计算机算法基础

(2014.9)

何琨 brooklet60@gmail.com

华中科技大学计算机学院

第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- **6.3 每对结点之间的最短路径**
- **6.4 最优二分检索树**
- 6.5 0/1背包问题

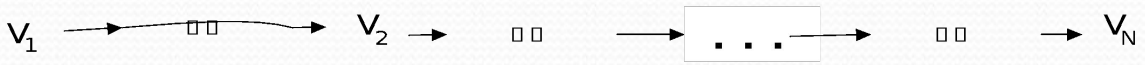
第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- 6.3 每对结点之间的最短路径
- 6.4 最优二分检索树
- 6.5 0/1背包问题

6.1 动态规划的一般方法

动态规划(dynamic programming)是运筹学的一个分支，是求解决策过程(decision process)最优化的数学方法。

1. 多阶段决策过程



阶段：事件的发展过程分为若干个相互联系的阶段。事件的发展总是从初始阶段开始，依次经过第一阶段、第二阶段、第三阶段、...，直至最后一个阶段结束。过程不同，阶段数就可能不同。

阶段变量：描述阶段的变量称为阶段变量。

状态:状态表示每个阶段所面临的自然状况或客观条件。

既是当前阶段某途径的起点，也是前面阶段某途径的终点。

状态变量:过程的状态通常可以用一个或一组数来描述，称为状态变量。

状态变量可以是多维的，用向量表示;实际中在每个阶段的状态维数甚至可以不同。

状态集合:当过程按所有可能不同的方式发展时，过程各段的状态变量将在某一确定的范围内取值，状态变量取值的集合称为状态集合。

假设事件在初始状态后需要经过n个这样的阶段。一般情况下，从i阶段发展到i+1阶段（0≤i＜n）可能有多种不同的途径，而事件必须从中选择一条途径往前进展。使过程从一个状态演变到下一状态。

决策：在一个阶段的状态给定以后，从该状态演变到下一阶段某个状态的一种选择称为决策。也就是在两个阶段间选择发展途径的行为。

决策变量：描述决策的变量称决策变量，用一个数或一组数表示。不同的决策对应着不同的数值。

决策序列：事件的发展过程之中需要经历n个阶段，需要做n次“决策”，这些“决策”就构成了事件整个发展过程的一个决策序列。

多阶段决策过程：具备上述性质的过程称为多阶段决策过程(multistep decision process)，求解多阶段决策过程问题就是求取事件发展的决策序列。

无后效性：对任意的阶段i，**阶段i以后的行为仅依赖于i阶段的状态，而与i阶段之前过程如何达到这种状态的方式无关**，这种性质称为无后效性。

状态的这个性质意味着过程的历史只能通过当前的状态去影响它的未来的发展，而不能直接作用于未来的发展。

如果某一阶段的状态已确定，则在这一阶段以后，过程的发展就不再受这阶段以前各段状态的直接影响，所有各阶段都确定时，整个过程也就确定了。

因状态满足无后效性，故在每个阶段选择决策时只需考虑当前的状态而无须考虑过程的历史。

状态转移方程:阶段之间状态变量值的变化存在一定的关系，如果给定i阶段的状态变量x(i)的值后，第i+1阶段的状态变量x(i+1)就可以完全确定，即**x(i+1)的值随x(i)和第i阶段的决策u(i)的值变化而变化**，可以把这一关系看成(x(i)，u(i))与x(i+1)确定的对应关系，用

$$x(i+1)=T_i(x(i),u(i))$$

表示。

这种从i阶段到i+1阶段的状态转移规律称为状态转移方程。

最优化问题：

每一决策都附有一定的“成本”，决策序列的成本是序列中所有决策的成本之和。

设从阶段i到阶段i+1有p_i种不同的选择，则从阶段1至阶段n共有$p_1p_2\cdots\cdots p_n$种不同的途径，每个途径对应一个决策序列。

问：这些途径里面，哪一个的成本最小？

——如何求取最优决策序列？

可行解：从问题的开始阶段到最后阶段每一个合理的决策序列都是问题的一个可行解。

目标函数：用来衡量可行解优劣的标准，通常以函数形式给出。

最优解：能够使目标函数取极值的可行解。

多阶段决策过程的最优化问题就是：求能够获得问题最优解的决策序列——最优决策序列。

2. 多阶段决策过程的求解策略

问题的决策序列表示为：$(x_1, x_2, \ldots, x_n)$

其中x_i表示第i阶段的决策：

$$S_0 \xrightarrow{x_1} S_1 \xrightarrow{x_2} S_2 \xrightarrow{x_3 \ldots x_n} \ldots\ldots \rightarrow S_n$$

1）枚举法

穷举可能的决策序列，从中选取可以获得最优解的决策序列：

若问题的决策序列由n次决策构成，每一阶段分别有p_1、p_2、…、p_n选择，则可能的决策序列将有$p_1p_2\ldots p_n$个。

2）动态规划

20世纪50年代初美国数学家R.E.Bellman等人在研究多阶段决策过程的优化问题时，提出了著名的**最优化原理**(principle of optimality)，从而把多阶段过程转化为一系列单阶段问题，创立了解决这类过程优化问题的新方法——动态规划。

动态规划问世以来，在经济管理、生产调度、工程技术和最优控制等方面得到了广泛的应用。例如最短路线、库存管理、资源分配、设备更新、排序、装载等问题，用动态规划方法比用其它方法求解更为方便。

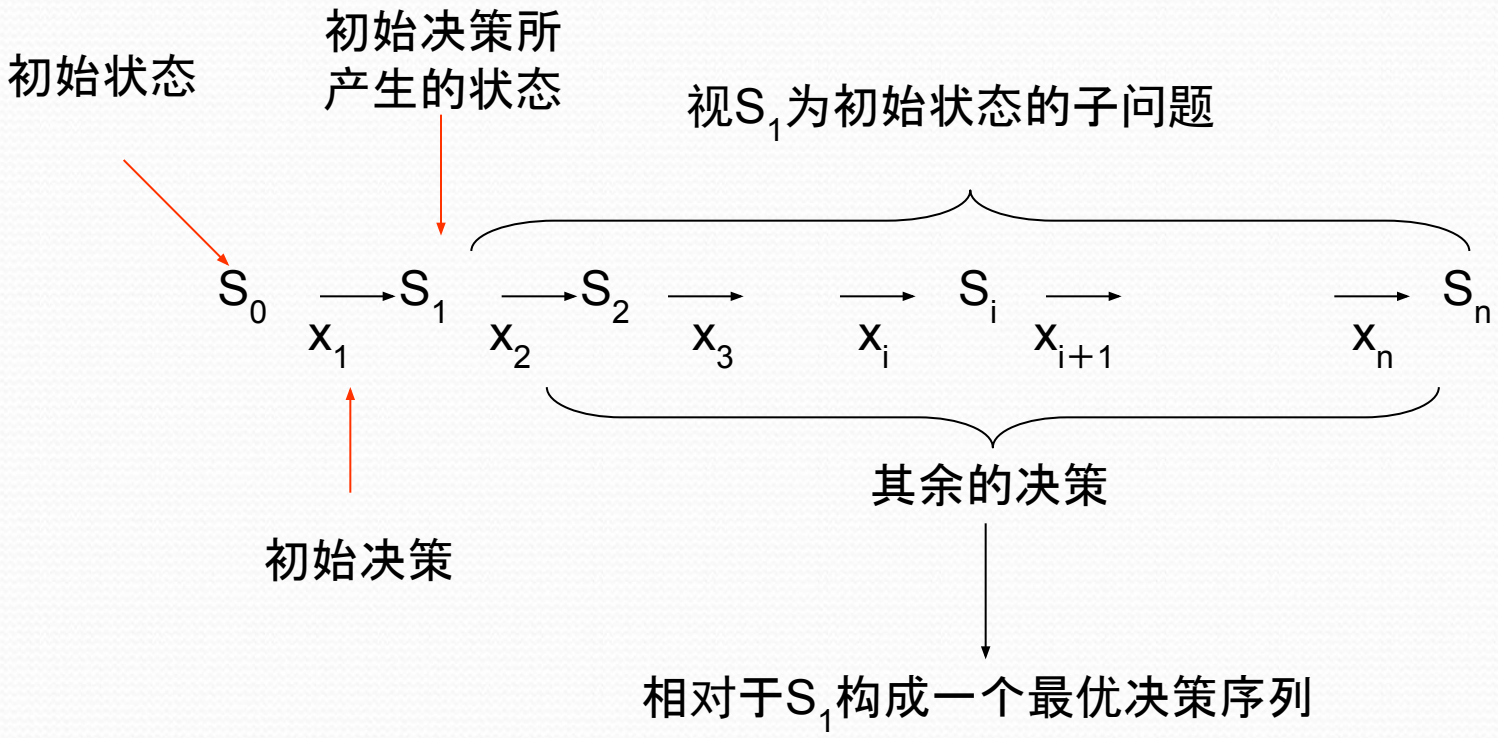
注意： 适用动态规划的问题必须满足：

1）最优化原理

2）无后效性

3. 最优性原理(Principle of Optimality)

过程的最优决策序列具有如下性质：无论过程的初始状态和初始决策是什么，其余的决策都必须相对于初始决策所产生的状态构成一个最优决策序列。



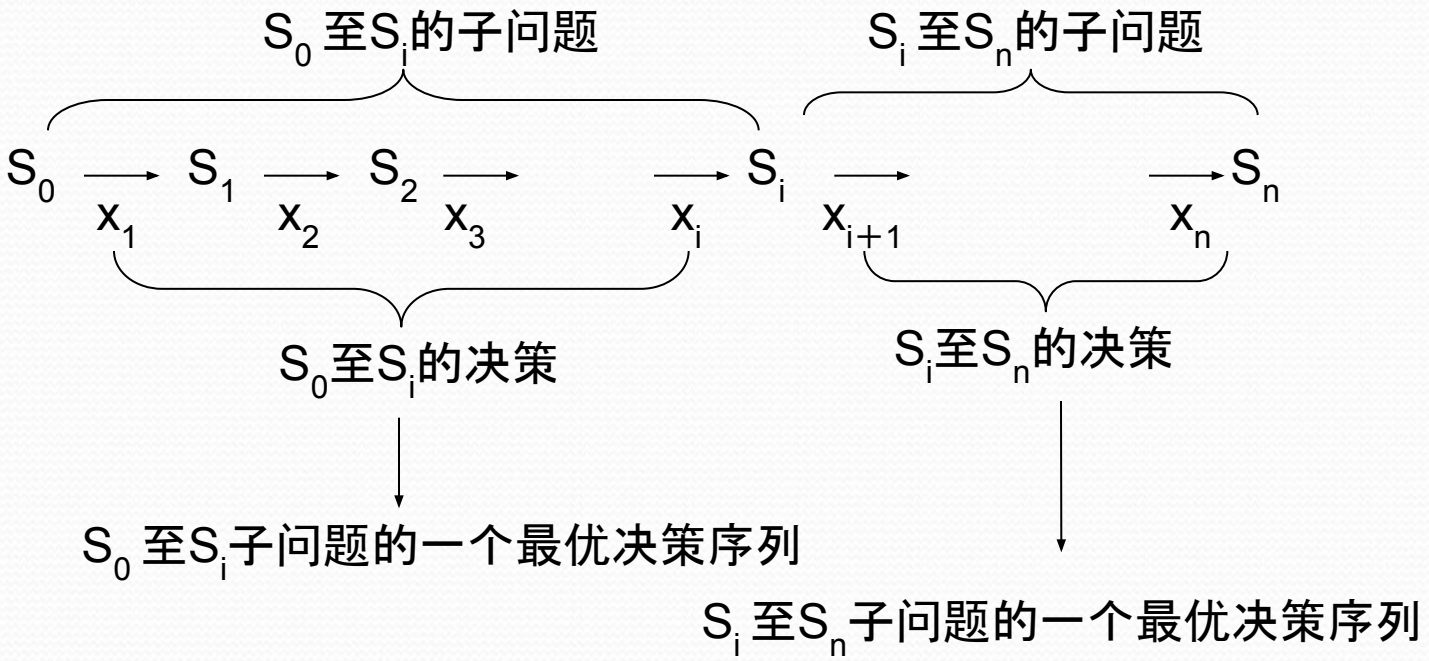
对最优性原理的另外陈述：

一个最优策略具有这样的性质，不论过去状态和决策如何，对前面的决策所形成的状态而言，余下的决策必须构成最优(子)策略。

简而言之，**一个最优策略的子策略应是最优的**。

一个问题满足最优性原理又称其具有“**最优子结构性质**”。

延伸：若全局是最优的，则局部亦是最优的



特征：如果整个序列是最优决策序列，则该序列中的任何一段子序列将是相对于该子序列所对应子问题的最优决策子序列（最优子结构）。

例：最短路径

若$v_1v_2v_3……v_n$是从节点v_1到节点v_n的最短路径。则：

- $v_2v_3……v_n$是从v_2到v_n的最短子路径；
- $v_3……v_n$是从v_3到v_n的最短子路径；

……

推广：

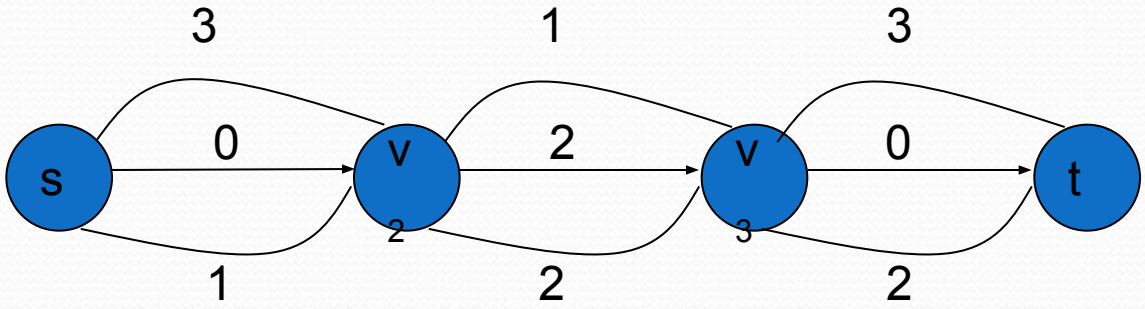
对$v_1v_2v_3……v_n$中的任意一段子路径：

$$v_pv_{p+1}……v_q(p\le q,1\le p,q\le n),$$

将代表从v_p至v_q的最短子路径。

反例——模4最优路径问题：

如下图，求由s至t的一条路径，使得该路径的长度模4的余数(即Length(s,t)mod 4)最小。



此时，问题的一个最优决策序列是：

$s\text{-}3\text{->}v_2\text{-}2\text{->}v_3\text{-}3\text{->}t$

但最优性原理不一定成立：最优决策序列上的任一子决策序列相对于当前子问题不是最优的。

利用动态规划求解问题的方法：

第一步：证明问题满足最优性原理

如果证明了所求解问题满足最优性原理，则说明用动态规划方法有可能解决该问题。

所谓“问题满足最优性原理”即：问题的最优决策序列具有最优性原理所阐述的性质：最优子结构性。

第二步：获得问题状态的递推关系式（即状态转移方程）

能否求得各阶段间状态变换的递推关系式是解决问题的关键。

$f(x_1,x_2,\ldots,x_i) \rightarrow x_{i+1}$　　向后递推

或　$f(x_i,x_{i+1},\ldots,x_n) \rightarrow x_{i-1}$　　向前递推

例6.1 [多段图问题]

