  - 构造优化解

问题定义

- 多边形
  - 多边形表示为顶点坐标集P=(v_0,v_1,…v_n)或顶点序列$v_0v_1…v_{n-1}v_n$
- 简单多边形
  - 除了顶点以外没有任何边交叉点的多边形
- 弦
  - 多边形P上的任意两个不相邻结点v_i、v_j所对应的线段 v_iv_j称为弦

问题定义

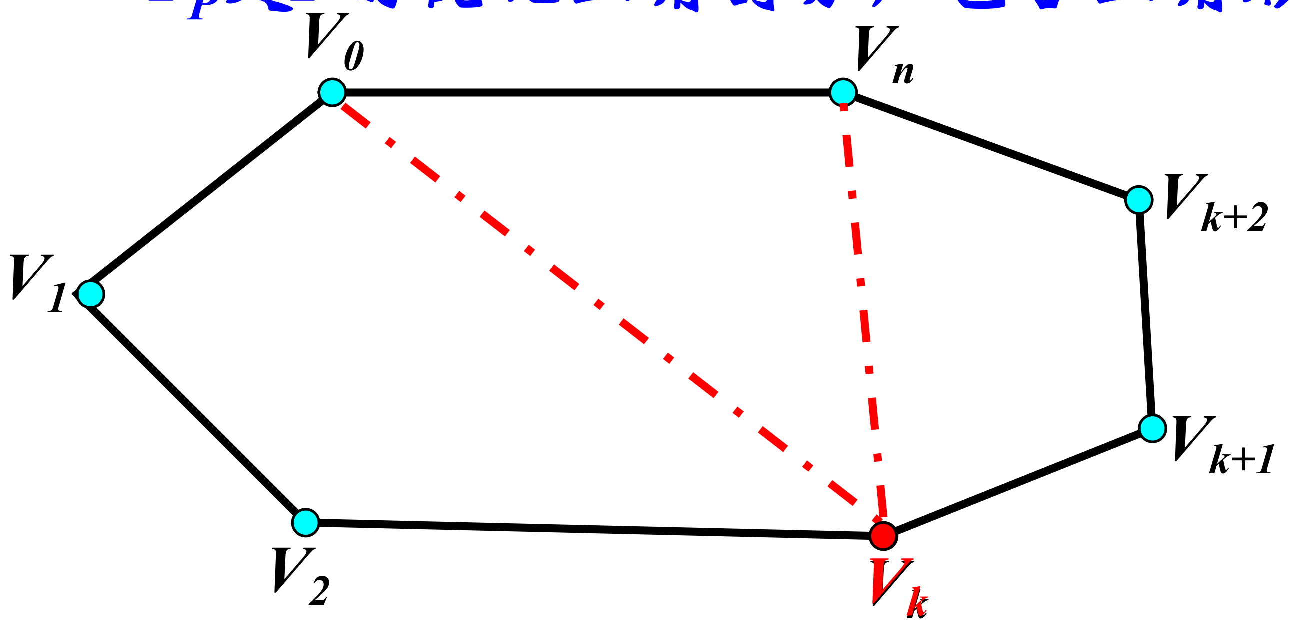
- 三角剖分

  一个多边形P的三角剖分是将P划分为不相交三角形的弦的集合

- 优化三角剖分问题
  - 输入：简单多边形P和代价函数W
  - 输出：求P的三角剖分T，使得代价$\sum_{s \in S_T} W(s)$最小，其中S_T是T所对应的三角形集合

优化解结构分析

- 设
  - $P=(v_0,v_1,\ldots v_n)$是$n+1$个顶点的多边形
  - T_P是P的优化三角剖分，包含三角形$v_0v_k v_n$



$$T_P=T(v_0, \ldots, v_k) \cup T(v_k, \ldots, v_n) \cup \{v_0v_k, v_kv_n\}$$

优化解结构分析

- 三角剖分问题具有优化子结构

定理. 设$P=(v_0,v_1,...v_n)$是$n+1$个顶点的多边形. 如果T_P是P的包含三角形$v_0v_kv_n$的优化三角剖分，即

$$T_P=T(v_0, ..., v_k)\cup T(v_k, ..., v_n)\cup \{v_0v_k, v_kv_n\},$$

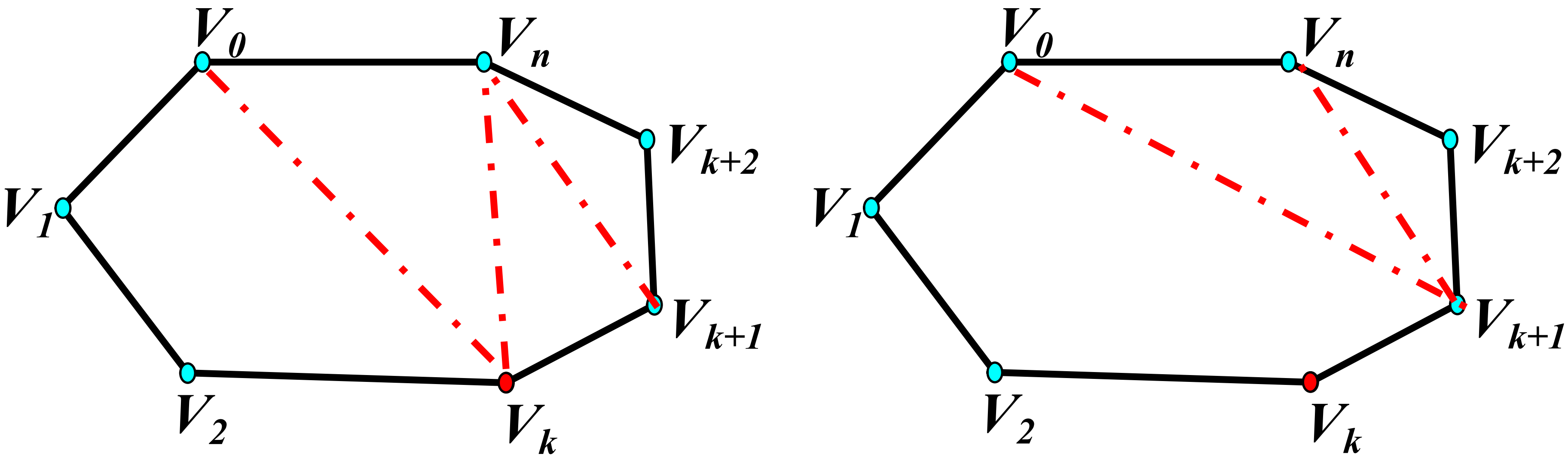
则

(1). $T(v_0, ..., v_k)$是$P_1=(v_0,v_1, ...,v_k)$的优化三角剖分,

(2). $T(v_k, ..., v_n)$是$P_2=(v_k,v_{k+1},...,v_n)$的优化三角剖分。

优化解结构分析

- 三角剖分问题具有子问题重叠性

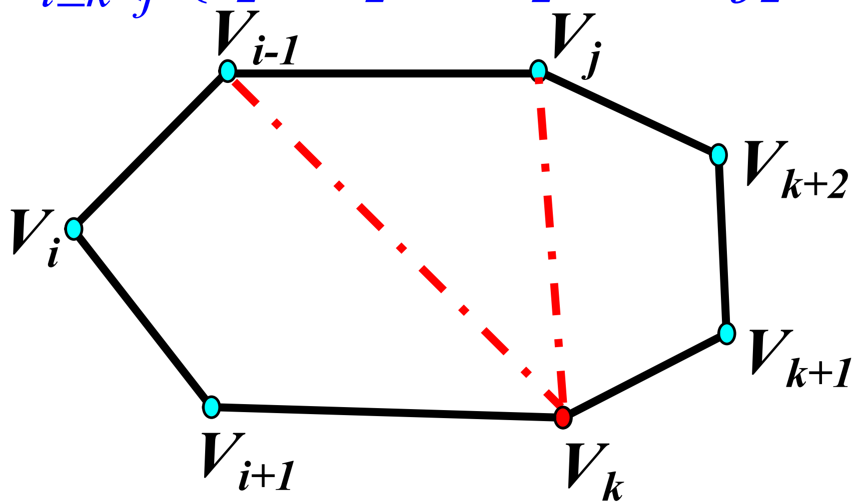


优化三角剖分的代价函数

- 设 $t[i,j] = <v_{i-1}, v_i, ..., v_j>$ 的优化三角剖分代价

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

$t[i, j] = min_{i \le k < j} \{t[i, k] + t[k+1, j] + w(\Delta v_{i-1} v_k v_j)\}$



$t[i, k] = <v_{i-1}, v_i, ..., v_k>$ 的优化三角剖分代价

$t[k+1, j] = <v_k, v_{k+1}, ..., v_j>$ 的优化三角剖分代价

优化三角剖分动态规划算法

- 优化三角剖分与矩阵链乘法问题一致.
- Homework 1:

修改算法

Matrix-chain-Order

Print-Optimal-Parens

使其计算$t[i,j]$并构造优化三角剖分解

4.5 0/1 背包问题

- 问题定义
- 问题求解
  - 优化解的结构分析
  - 建立优化解代价的递归方程
  - 递归地划分子问题
  - 自底向上计算优化解的代价记录优化解的构造信息
  - 构造优化解

问题定义

给定n种物品和一个背包，物品i的重量是w_i，价值v_i，背包承重为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：$(x_1, x_2, \ldots, x_n)$, $x_i \in \{0, 1\}$, 满足
  $\Sigma_{1\leq i\leq n} w_i x_i \leq C$, $\Sigma_{1\leq i\leq n} v_i x_i$ 最大

Naïve方法：

每个物品有两种选择：1(装)或0(不装)