多段图G=(V,E)是一个有向图，且具有特性：

结点：结点集V被分成k≥2个不相交的子集合V_i，1≤i≤k，其中V_1和V_k分别只有一个结点：s(源结点)和t(汇点)。

段：每一子集合V_i定义图中的一段——共k段。

边：所有的边(u,v)均具有如下性质：设<u,v>∈E，若u∈V_i，则v∈V_{i+1}，即边连接在相邻的两段之间，从i段的某个结点u指向i+1段的某个结点v，1≤i≤k－1。

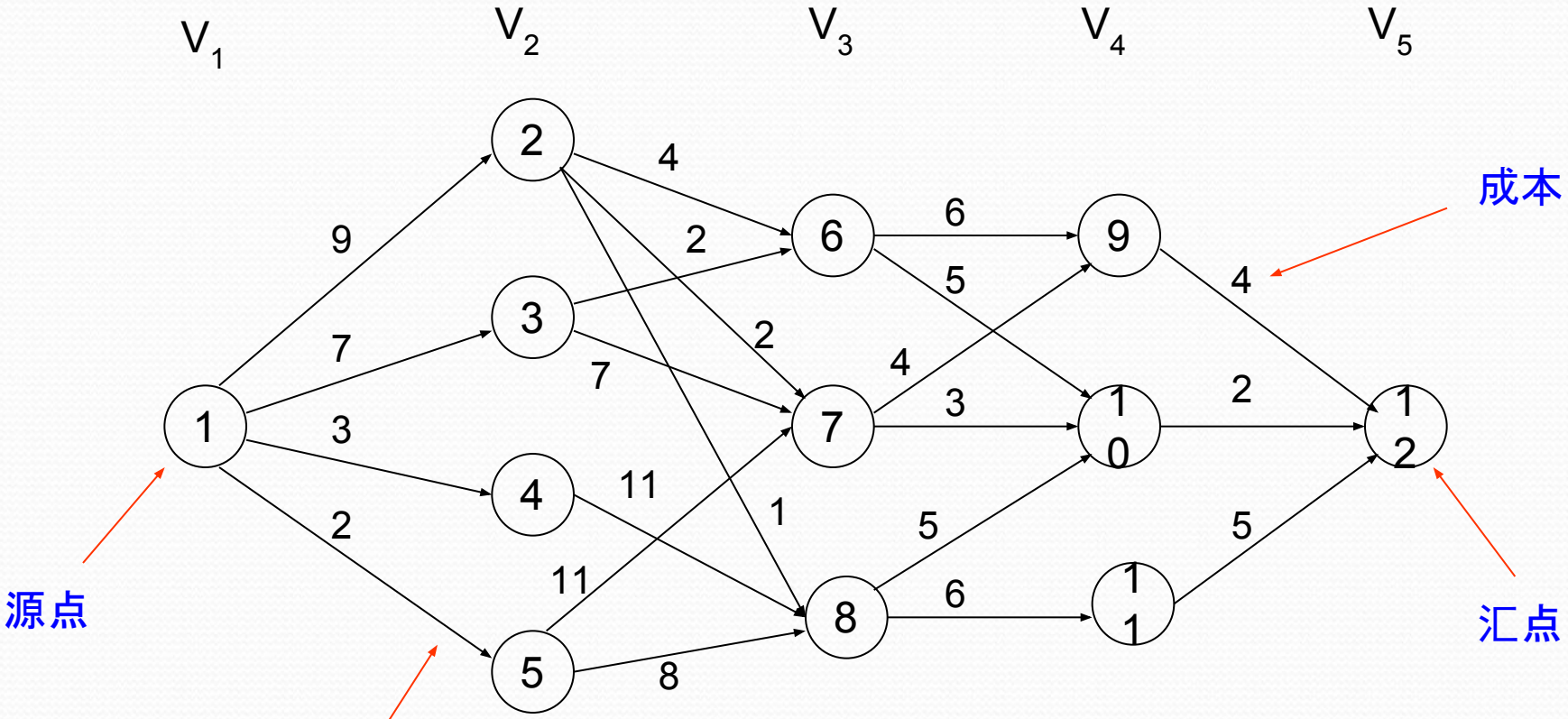
成本：每条边<u,v>均附有成本，记为c(u,v)。

s到t的路径：是一条从第1段的源点s出发，依次经过第2段的某结点$v_{2,i}$，第3段的某结点$v_{3,j}$、…、最后在第k段的汇点t结束的路径。

该路径的成本是这条路径上边的成本之和。

多段图问题：求由s到t的最小成本路径。

段是不相交的结点子集



边连接在相邻的两段之间

5段图

多段图问题的求解：

将生成从s到t的最小成本路径看成是在k-2个阶段（除s和t外）进行某种决策的多阶段决策过程：

从s开始，第i次决策决定V_{i+1}($1 \le i \le k-2$)中的哪个结点在从s到t的最短路径上。

问题：最优性原理对多段图问题成立吗？

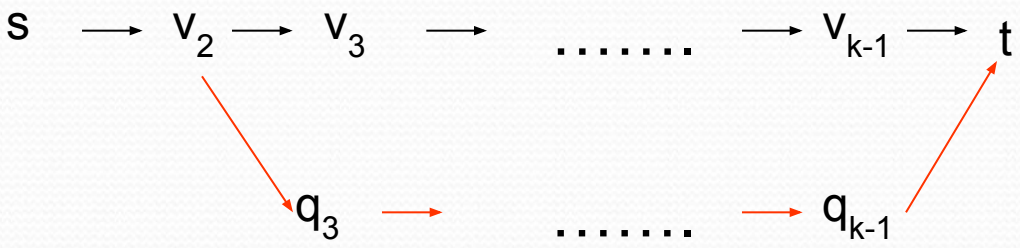
假设$s,v_2,v_3,\ldots,v_{k-1},t$是一条由s到t的最短路径。

- 初始状态：s
- 初始决策：(s,v_2)，$v_2 \in V_2$
- 初始决策产生的状态：v_2

则，其余的决策：$v_3,\ldots,v_{k-1}$相对于v_2必须是一条从v_2到t的最小成本子路径——**最优性原理成立**。

反证:

若不然，设$v_2,q_3,\ldots,q_{k-1},t$是一条由v_2到t的更短的路径，则$s,v_2,q_3,\ldots,q_{k-1},t$将是比$s,v_2,v_3,\ldots,v_{k-1},t$更短的从s到t的路径。与假设矛盾。故，最优性原理成立



例6.2[0/1背包问题] KNAP(1,j,X)

目标函数：

$$\sum_{1\le i\le j} p_i x_i$$

约束条件：

$$\sum_{1\le i\le j} w_i x_i \le X$$

$$x_i = 0或1, p_i > 0, w_i > 0, 1 \le i \le j$$

0/1背包问题：KNAP(1,n,M)

证明最优性原理对0/1背包问题成立：

设$y_1,y_2,\ldots,y_n$是KNAP(1,n,M)的最优序列，它是$x_1,x_2,\ldots,x_n$的一个0/1值实例。

1) 初始状态：KNAP(1,n,M)

2) 初始决策：决定y_1等于1还是等于0，两种情况分开讨论：

★ 若$y_1=0$，则KNAP(2,n,M)是初始决策产生的状态。此时，$y_2,\ldots,y_n$必将相对于KNAP(2,n,M)将构成一个最优决策序列。否则，$y_1,y_2,\ldots,y_n$也就不是KNAP(1,n,M)的最优解了。

★若$y_1=1$, 则KNAP(2,n,M-w_1)是初始决策产生的状态。此时，$y_2,\ldots,y_n$相对于KNAP(2,n,M－w_1)也将构成一个最优决策序列。

如若不然，设存在另一0/1序列$z_2,z_3,\ldots,z_n$,使得

且 $\sum_{2\le i\le n} p_i z_i \ge \sum_{2\le i\le n} p_i y_i \qquad \sum_{2\le i\le n} w_i z_i \le M - w_1$

则序列$y_1,z_2,\ldots,z_n$将是KNAP(1,n,M)的一个有更大效益值得序列。与假设矛盾。

故，最优性原理成立。

4.最优决策序列的表示

设
- S_0是问题的初始状态
- 问题需要做n次决策
- i阶段的决策值记为x_i：$1 \le i \le n$。

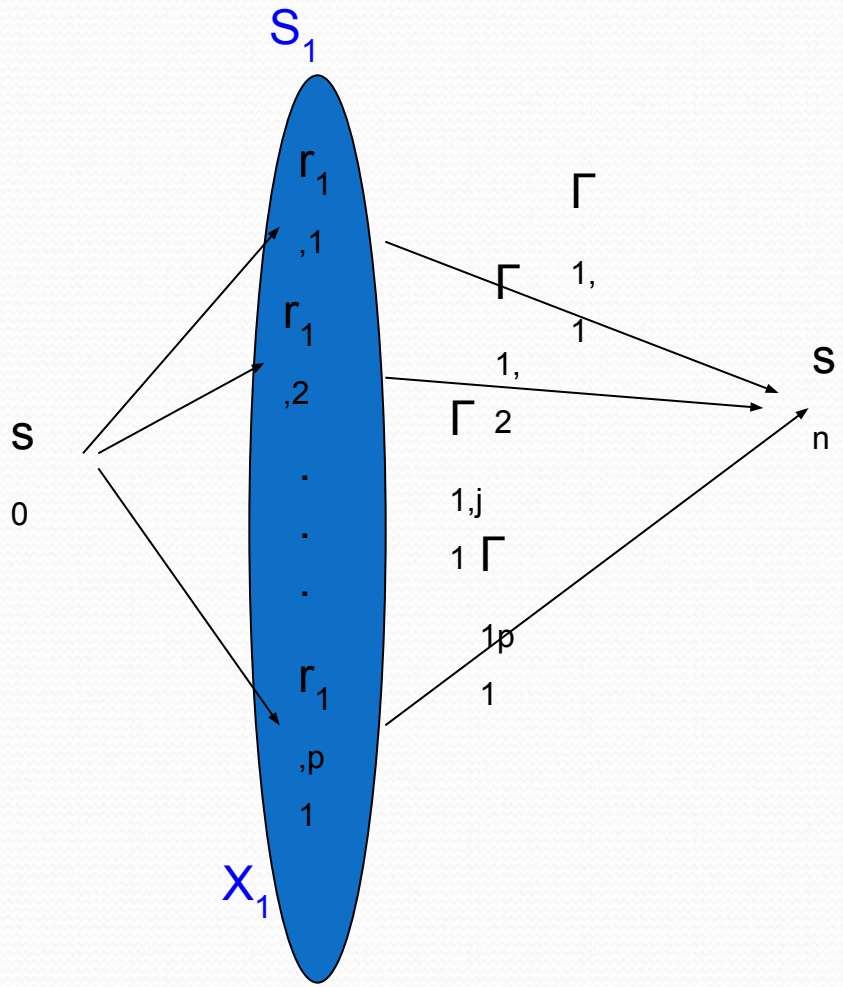
设 1）$X_1=\{r_{1,1},r_{1,2},\ldots,r_{1,p_1}\}$是$x_1$可选决策值的集合。

2）S_{1,j_1}是在选择决策值r_{1,j_1}之后所产生的状态——“初始决策”所产生的状态。

3）Γ_{1,j_1}是相应于状态S_{1,j_1}的最优决策序列。

则，相应于S_0的最优决策序列就是$\{r_{1,j_1}\Gamma_{1,j_1} | 1 \le j_1 \le p_1\}$中最优的序列，记为

$$\underset{1 \le j_1 \le p_1}{OPT}\{r_{1,j_1}\Gamma_{1,j_1}\} = r_1\Gamma_1$$


S1
r1
,1
r1
,2
.
.
.
r1
,p
1
X1
Γ
1,
1
Γ
1,
Γ 2
1,j
1 Γ
1p
1
s
0
s
n

若已经做了k-1次决策，1≤k-1＜n，设$x_1,x_2,\ldots,x_{k-1}$的最优决策值是$r_1,r_2,\ldots,r_{k-1}$，所产生的状态依次为$S_1,S_2,\ldots,S_{k-1}$。

x_k将是基于S_{k-1}的决策。

设 1）$X_k=\{r_{k,1},r_{k,2},\ldots,r_{k,pk}\}$是$x_k$可能的决策值的集合。

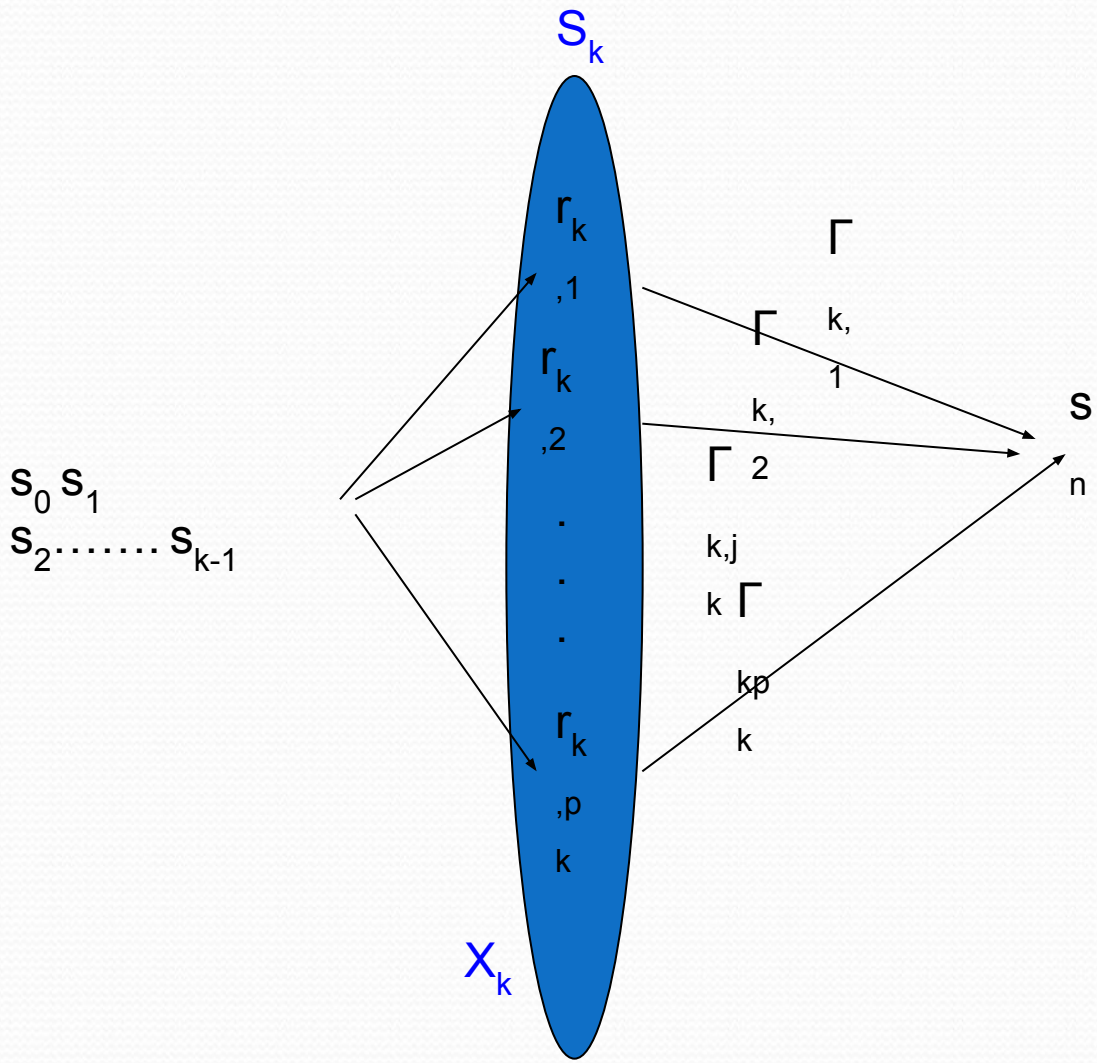
2）$S_{k,jk}$是在选择决策值$r_{k,jk}$之后所产生的状态，$1\le j_k\le p_k$。

3）$\Gamma_{k,jk}$是相应于状态$S_{k,jk}$的最优决策子序列。

则，相应于S_{k-1}的最优决策序列是

$$\underset{1\le j_k\le pk}{OPT}\{r_{k,j_k}\Gamma_{k,j_k}\}=r_k\Gamma_k$$

相应于S_0的最优决策序列为$r_1,\ldots,r_{k-1}$，r_k，Γ_k

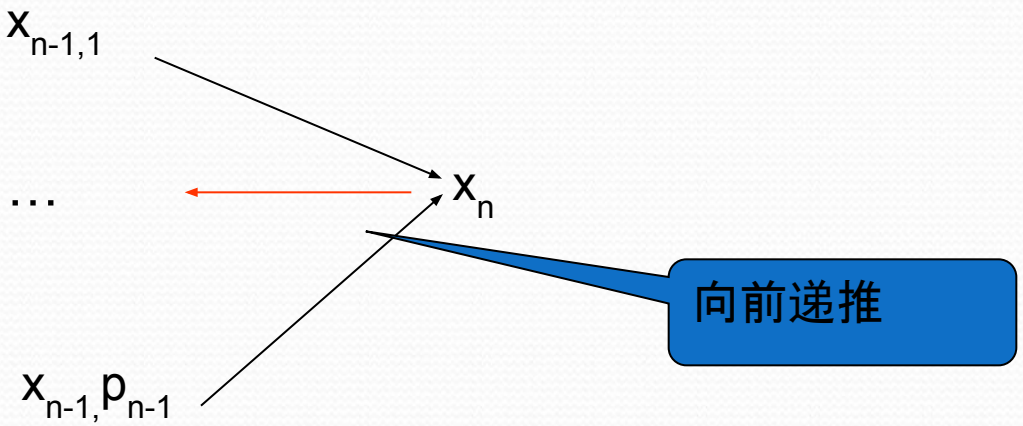

S_k
$r_{k,1}$
$r_{k,2}$
r_{k,p_k}
X_k
$\Gamma_{k,1}$
$\Gamma_{k,2}$
$\Gamma_{k,j}$
Γ_{kp_k}
s
n
$s_0 s_1 s_2 \ldots\ldots s_{k-1}$

5. 递推策略

1) **向前处理法**：从最后一个阶段开始，逐步向前递推求出各阶段的决策值。

即，先给出x_n的决策，然后根据x_n求x_{n-1}，再根据x_n，x_{n-1}求x_{n-2}，…。

向前递推关系式：根据$x_{i+1},\ldots,x_n$求x_i的关系式。



例6.3 利用向前处理法求解k段图问题

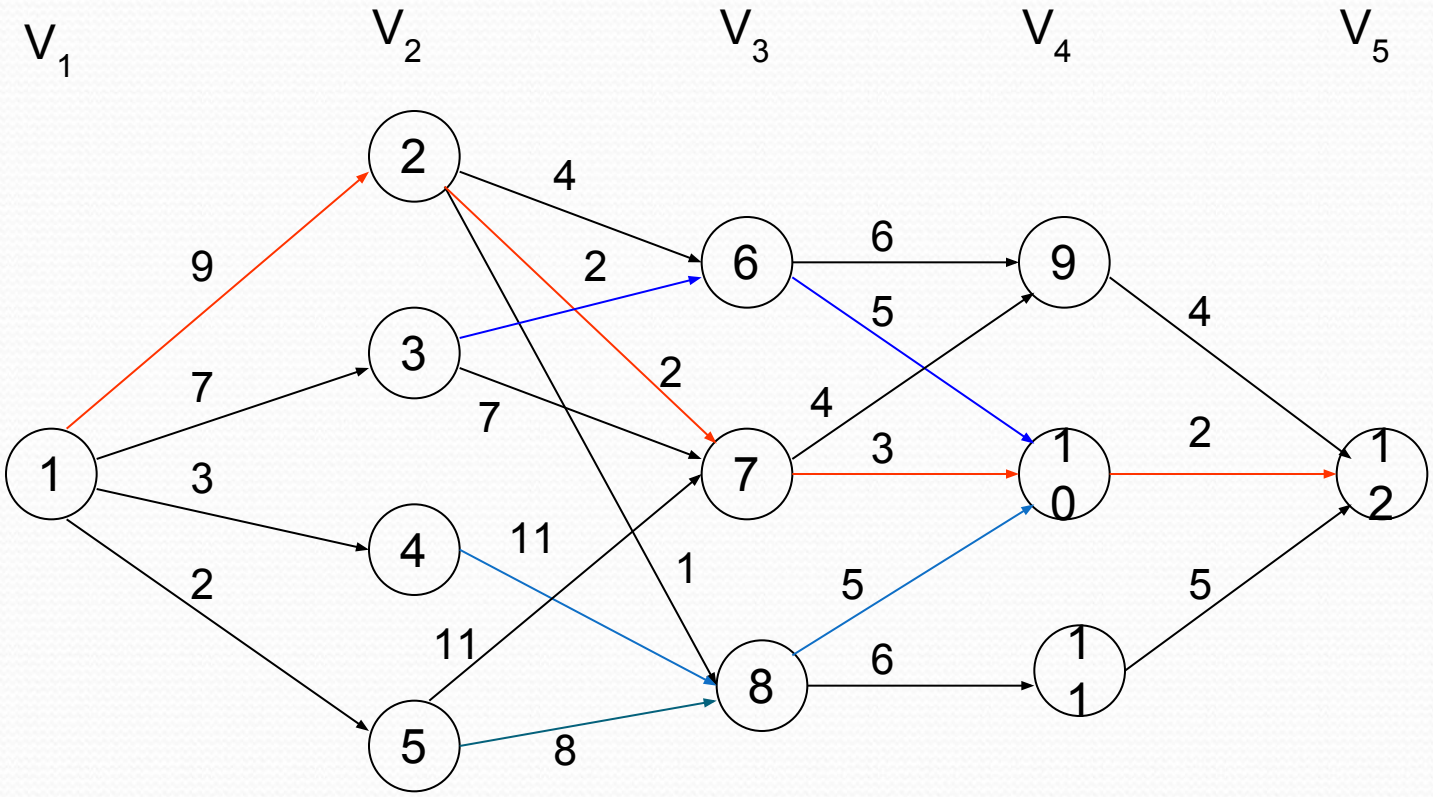
设 $v_{2,j_2} \in V_2$, $1 \le j_2 \le p_2$, $|V_2| = p_2$;

是 $\Gamma_{v_{2,j_2}} v_{2,j_2}$ 到t的最短路径，则s到t的最短路径是

$\{s \mid \Gamma_{v_{2,j_2}} v_{2,j_2} \in V_2, 1 \le j_2 \le p_2\}$ 中最短的那条路径。

则，从 V_i 中的结点 j_i 到t的最短路径将是：

$$\min\{ |\langle v_{i,j_i}, \Gamma_{v_{i+1,j_{i+1}}} \rangle \in E_{v_{i,j_i} v_{i+1,j_{i+1}} \in V_{i+1}}, \quad v_{i+1,j_{i+1}} \quad 1 \le j_{i+1} \le p_{i+1}\}$$



5段图

例6.4 利用向前处理法求解0/1背包问题

设 $g_i(x)$是KNAP(i+1,n,X)的最优解。

- $g_0(M)$：KNAP(1,n,M)的最优解。
- 由于x_1的取值等于1或0，可得：

  $g_0(M)=\max\{g_1(M),g_1(M-w1)+p_1\}$
- 对于某个x_{i+1}，x_{i+1}等于1或0，则有：

  $g_i(X)=\max\{g_{i+1}(X),g_{i+1}(X-w_{i+1})+p_{i+1}\}$

初始值：

$$g_n(X)=\begin{cases}0 & X\geq 0\\ -\infty & X<0\end{cases}$$

最优解是$g_0(M)$

2）向后处理法

列出根据$x_1,\ldots,x_{i-1}$的决策值求取x_i的关系式。从第一个阶段，逐步向后递推求出各阶段的决策值。

$$x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$$

例6.5 k段图问题(向后处理策略)

设 $v_{k-1,j_{k-1}} \in V_{k-1}$, $1 \le j_{k-1} \le p_{k-1}$, $|V_{k-1}| = p_{k-1}$;