n个物品共2^n个装取方案

每个装取方案的计算代价为n

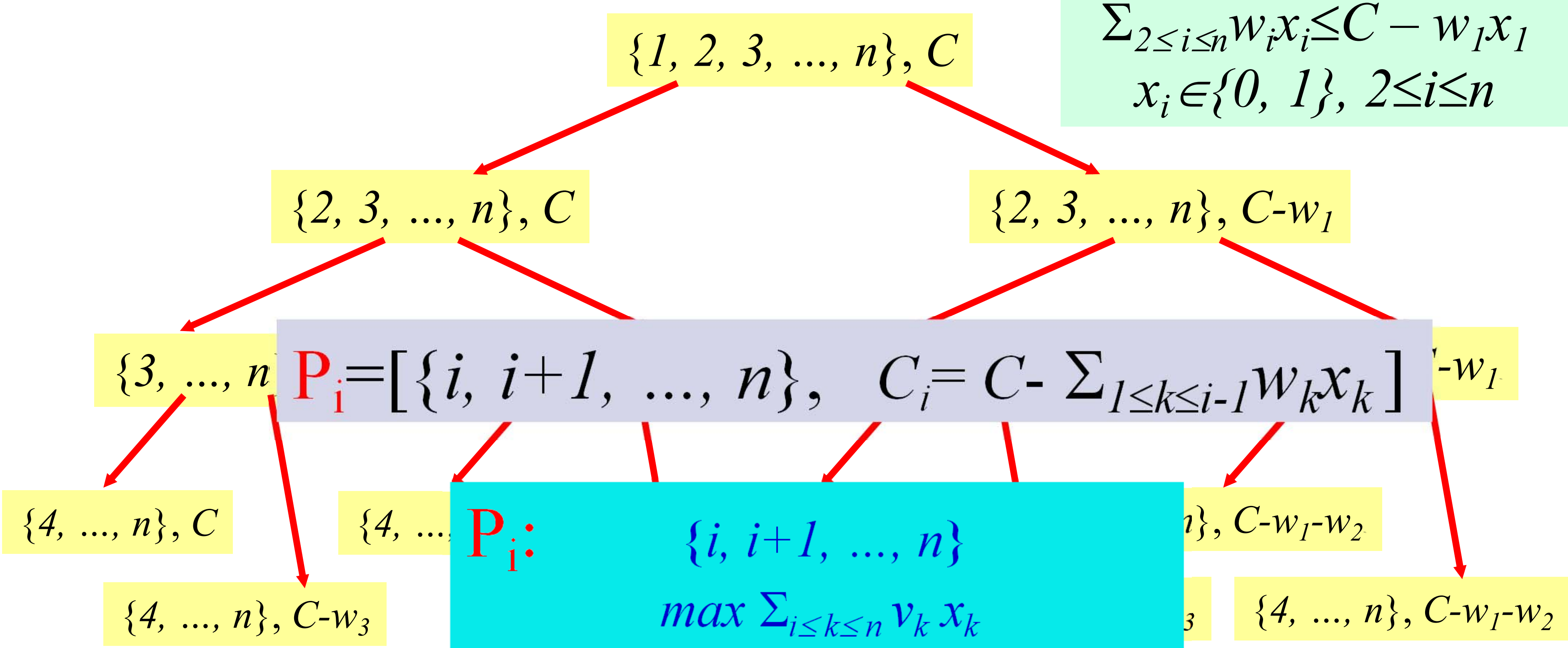
总计算代价为$O(n2^n)$

- **优化解的结构分析**
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价记录优化解的构造信息
- 构造优化解

问题优化子结构

问题的划分

$$P_2: \max \sum_{2 \le i \le n} v_i x_i$$
$$\sum_{2 \le i \le n} w_i x_i \le C - w_1 x_1$$
$$x_i \in \{0, 1\}, \ 2 \le i \le n$$



$$P_i = [\{i, i+1, \ldots, n\}, \ C_i = C - \sum_{1 \le k \le i-1} w_k x_k]$$

P_i: $\{i, i+1, \ldots, n\}$

$$\max \sum_{i \le k \le n} v_k x_k$$
$$\sum_{i \le k \le n} w_k x_k \le C_i = C - \sum_{1 \le k \le i-1} w_k x_k$$
$$x_k \in \{0, 1\}, \ i \le k \le n$$

优化解结构的分析

定理 如果$S_i=(y_i, y_{i+1}, \ldots, y_n)$是0-1背包子问题$P_i=[\{i, i+1, \ldots, n\}, C_i=C-\sum_{1\leq k\leq i-1} w_k y_k]$的优化解，则$(y_{i+1}, \ldots, y_n)$是如下子问题$P_{i+1}$的优化解：

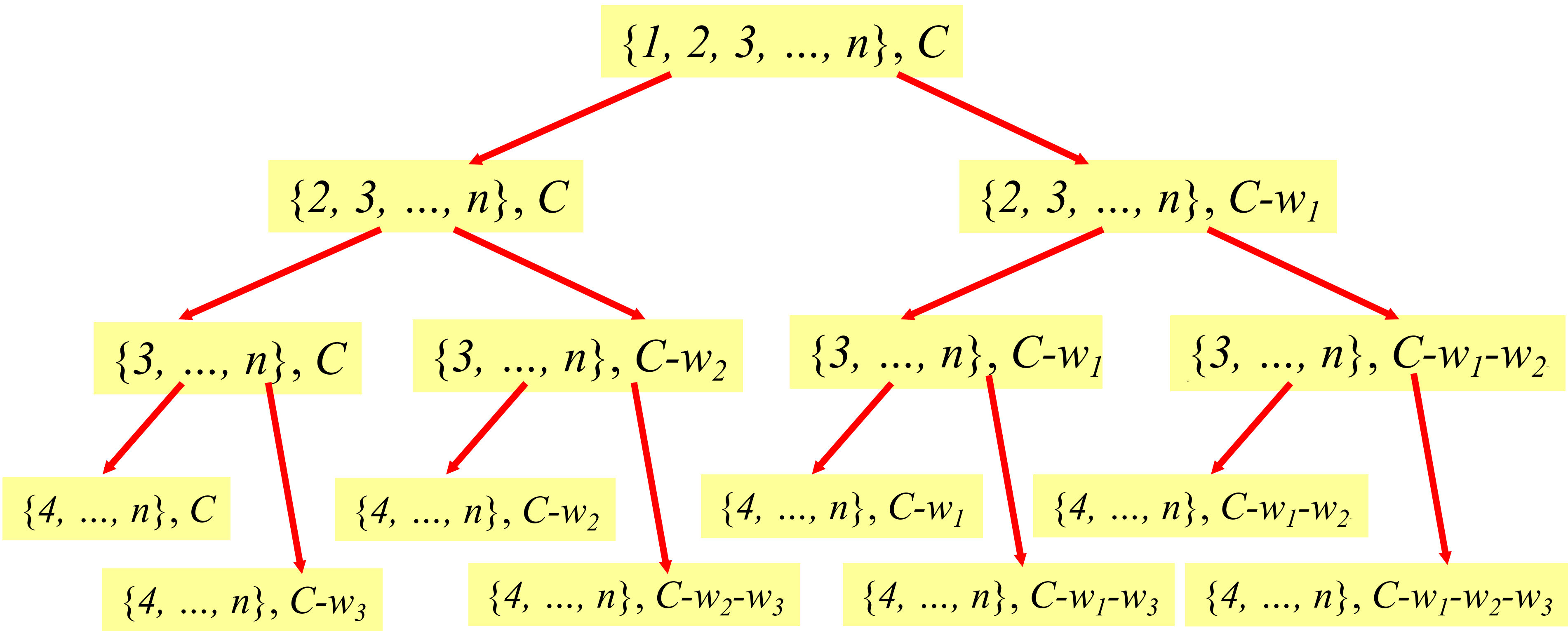
$$\max \sum_{i+1\leq k\leq n} v_k x_k$$

$$\sum_{i+1\leq k\leq n} w_k x_k \leq C_i - w_i y_i$$

$$x_k \in \{0, 1\},\ i+1\leq k \leq n$$

证明： 如果$S_{i+1}=(y_{i+1}, \ldots, y_n)$不是子问题$P_{i+1}$的优化解，则存在$S'_{i+1}=(z_{i+1}, \ldots, z_n)$是$P_{i+1}$的更优解。$S'_i=(y_i, z_{i+1}, \ldots, z_n)$是问题$P_i$之比$S_i$更优的解，与$S_i$优化矛盾。

子问题重叠性



当w_i皆为1时，存在大量重叠子问题

- 优化解的结构分析
- **建立优化解代价的递归方程**
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

- 定义代价矩阵 m

  矩阵元素 $m(i, j)$ 是子问题 $[(i, i+1, \ldots, n), j]$ 的优化解 $(x_i, x_{i+1}, \ldots, x_n)$ 的代价，

- 形式地

$$m(i, j)= \Sigma_{i\le k\le n} v_k x_k$$

问题 $\max \Sigma_{i\le k\le n} v_k x_k$

$\Sigma_{i\le k\le n} w_k x_k \le j$

$x_k \in \{0, 1\}, \quad i\le k \le n$

的最优解代价为 $m(i, j)=\Sigma_{i\le k\le n} v_k x_k$.