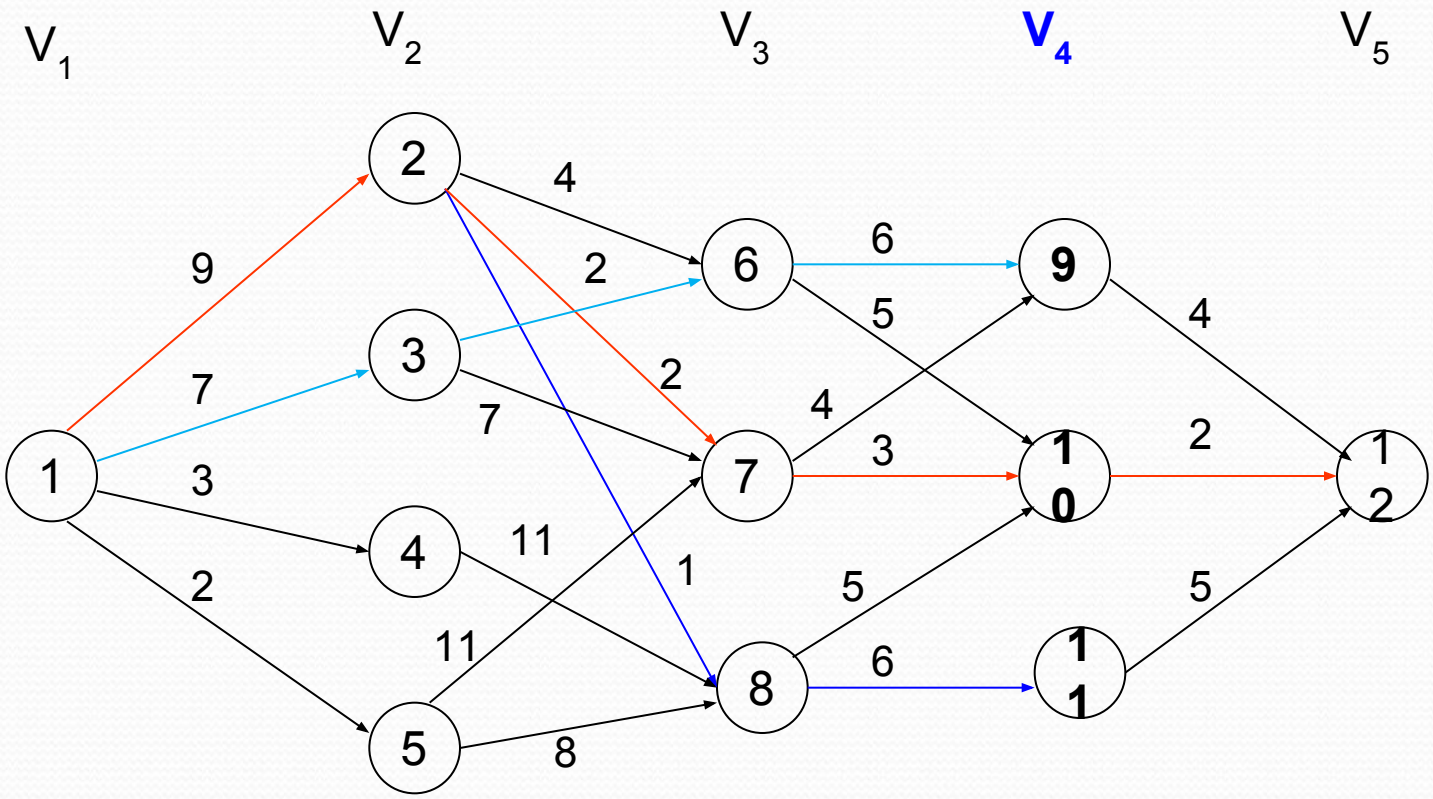
$\Gamma_{v_{k-1,j_{k-1}}}$是由s到 $v_{k-1,j_{k-1}}$ 的最短路径, 则s到t的最短路径是

$\{ t \mid \Gamma_{v_{k-1,j_{k-1}}} \in V_{k-1}, 1 \le j_{k-1} \le p_{k-1} \}$

中最短的那条路径。

从s到V_i中的结点j_i的最短路径将是:

$\min\{ \Gamma_{v_{i-1,j_{i-1}}} \mid < v_{i-1,j_{i-1}}, v_{i,j_i} > \in E, v_{i,j_i} \in V_{i+1}, 1 \le j_{i+1} \le p_{i+1} \}$



5段图

例6.6 0/1背包问题(向后处理策略)

设 $f_i(x)$是KNAP(1,i,X)的最优解。

则, $f_n(M)$ = KNAP(1,n,M)

- 由于x_n的取值等于1或0,可得:

  $f_n(M)=\max\{f_{n-1}(M),f_{n-1}(M-w_n)+p_n\}$

- 对于某个x_i, x_i等于1或0, 则有向后递推关系式:

  $f_i(X)=\max\{f_{i-1}(X),f_{i-1}(X-w_i)+p_i\}$

初始值:

$$f_0(X)=\begin{cases}0 & X\geq 0\\ -\infty & X<0\end{cases}$$

关于动态规划求解策略的进一步说明

1）动态规划是求解最优化问题的一种策略、途径和方法，而不是一种特殊算法。

因此，动态规划求解不象搜索或数值计算，具有一个标准的数学表达式和明确清晰的解题方法。动态规划针对的最优化问题，由于各种问题的性质不同，确定最优解的条件也互不相同，因而**动态规划的设计方法对不同的问题，有各具特色的解题方法**。

实际应用中，在对基本概念和方法正确理解的基础上，必须具体问题具体分析，以丰富的想象力去建立模型，用创造性的技巧去求解。

2. 对动态规划带来的改进的理解

若问题的决策序列由n次决策构成，而每次决策有p种选择，若采用枚举法，则可能的决策序列将有p^n个。而利用动态规划策略的求解过程中仅保存了所有子问题的最优解，而舍去了所有不能导致问题最优解的次优决策序列，可能有多项式的计算复杂度。

3.子问题的重叠性。

动态规划将原来具有指数级复杂度的搜索算法改进成了具有多项式时间的算法，其中的**关键在于解决冗余**，这是动态规划算法的根本目的。动态规划实质上是一种以空间换时间的技术，它在实现的过程中，**不得不存储过程中产生的各种状态，所以它的空间复杂度要大于其它的算法**。

补充

- 矩阵链的乘法
- 通过矩阵链的乘法理解动态规划的思想

第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- 6.3 每对结点之间的最短路径
- 6.4 最优二分检索树
- 6.5 0/1背包问题

6.2 多段图问题

1. 问题的描述

- 在多段图中求从s到t的一条最小成本的路径，可以看作是在k-2个段作出某种决策的结果。
- 第i次决策决定V_{i+1}中的哪个结点在这条路径上，这里$1 \le i \le k-2$；
- 最优性原理对多段图问题成立

2.用向前处理策略求解多段图问题

设 P(i,j)是一条从V_i中的结点j到汇点t的最小成本路径，

COST(i,j)是这条路径的成本。

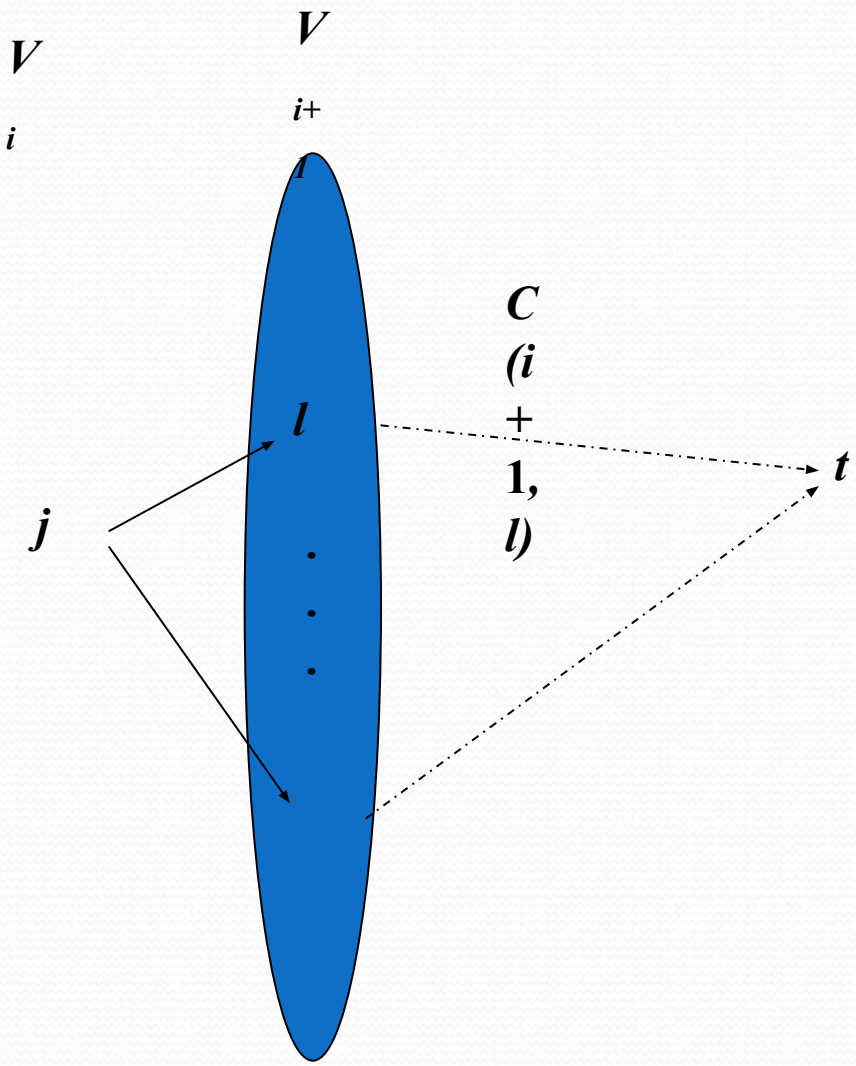
1) 向前递推式

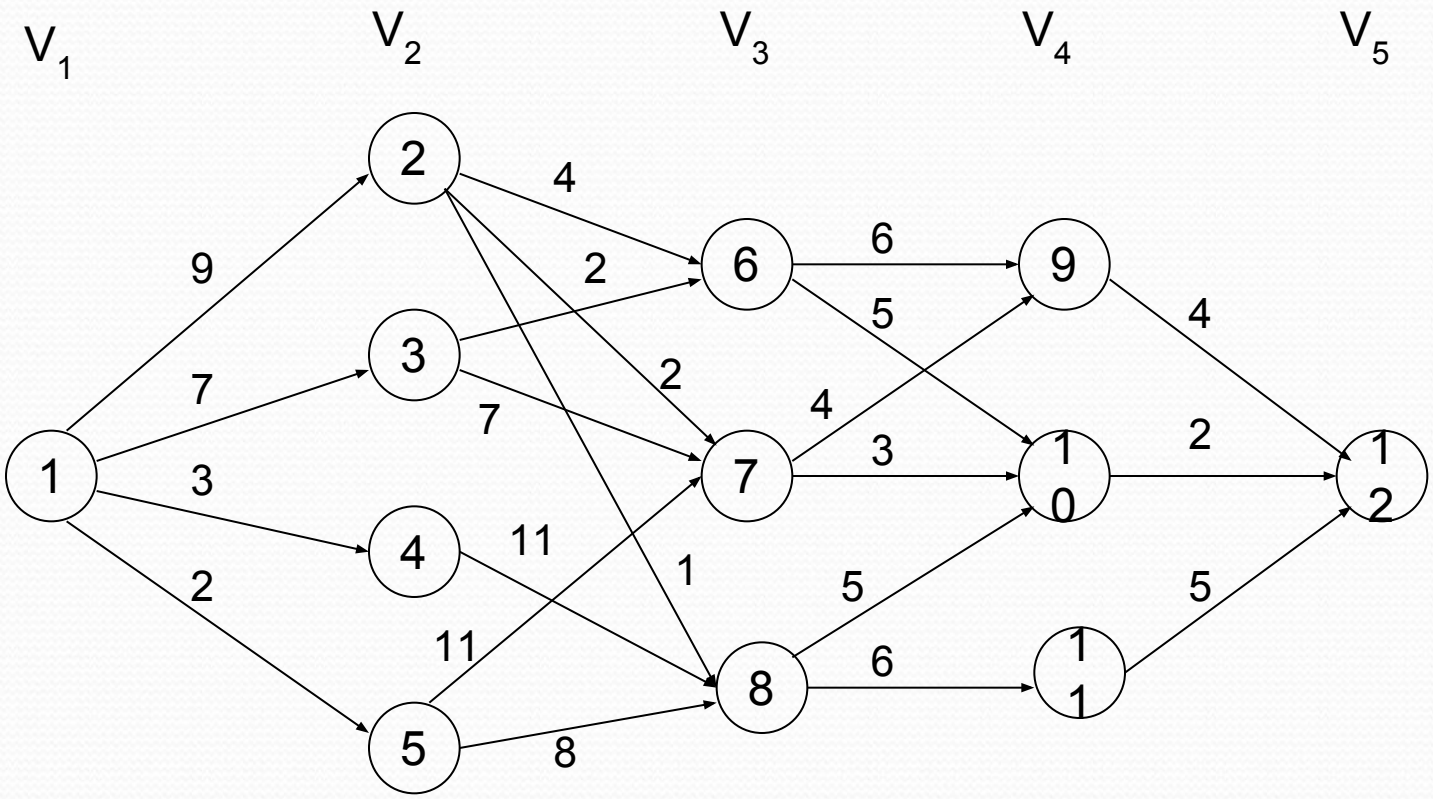
$$COST(i,j)=\min_{\substack{l\in v_{i+1}\\ <j,l>\in E}}\{c(j,l)+COST(i+1,l)\}$$

2) 递推过程

★ 第k-1段

$$COST(k-1,j)=\begin{cases} c(j,t) & <j,t>\in E \\ \infty & <j,t>\notin E \end{cases}$$


V
i
V
i+
1
C
(i
+
1,
l)
l
j
t


V1
V2
V3
V4
V5
1
2
3
4
5
6
7
8
9
10
11
12
9
7
3
2
4
2
7
11
2
1
11
8
6
5
4
3
6
5
4
2
5
6
5

5段图

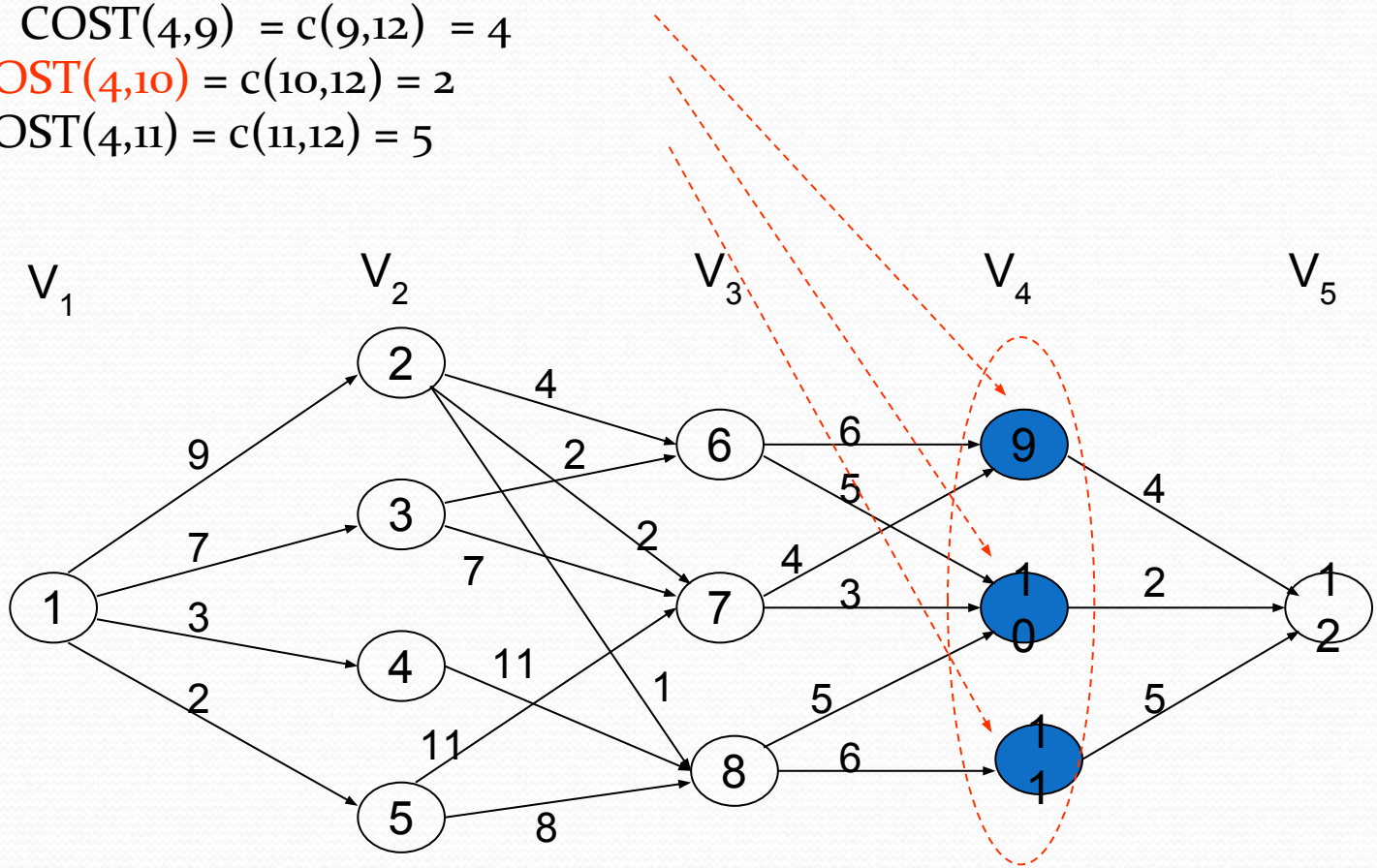
$$COST(i,j)=\min_{\substack{l\in v_{i+1}\\ <j,l>\in E}}\{c(j,l)+COST(i+1,l)\}$$

★各递推结果

第4段 COST(4,9) = c(9,12) = 4

COST(4,10) = c(10,12) = 2

COST(4,11) = c(11,12) = 5



5段图

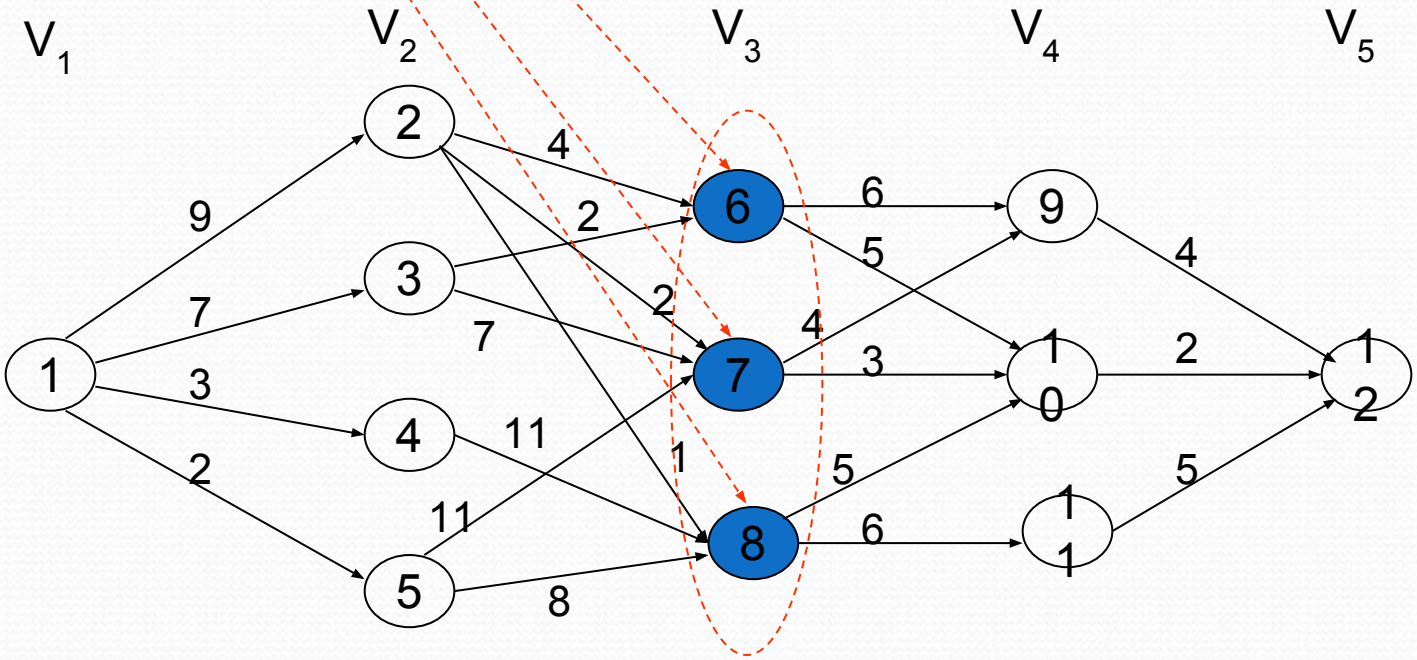
$$COST(i,j) = \min_{\substack{l \in v_{i+1} \\ <j,l> \in E}} \{c(j,l) + COST(i+1,l)\}$$

★各递推结果

第3段 COST(3,6) = min{6+COST(4,9),5+COST(4,10)} = 7

COST(3,7) = min{4+COST(4,9),3+COST(4,10)} = 5

COST(3,8) = min{5+COST(4,10),6+COST(4,11)} = 7



5段图

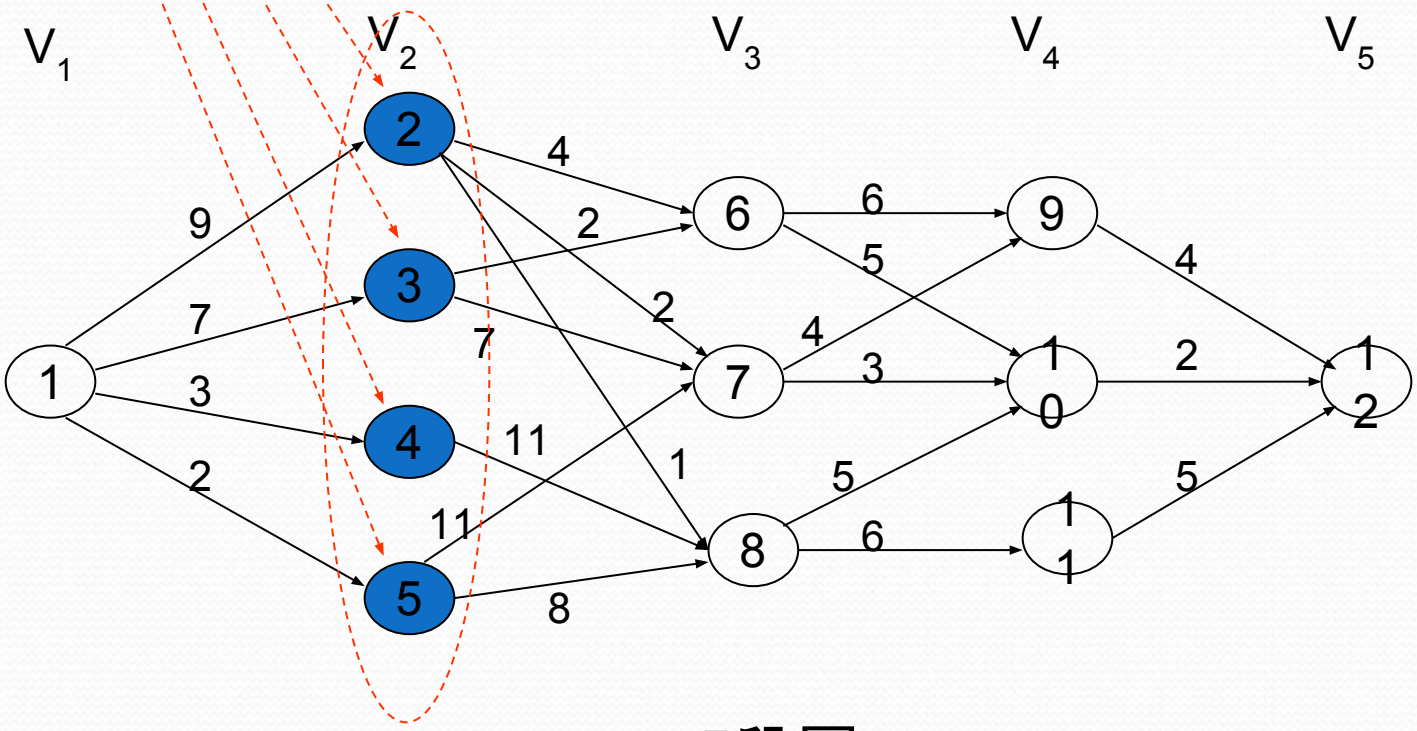
$$COST(i,j) = \min_{\substack{l \in v_{i+1} \\ <j,l> \in E}} \{c(j,l) + COST(i+1,l)\}$$

第2段 COST(2,2) = min{4+COST(3,6) , 2+COST(3,7), 1+COST(3,8)} = 7

COST(2,3) = 9

COST(2,4) = 18

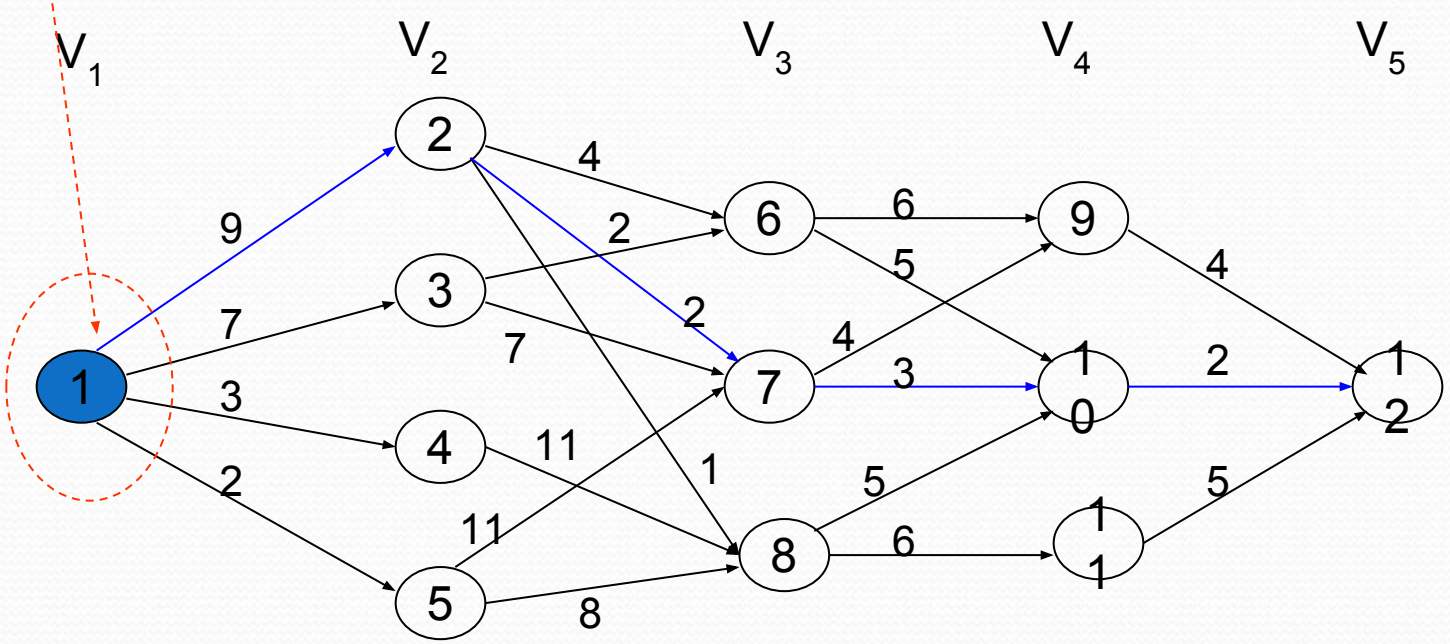
COST(2,5) = 15



5段图

$$COST(i,j)=\min_{\substack{l\in V_{i+1}\\ <j,l>\in E}}\{c(j,l)+COST(i+1,l)\}$$

第1段 COST(1,1) = min{9+COST(2,2),7+COST(2,3),
3+COST(2,4),2+COST(2,5)}
= 16



S到t的最小成本路径的成本 = 16

★ 最小成本路径的求取

记 D(i,j)＝每一COST(i,j)的决策

即，使c(j,l)+COST(i+1,l)取得最小值的l值。

例：D(3,6) = 10, D(3,7)= 10 D(3,8) = 10

D(2,2) = 7, D(2,3) = 6,D(2,4) = 8,D(2,5) = 8

D(1,1) = 2

根据D(1,1)的决策值向后递推求取最小成本路径：

- v_2 = D(1,1)=2
- v_3 = D(2,D(1,1)) = D(2,2) =7
- v_4 = D(3,D(2,D(1,1))) = D(3,7) = 10