- 递归方程：

$[\{i, i+1, \ldots, n\}, j]$

$0 \le j < w_n$

总结：

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

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

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

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

$m(n, j) = 0$

$j \ge w_n,$

$$m(n, j) = v_n$$

- 优化解的结构分析
- 建立优化解代价的递归方程
- **递归地划分子问题**
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

HIT
CS&E

$$m(i, j) = m(i+1, j),\ 0 \le j < w_i$$
$$m(i, j) = \max\{m(i+1, j),\ m(i+1, j-w_i)+v_i\},\ j \ge w_i$$
$$m(n, j) = 0,\quad 0 \le j < w_n$$
$$m(n, j) = v_n,\quad j \ge w_n$$

	$m(i, j)$
$m(i+1, j-w_i)$	$m(i+1, j)$

令n=4

$m(1,C)$

$m(2,C-w_1)$ … $m(2,C)$

… $m(3, C-w_1-w_2)$ … $m(3,C-w_1)$ … $m(3,C-w_2)$ … $m(3,C)$

… $m(4, C-w_2-w_3)$ … $m(4,C-w_2)$ … $m(4,C-w_3)$ … $m(4,C)$

- 优化解的结构分析
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

HIT
CS&E

$$m(i, j) = m(i+1, j),\ 0 \le j < w_i$$
$$m(i, j) = \max\{m(i+1, j),\ m(i+1, j-w_i)+v_i\},\ \ j \ge w_i$$
$$m(n, j) = 0,\quad 0 \le j < w_n$$
$$m(n, j) = v_n,\quad j \ge w_n$$

令w_i=整数, $n=4$

	$m(i, j)$
$m(i+1, j-w_i)$	$m(i+1, j)$

$m(1,C)$

$m(2, 0)$ … $m(2,w_2-1)$ $m(2, w_2)$ … $m(2,C-1)$ $m(2,C)$

$m(3, 0)$ … $m(3,w_3-1)$ $m(3, w_3)$ … $m(3,C-1)$ $m(3,C)$

$m(4, 0)$ … $m(4,w_4-1)$ $m(4, w_4)$ … $m(4,C-1)$ $m(4,C)$

• 算法

(设 w_i-$1<C$)

$$m(i, j) = m(i+1, j),\ 0 \le j < w_i$$
$$m(i, j) = \max\{m(i+1, j),\ m(i+1, j-w_i)+v_i\},\ \ j \ge w_i$$
$$m(n, j) = 0,\quad 0 \le j < w_n$$
$$m(n, j) = v_n,\quad j \ge w_n$$

							$m(1,C)$
$m(2, 0)$	…	$m(2,w_2$-$1)$	$m(2, w_2)$	…	$m(2,C$-$1)$		$m(2,C)$
$m(3, 0)$	…	$m(3,w_3$-$1)$	$m(3, w_3)$	…	$m(3,C$-$1)$		$m(3,C)$
$m(4, 0)$	…	$m(4,w_4$-$1)$	$m(4, w_4)$	…	$m(4,C$-$1)$		$m(4,C)$

For $j=0$ To w_n-1 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 $j=0$ To w_i-1 Do
        $m[i, j] = m[i+1, j]$;
    For $j=w_i$ To C Do
        $m[i, j]=\max\{m[i+1, j], m[i+1, j-w_i]+v_i\}$;
If $C<w_1$ Then $m[1, C]=m[2, C]$;
    Else $m[1, C]=\max\{m[2, C], m[2, C-w_1]+v_1\}$;

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

$m[n, j] = 0$;

For $j=w_n$ To C Do

$m[n, j] = v_n$;

For $i=n\text{-}1$ To 2 Do

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

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

For $j=w_i$ To C Do

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

If $C<w_1$ Then $m[1, C]=m[2, C]$;

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

HIT
CS&E

- 优化解的结构分析
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

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 $x_1 = 1$;

	$m(i, j)$
$m(i+1, j\text{-}w_i)$	$m(i+1, j)$

2. 如果 $x_1=0$，由$m(2, C)$继续构造最优解；
3. 如果 $x_1=1$，由$m(2, C\text{-}w_1)$继续构造最优解.

$m(1,C)$

Homework: 给出构造最优解的详细精确算法

$m(3, C\text{-}w_1\text{-}w_2)$ $m(3,C\text{-}w_1)$ $m(3,C\text{-}w_2)$ … $m(3,C)$

… $m(4, C\text{-}w_2w_3)$ … $m(4,C\text{-}w_2)$ $m(4,C\text{-}w_3)$ $m(4,C)$

算法复杂性

- 时间复杂性
  - 计算代价时间
    - $O(Cn)$
  - 构造最优解时间：
    $O(Cn)$
  - 总时间复杂性为：
    $O(Cn)$
- 空间复杂性
  - 使用数组m
  - 需要空间$O(Cn)$

For $j=0$ To $\min(w_n-1, C)$ Do
  $m[n, j] = 0;$
For $j=w_n$ To C Do
  $m[n, j] = vn;$
For $i=n-1$ To 2 Do
  For $j=0$ To $\min(w_i-1, C)$ Do
    $m[i, j] = m[i+1, j];$

这是一个伪多项式算法！

当$C=2^n$时：

$T(n)=O(n2^n)$

当w_i不限定为正整数时：

$T(n)=O(2^n)$

3. If $x_1=1$，由$m(2, C-w_1)$继续构造x_2;

……

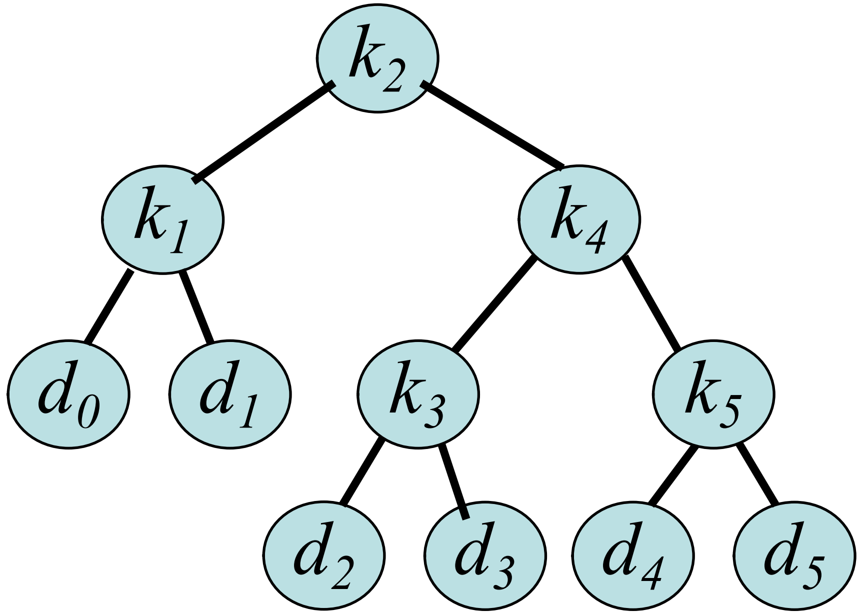
部分背包问题NP-C？

4.6 最优二叉搜索树的构建

- 问题定义
- 问题求解
  - 优化解的结构分析
  - 建立优化解代价的递归方程
  - 递归地划分子问题
  - 自底向上计算优化解的代价
    记录优化解的构造信息
  - 构造优化解

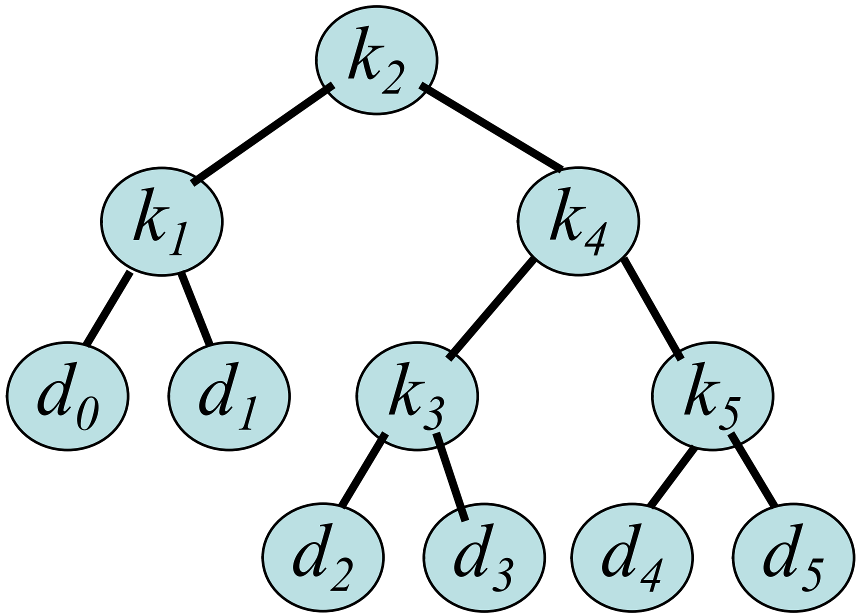
问题的定义

- 二叉搜索树T
  - 结点
    - $K=\{k_1, k_2, \ldots, k_n\}$
    - $D=\{d_0, d_1, \ldots, d_n\}$
    - d_i对应区间(k_i, k_{i+1})
      d_0对应区间$(-\infty, k_1)$
      d_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



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

- 搜索树的期望代价



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

- 问题的定义

输入：$K=\{k_1, k_2, \ldots, k_n\}, k_1< k_2< \ldots < k_n$,

$P=\{p_1, p_2, \ldots, p_n\}$, p_i为搜索k_i的概率

$Q=\{q_0, q_1, \ldots, q_n\}$, q_i为搜索值d_i的概率

输出：构造K的二叉搜索树T, 最小化

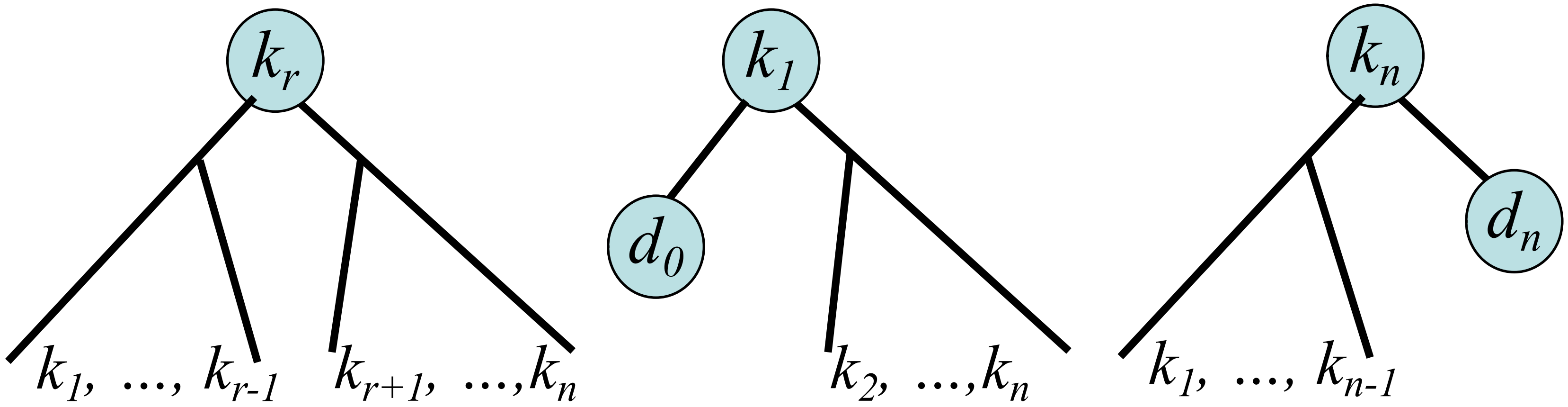
$$E(T)=\sum_{i=1}^{n}(DEP_T(k_i)+1)p_i+\sum_{j=0}^{n}(DEP_T(d_i)+1)q_i$$