故由s到t的最小成本路径是：1→2→7→10→12

3) 算法描述

★ 结点的编号规则

源点s编号为1,然后依次对V_2、V_3...V_{k-1}中的结点编号，汇点t编号为n。

目的:使对COST和D的计算仅按n-1,n-2,...,1的次序计算即可，无需考虑标示结点所在段的第一个下标。

★ 算法描述

算法6.1 多段图的向前处理算法

```
procedure FGRAPH(E,k,n,P)
//输入是按段的顺序给结点编号的，有n个结点的k段图。E是边集，c(i,j)是边<i,j>的成本。P(1:k)带出最小成本路径//
  real COST(n); integer D(n-1),P(k),r,j,k,n
  COST(n)←0
  for j←n-1 to 1 by -1 do //计算COST(j)//
    设r是具有性质：<j,r>∈E且使c(j,r)+COST(r)取最小值的结点
    COST(j)←c(j,r) + COST(r)
    D(j) ←r     //记录决策值//
  repeat
  P(1)←1; P(k)←n
  for j←2 to k-1 do     //找路径上的第j个结点//
    P(j) ←D(P(j-1))     //回溯求出该路径//
  repeat
end FGRAPH
```

算法的时间复杂度

若G采用邻接表表示，总计算时间为：

$$\Theta(n+e)$$

3.用向后处理策略求解多段图问题

设 BP(i,j)是一条从源点s到V_i中的结点j的最小成本路径，

BCOST(i,j)是这条路径的成本。

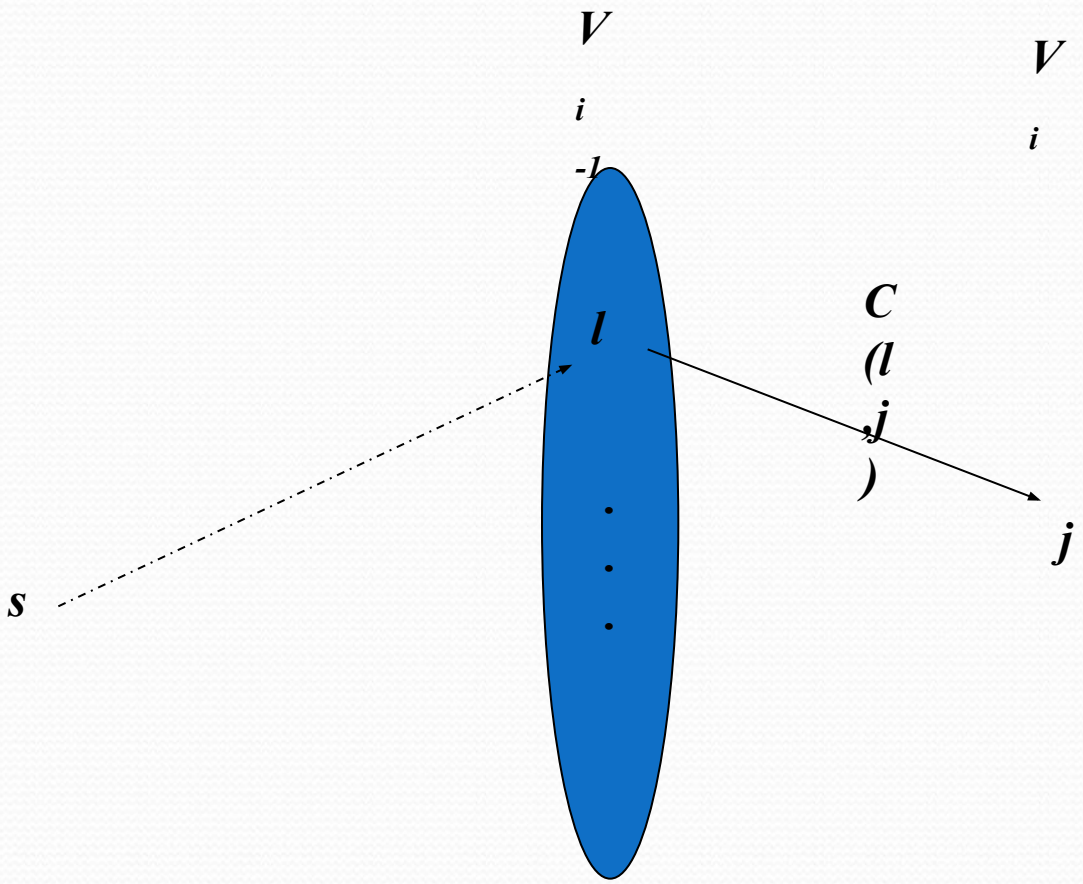
1) 向后递推式

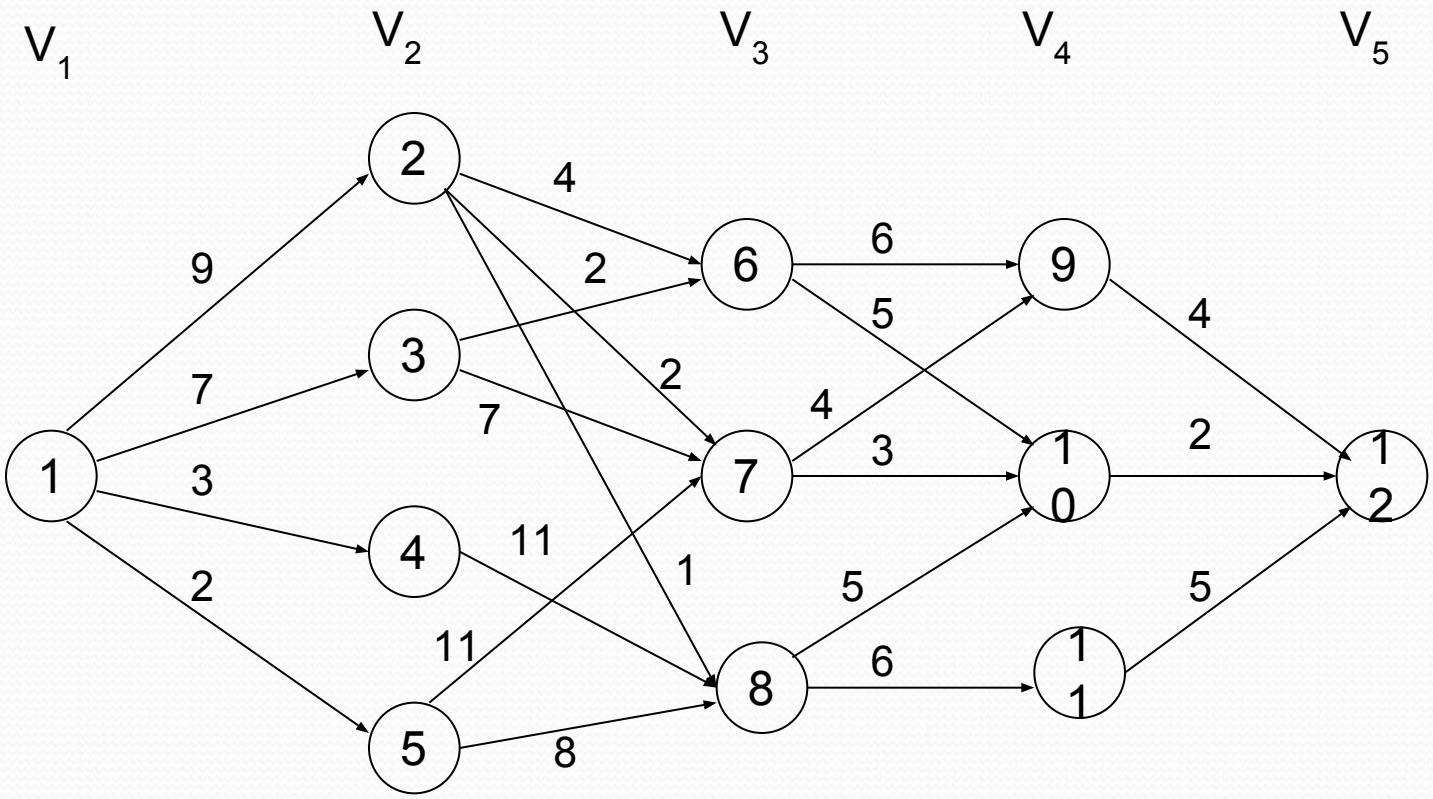
$$BCOST(i,j)=\min_{\substack{l\in v_{i-1}\\(l,j)\in E}}\{BCOST(i-1,l)+c(l,j)\}$$

2) 递推过程

★ 第2段

$$BCOST(2,j)=\begin{cases} c(1,j) & <1,j>\in E \\ \infty & <1,j>\notin E \end{cases}$$


V
i
-1
V
i
l
C
(l
j
)
j
s



5段图

$$BCOST(i,j) = \min_{\substack{l \in v_{i-1} \\ (l,j) \in E}} \{BCOST(i-1,l) + c(l,j)\}$$