- **优化解的结构分析**
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

优化二叉搜索树结构的分析

- 优化解的结构观察

$K=\{k_1, k_2, \ldots, k_n\}$的优化解的根必为$K$中某个$k_r$



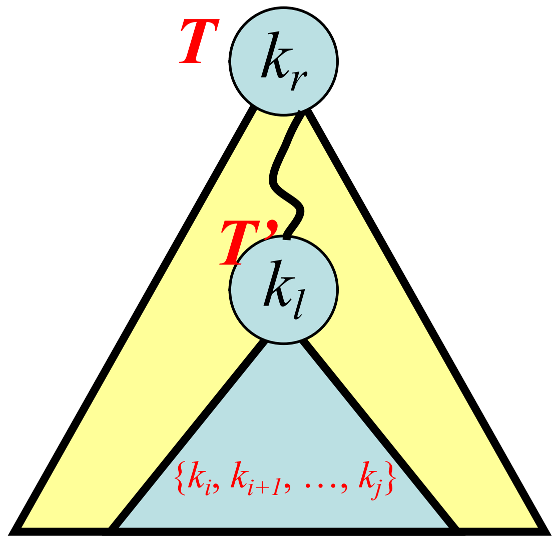
如果$r=1$, 左子树仅包含d_0
如果$r=n$, 右子树仅包含d_n

- 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$\{k_i, k_{i+1}, \ldots, k_j\}$的子树$T'$, 则$T'$是关于关键字集合$\{k_i, k_{i+1}, \ldots, k_j\}$的子问题的优化解.

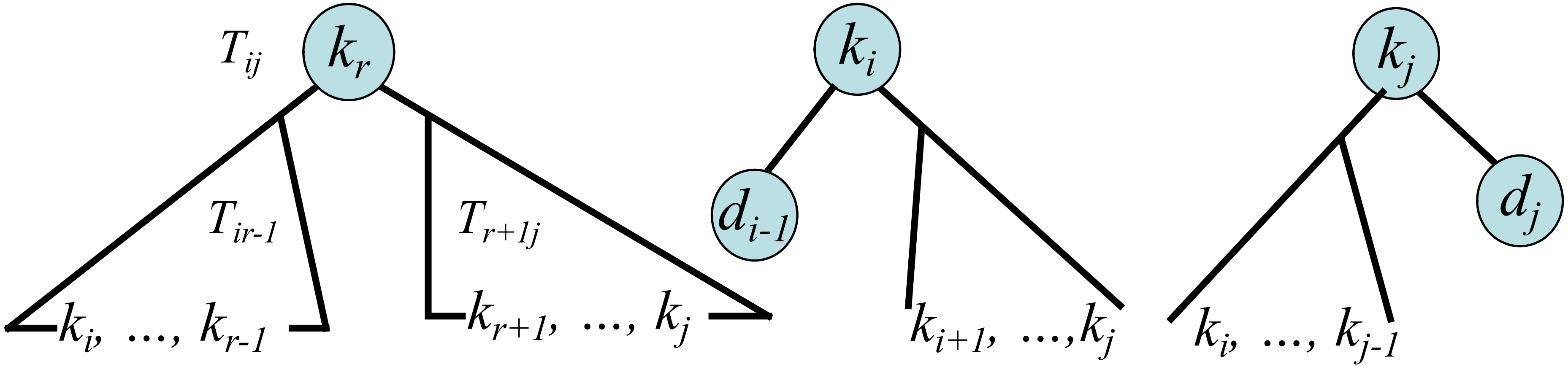
证明： 若不然，必有关键字集$\{k_i, k_{i+1}, \ldots, k_j\}$子树$T''$，$T''$的期望搜索代价低于$T'$.