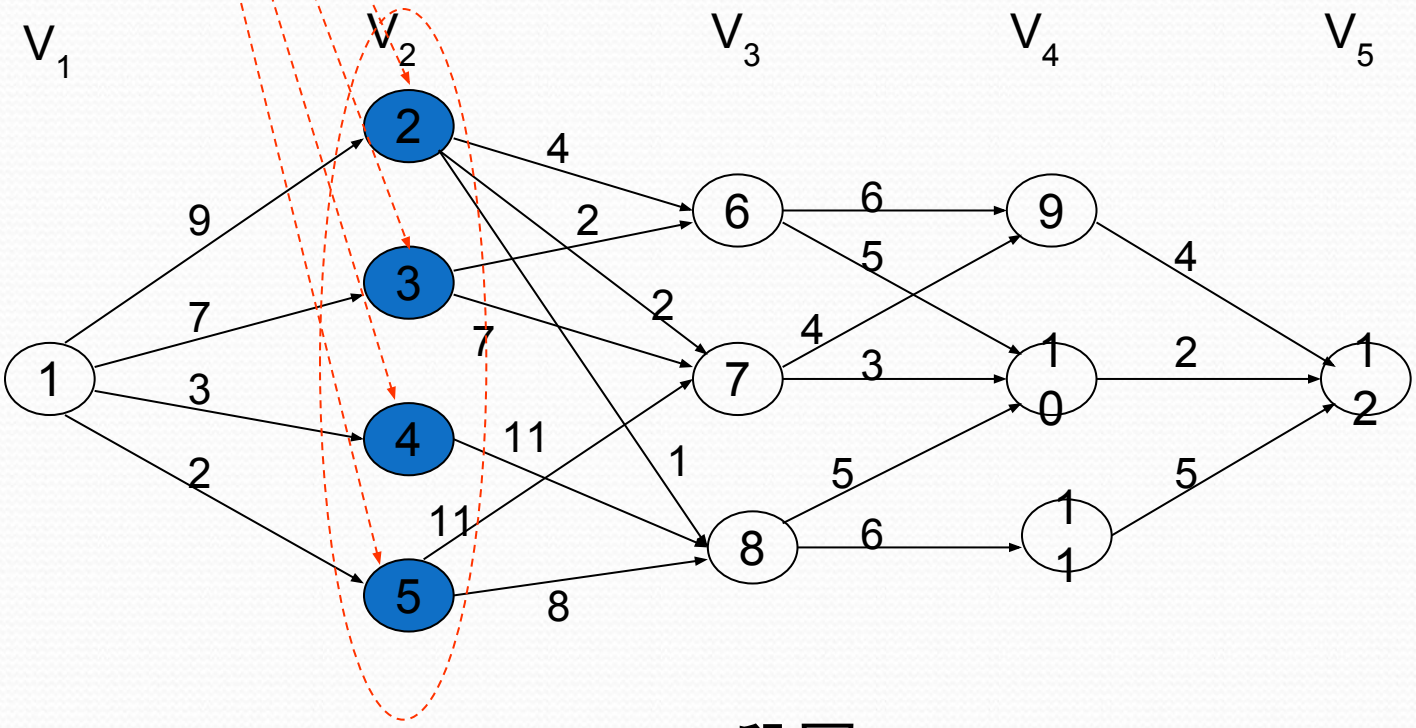
★各递推结果

第2段 BCOST(2,2) = 9

BCOST(2,3) = 7

BCOST(2,4) = 3

BCOST(2,5) = 2



5段图

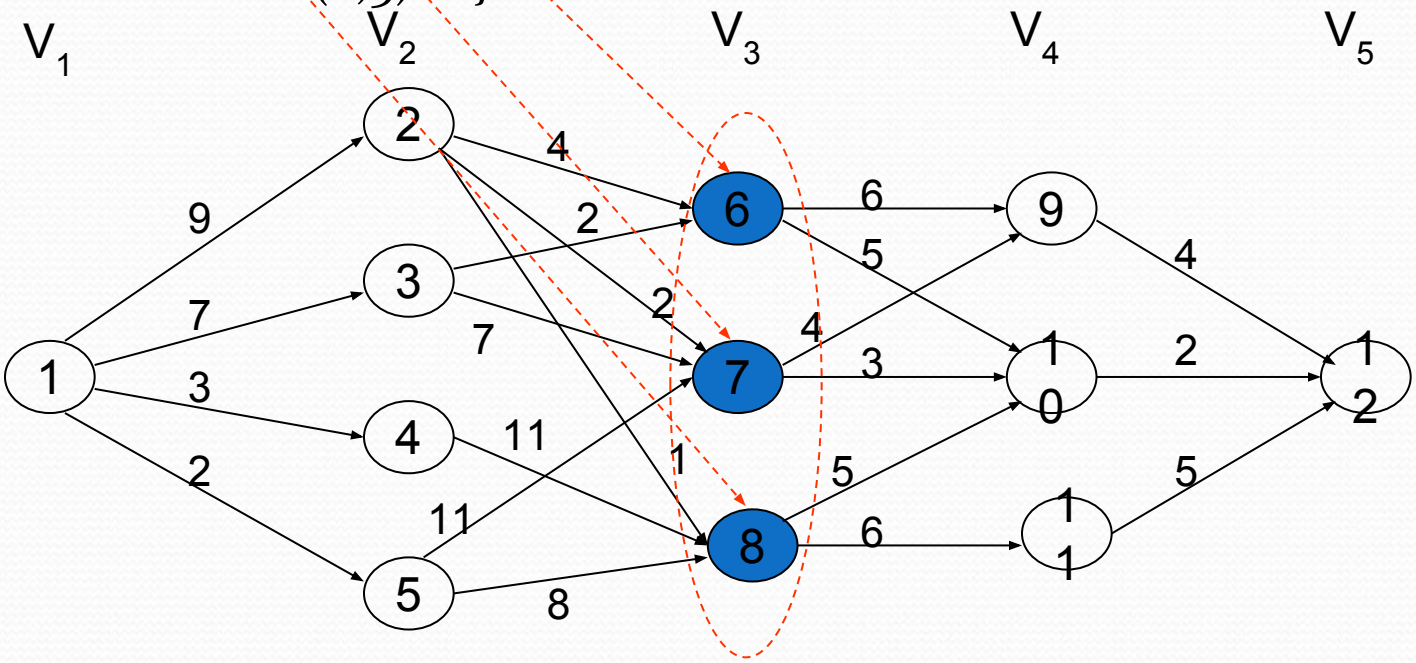
$$BCOST(i,j)=\min_{\substack{l\in v_{i-1}\\(l,j)\in E}}\{BCOST(i-1,l)+c(l,j)\}$$

★各递推结果

第3段 BCOST(3,6) = min{BCOST(2,2)+4,BCOST(2,3)+2} = 9

BCOST(3,7) = min{BCOST(2,2)+2,BCOST(2,3)+7,
BCOST(2,5)+11} = 11

BCOST(3,8) = min{BCOST(2,2)+1, BCOST(2,4)+11,
BCOST(2,5)+8} = 10



5段图

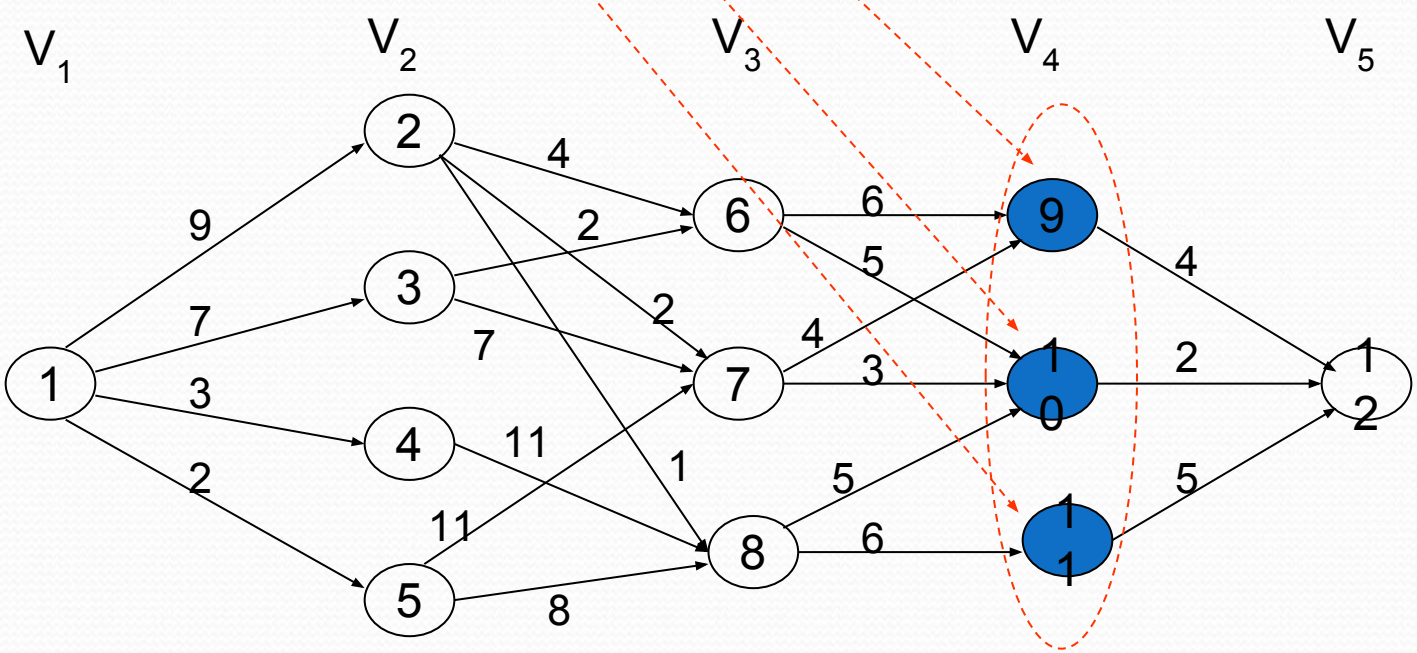
$$BCOST(i,j) = \min_{\substack{l \in V_{i-1} \\ (l,j) \in E}} \{BCOST(i-1,l) + c(l,j)\}$$

第4段 BCOST(4,9) = min{BCOST(3,6)+6,BCOST(3,7)+4} = 15

BCOST(4,10) = min{BCOST(3,6)+5,BCOST(3,7)+3,

BCOST(3,8)+5} = 14

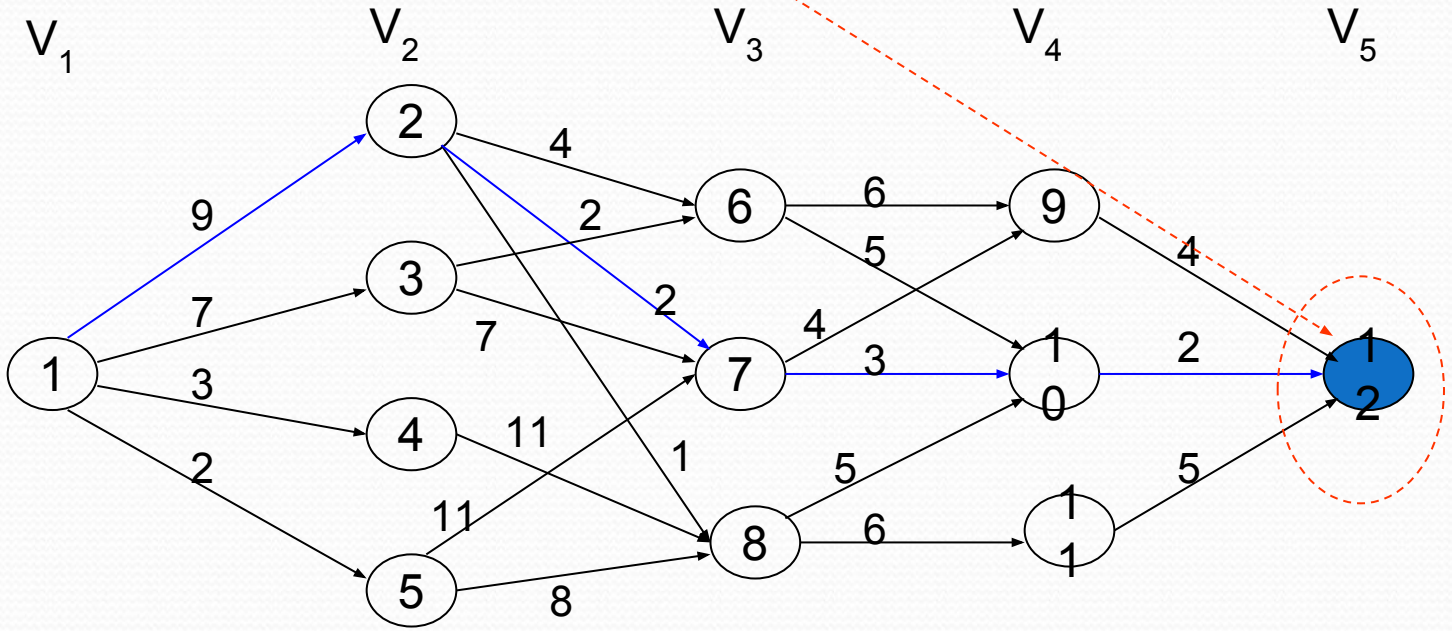
BCOST(4,11) = min{BCOST(3,8)+6} = 16



5段图

$$BCOST(i,j) = \min_{\substack{l \in V_{i-1} \\ (l,j) \in E}} \{BCOST(i-1,l) + c(l,j)\}$$

第5段 BCOST(5,12) = min{BCOST(4,9)+4, BCOST(4,10)+2,
BCOST(4,11)+5}
= 16



S到t的最小成本路径的成本 = 16

★ 最小成本路径的求取

记 BD(i,j)＝每一BCOST(i,j)的决策

即, 使BCOST(i-1,l)+c(l,j)取得最小值的l值。

例:BD(3,6) = 3, BD(3,7) = 2 ,BD(3,8) = 5

BD(4,9) = 6, BD(4,10) = 7, BD(4,11) = 8

BD(5,12) = 10

根据BD(5,12)的决策值向前递推求取最小成本路径:

- v_4 = BD(5,12)=10
- v_3 = BD(4,BD(5,12)) = 7
- v_2 = BD(3,BD(4,BD(5,12))) = BD(3,7) = 2

故由s到t的最小成本路径是:1→2→7→10→12

算法6.2 多段图的向后处理算法

```
procedure BGRAPH(E,k,n,P)
//输入是按段的顺序给结点编号的，有n个结点的k段图。E是边集，c(i,j)是边<i,j>的成本。P(1:k)带出最小成本路径//
  real BCOST(n); integer BD(n-1),P(k),r,j,k,n
  BCOST(1)←0
  for j←2  to n  do //计算BCOST(j)//
    设r是具有<r,j>∈E且使BCOST(r)+ c(r,j)取最小值性质的结点
    BCOST(j)← BCOST(r)+ c(r,j)
    BD(j) ←r    //记录决策值//
  repeat
  P(1)←1; P(k)←n
  for j←k-1 to 2 by -1 do     //找路径上的第j个结点//
     P(j) ←BD(P(j+1))        //回溯求出该路径//
  repeat
end BGRAPH
```

4.多段图问题的应用实例－资源的分配问题

将n个资源分配给r个项目的问题：如果把j个资源，$o \le j \le n$，分配给项目i，可以获得N(i,j)的净利。

问题：如何将这n个资源分配给r个项目才能使各项目获得的净利之和达到最大。

转换成一个多段图问题求解。

用r＋1段图描述该问题：

段：1到r之间的每个段i表示项目i。

结点：i=1时，只有一个结点——源点s =V(1,0)

当2≤i≤r时，每段有n+1个结点，每个结点具有形式

V(i,j)：表示到项目i之前为止，共把j个资源分配给了前i-1个项目，j=0,1,...,n。

汇点t=V(r+1,n)