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

$K=\{k_i, k_{i+1}, \ldots, k_j\}$ 的优化解 T_{ij} 的根必为 K 中某个 k_r

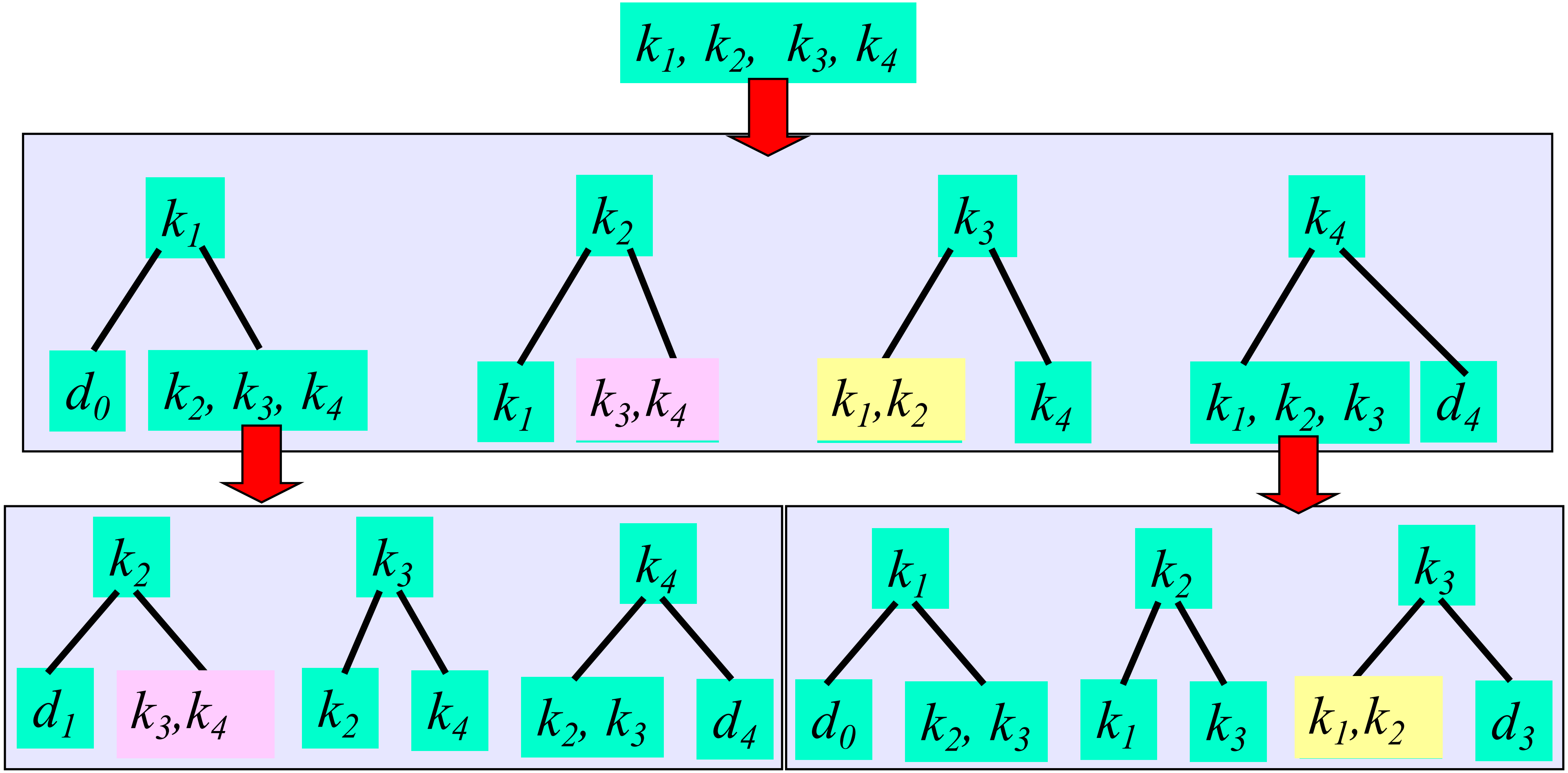


只要对于每个 $k_r \in K$, 确定 $\{k_i, \ldots, k_{r-1}\}$ 和 $\{k_{r+1}, \ldots, k_j\}$ 的优化解, 我们就可以求出 K 的优化解.

如果 $r=i$, 左子树 $T_{ii-1}\{k_i, \ldots, k_{i-1}\}$ 仅包含 d_{i-1}

如果 $r=j$, 右子树 $T\{k_{j+1}, \ldots, k_j\}$ 仅包含 d_j

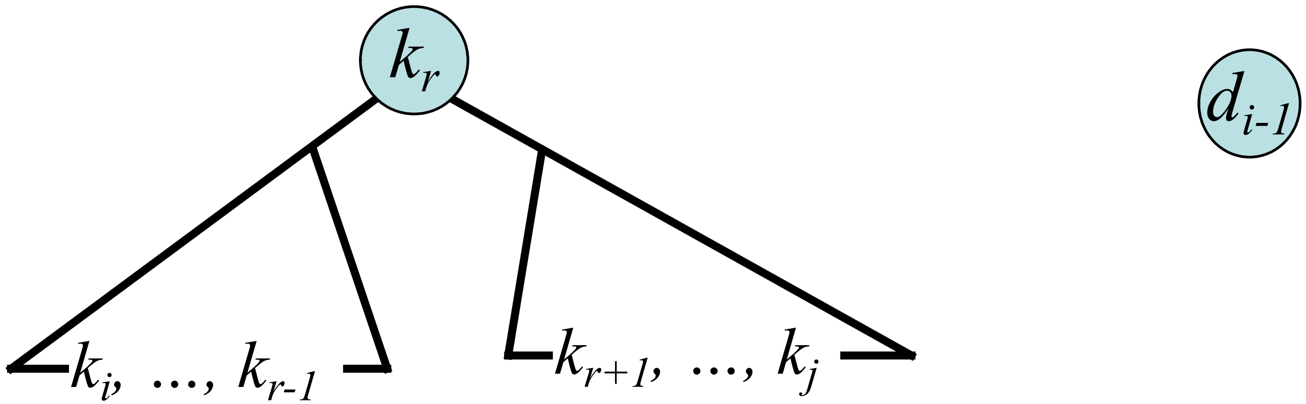
子问题重叠性


k_1, k_2, k_3, k_4
k_1
d_0
k_2, k_3, k_4
k_2
k_1
k_3,k_4
k_3
k_1,k_2
k_4
k_4
k_1, k_2, k_3
d_4
k_2
d_1
k_3,k_4
k_3
k_2
k_4
k_4
k_2, k_3
d_4
k_1
d_0
k_2, k_3
k_2
k_1
k_3
k_3
k_1,k_2
d_3

- 优化解的结构分析
- **建立优化解代价的递归方程**
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

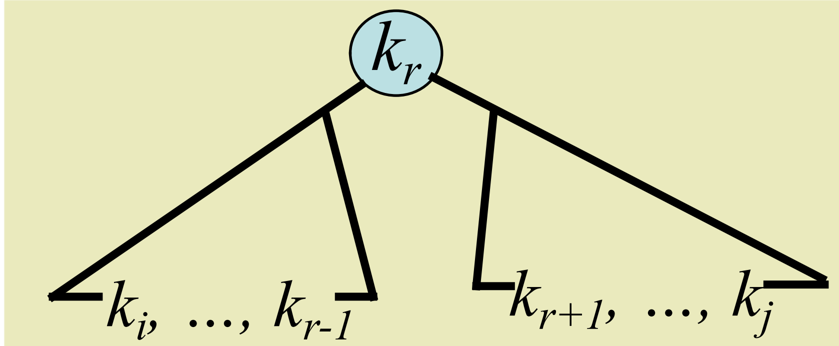
建立优化解的搜索代价递归方程

- 令$E(i, j)$为$\{k_i, \ldots, k_j\}$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，选择一个$k_r \in \{k_i, \ldots, k_j\}$:



当把左右优化子树放进T_{ij}时，每个结点的深度增加1

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



- 计算 $W(i, r-1)$ 和 $W(r+1, j)$

由 $E(LT+1)=\sum_{l=i}^{r-1}(DEP_{左}(k_l)+2)p_l+\sum_{l=i-1}^{r-1}(DEP_{左}(d_l)+2)q_l$

$E(LT)=\sum_{l=i}^{r-1}(DEP_{左}(k_l)+1)p_l+\sum_{l=i-1}^{r-1}(DEP_{左}(d_l)+1)q_l$

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

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

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

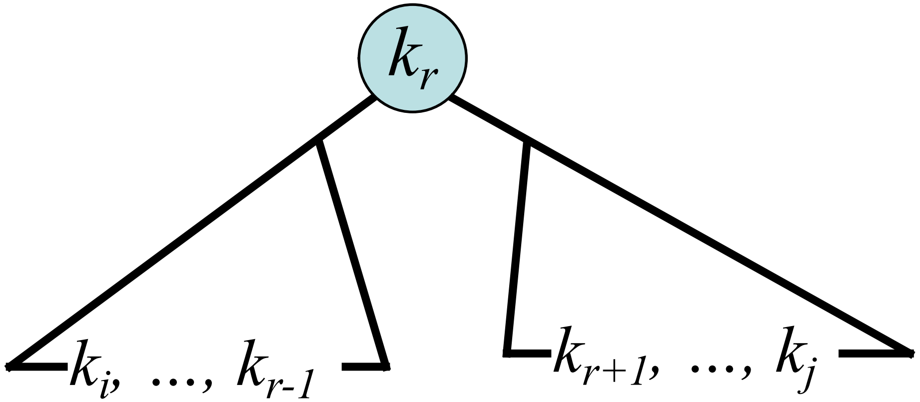
由 $W(i,r-1)=\sum_{l=i}^{r-1}p_l+\sum_{l=i-1}^{r-1}q_l$ ， $W(i, i-1)=q_{i-1}$

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

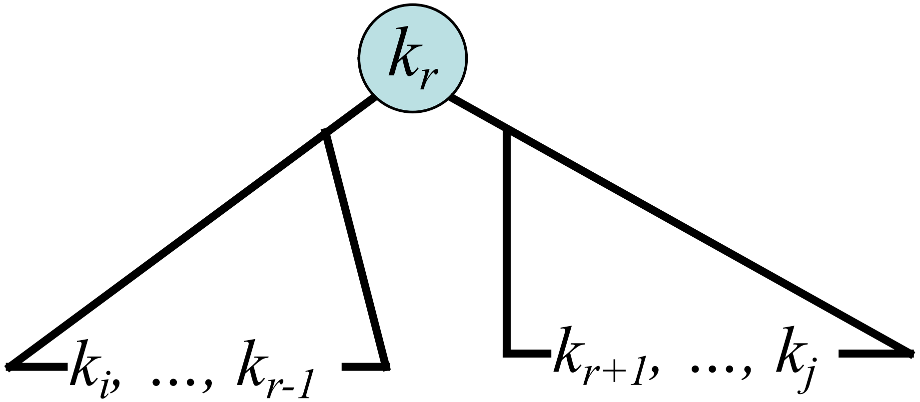
$W(i, j) = W(i, r\text{-}1) + W(r+1, j) + p_r = W(i, j\text{-}1) + p_j + q_j$

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

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



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



总之

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

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

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

$$E(i, j) = \min_{i \le r \le j} \{E(i, r\text{-}1) + E(r+1, j) + W(i, j)\} \quad \text{If } j \ge i$$

- 优化解的结构分析
- 建立优化解代价的递归方程
- **递归地划分子问题**
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

递归划分子问题

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

$$E(i, j)=\min_{i\le r\le j}\{E(i,r-1)+E(r+1,j)+W(i, j)\} \quad \text{If } j\ge i$$

$E(i,i-1)$	$E(i,i)$	……	$E(i,j-1)$	$E(i, j)$
				$E(i+1, j)$
				$E(i+2, j)$
				……
				$E(j+1, j)$

- 优化解的结构分析
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价
  记录优化解的构造信息
- 构造优化解

自下而上计算优化解的代价

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

$$E(i, j)=\min_{i\le r\le j}\{E(i,r-1)+E(r+1,j)+W(i, j)\} \quad \text{If } j\ge i$$

q_0	$E(1,0)$	$E(1,1)$	$E(1,2)$	$E(1,3)$	$E(1,4)$
	q_1	$E(2,1)$	$E(2,2)$	$E(2,3)$	$E(2,4)$
		q_2	$E(3,2)$	$E(3,3)$	$E(3,4)$
			q_3	$E(4,3)$	$E(4,4)$
				q_4	$E(5,4)$

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

$W(i, j\text{-}1)$	$W(i, j)$

q_0	$W(1,0)$	$W(1,1)$	$W(1,2)$	$W(1,3)$	$W(1,4)$
	q_1	$W(2,1)$	$W(2,2)$	$W(2,3)$	$W(2,4)$
		q_2	$W(3,2)$	$W(3,3)$	$W(3,4)$
			q_3	$W(4,3)$	$W(4,4)$
				q_4	$W(5,4)$

- 算法
  - 数据结构
    - $E[1{:}n+1;\ 0{:}n]$: 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 $\{k_i, \ldots, k_j\}$优化解的根

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

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

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

$E(1,0)$	$E(1,1)$	$E(1,2)$	$E(1,3)$	$E(1,4)$
	$E(2,1)$	$E(2,2)$	$E(2,3)$	$E(2,4)$
		$E(3,2)$	$E(3,3)$	$E(3,4)$
			$E(4,3)$	$E(4,4)$
				$E(5,4)$

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 $Root$

算法复杂性

- 时间复杂性
  - (l, i, r)三层循环，每层循环至多n步
  - 时间复杂性为$O(n^3)$
- 空间复杂性
  - 二个$(n+1)\times(n+1)$数组，一个$n\times n$数组
  - $O(n^2)$

- 优化解的结构分析
- 建立优化解代价的递归方程
- 递归地划分子问题
- 自底向上计算优化解的代价记录优化解的构造信息
- 构造优化解

Homework 2: 优化解的构造算法