边的一般形式：<V(i,j),V(i+1,l)>,j≤l,1≤i≤r

成本：

★ 当j≤l且1≤i≤r时，边<V(i,j),V(i+1,l)>赋予成本N(i,l-j),表示给项目i分配l-j个资源所可能获得的净利。

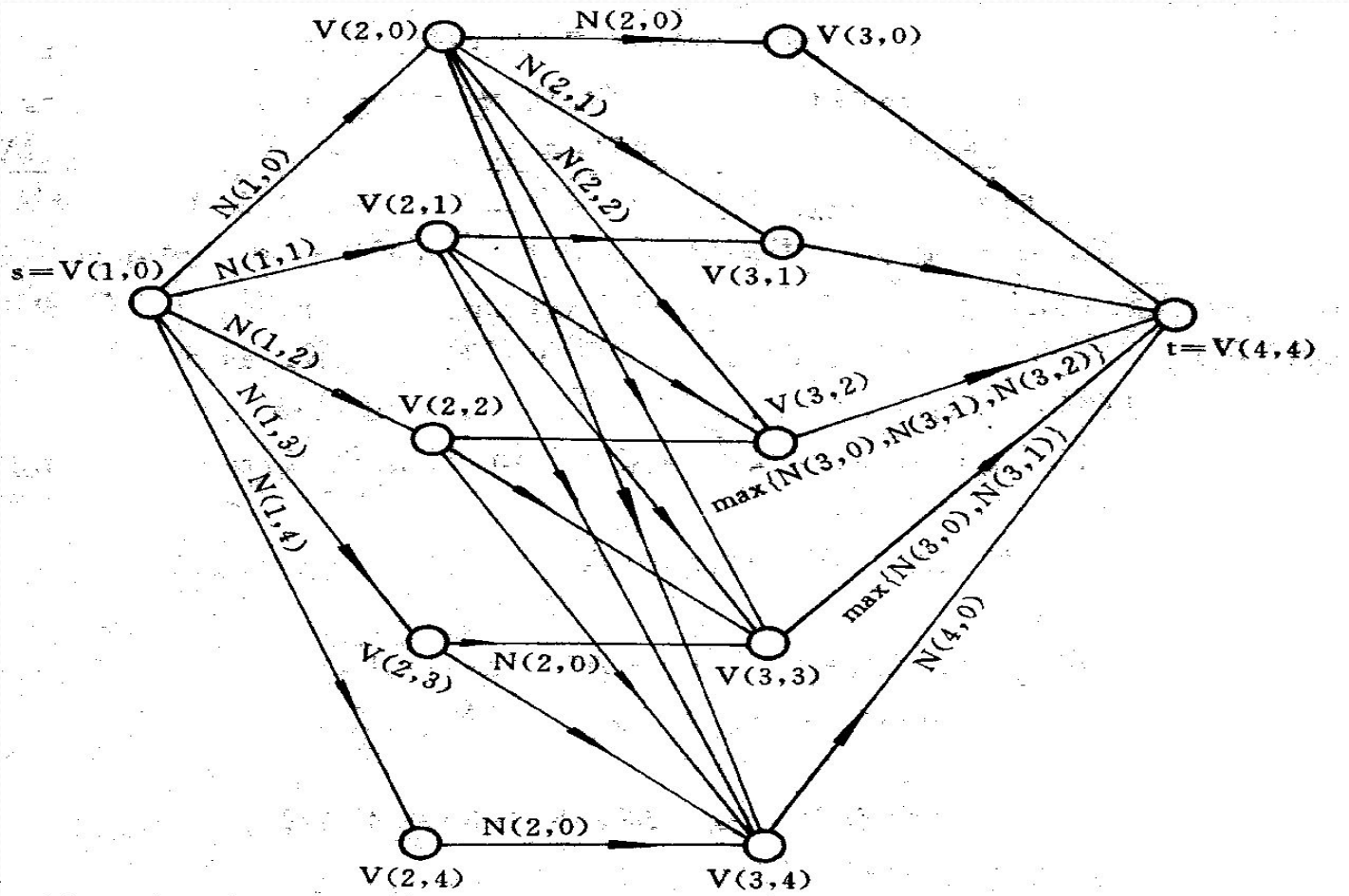
★ 当j≤n且i=r时，此时的边为：<V(r,j),V(r+1,n)>,该边赋予成本：

$$\max_{0\le p\le n-j}\{N(r,p)\}$$

注，并不是分给的资源越多，得到的净利就越大

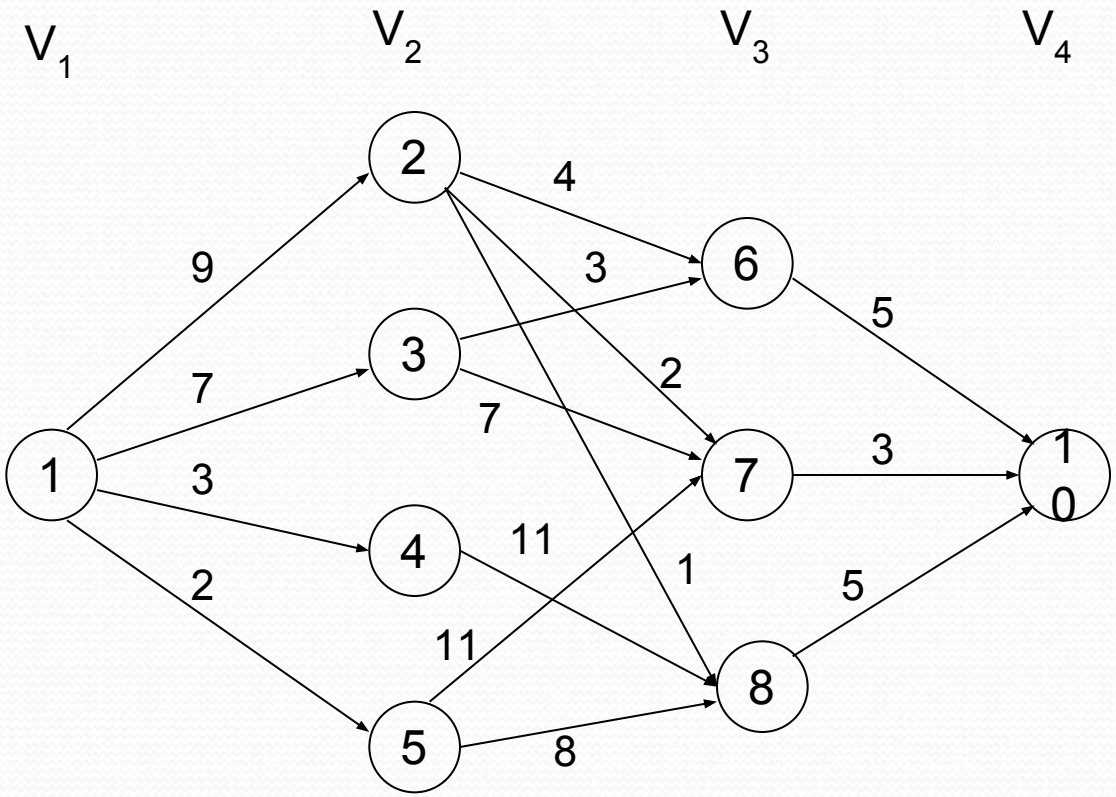
指向汇点的边

实例:将4个资源分配给3个项目。构成一个4段图。



问题的解:资源的最优分配方案由一条3到t的最大成本路径给出——改变边成本的符号, 从而将问题转换成为求取最小成本路径问题。

练习：



4段图

★各递推结果

第3段 COST(3,6) =
COST(3,7) =
COST(3,8) =

第2段 COST(2,2) =
COST(2,3) =
COST(2,4) =
COST(2,5) =

第1段 COST(1,1) =

★各递推结果

第3段　COST(3,6) = c(6,10) = 5

COST(3,7) = c(7,10) = 3

COST(3,8) = c(8,10) = 5

第2段

COST(2,2) = min{4+COST(3,6) , 2+COST(3,7), 1+COST(3,8)}

= 5

COST(2,3) = 8

COST(2,4) = 16

COST(2,5) = 13

第1段　COST(1,1) = min{9+COST(2,2),7+COST(2,3),

3+COST(2,4),2+COST(2,5)}

= 14

第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- 6.3 每对结点之间的最短路径
- 6.4 最优二分检索树
- 6.5 0/1背包问题

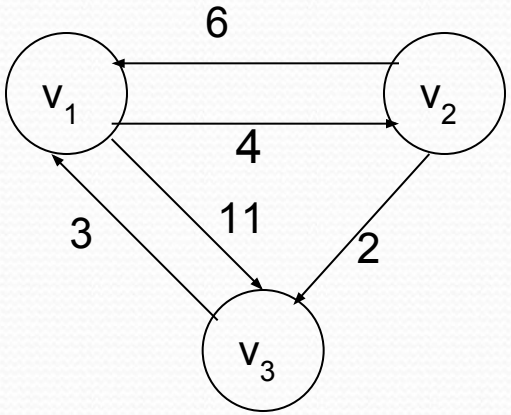
6.3 每对结点之间的最短路径

1. 问题描述

设G=(V,E)是一个有n个结点的有向图，C是G的成本邻接矩阵，C中元素有：

$$c(i,j)=\begin{cases}0, & i=j\\ \text{边}\langle i,j\rangle\text{的成本}, & i\neq j\text{且}\langle i,j\rangle\in E(G)\\ \infty, & i\neq j\text{且}\langle i,j\rangle\notin E(G)\end{cases}$$

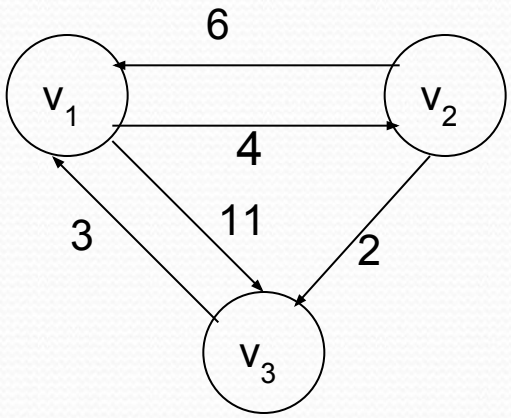
其中，1≤i，j≤n



	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

每对结点之间的最短路径问题：

求矩阵A，A中的任一元素A(i,j)代表结点i到结点j的最短路径成本。



	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

C：成本邻接矩阵

	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

A：每对结点间最短路径矩阵

分析：

- 利用单源最短路径算法求解——不能有负长度的边
  - 计算n个结点的单源最短路径。
  - 时间复杂度：$O(n^3)$

- 利用动态规划策略求解——不能有负长度的环

  将求解G中每对结点之间的最短路径问题转化成一个多阶段决策过程。
  - 决策什么？
  - 最优性原理对于该问题是否成立？

即证明每条最短路径都具有最优性原理刻画的性质

2. 动态规划求解策略

1) 最优性原理对于每对结点之间的最短路径问题成立

对G的一条由i到j的最短路径(假设该路径中不包含环)，设k是该路径上的一个中间结点：

i,...,k,...,j

由i到k的最短路径　k是中间结点　由k到i的最短路径

则，由i到k和k到j的两条子路径将分别是由i到k和由k到j的最短路径。(反证，否则i,...,k,...,j也将不是由i到j的最短路径)

故，最优性原理对于该问题成立。

2）多阶段决策过程

约定：对所有n个结点从1到n依次编号。

设 k是由i到j的最短路径上编号最高的中间结点：

i,...,k,...,j

k是编号最高的中间结点

则由i到k的子路径上将不会有比编号k-1更大的中间结点；同理，由k到j的子路径上也将不会有比编号k-1更大的中间结点。

构造一个“多阶段决策过程”：对由i到j的最短路径，尝试去寻求这个最大编号的中间结点k。

3）递推关系式

记 $A^k(i,j)$表示从i到j并且**不经过比k还大**的中间结点的最短路径成本。则

$A^0(i,j)$：i至j的路径上中间不包含任何中间结点

$A^1(i,j)$：i至j的路径上可以包含中间结点，但仅能是结点1

$A^2(i,j)$：i至j的路径上中间结点可以是结点1,2

$A^3(i,j)$：i至j的路径上中间结点可以是结点1,2,3

…

$A^n(i,j)$：i至j的路径上中间可以包含结点1,2,…,n(i,j除外)

因为所有结点的编号不会大于n，所以$A(i,j) = A^n(i,j)$，即由i到j的最短路径不通过编号比n还大的结点。

所有的A(i,j)构成结果矩阵A。

3）递推关系式

分析：对于 $A^n(i,j)$，路径可以经过结点n，也可以不经过结点n。

★ 若该路径经过结点n，则

$$A^n(i,j) = A^{n-1}(i,n) + A^{n-1}(n,j)$$

★ 若该路径不经过结点n，则

$$A^n(i,j) = A^{n-1}(i,j)$$

故可得，

$$A^n(i,j) = \min\{A^{n-1}(i,j) , A^{n-1}(i,n) + A^{n-1}(n,j)\}$$

不经过n结点　　经过n结点

中间结点不能比k大

对任意的k，k≥1，有，

$$A^k(i,j) = \min\{A^{k-1}(i,j)\ ,\ A^{k-1}(i,k) + A^{k-1}(k,j)\}$$

不经过结点k　　刚好经过结点k

初值：

$$A^0(i,j) = C(i,j),\ 1 \le i \le n,\ 1 \le j \le n$$

递推计算：$A^1(i,j) = \min\{A^0(i,j)\ ,\ A^0(i,1) + A^0(1,j)\}$

$A^2(i,j) = \min\{A^1(i,j)\ ,\ A^1(i,2) + A^1(2,j)\}$

…

$A^n(i,j) = \min\{A^{n-1}(i,j)\ ,\ A^{n-1}(i,n) + A^{n-1}(n,j)\}$

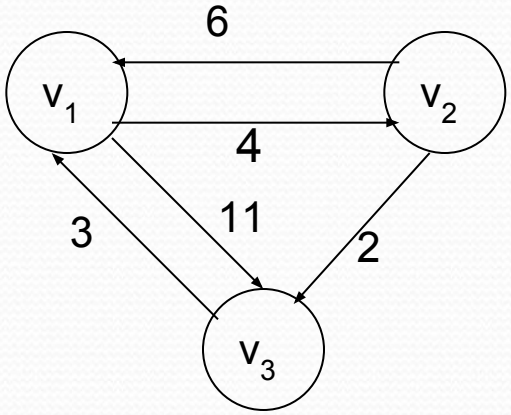
$= A(i,j)$

3. 算法描述

算法6.3 每对结点之间的最短路径长度

```
procedure ALL-PATHS(COST,A,n)
//COST(n,n)是n结点图的成本邻接矩阵;A(i,j)是结点v_i到v_j的最短路径的成本;COST(i,i)=0,1≤i≤n//
  integer i,j,k,n; real COST(n,n),A(n,n)
  for i←1 to n do
    for j←1 to n do
      A(i,j) ←COST(i,j)     //用COST(i,j)对A^0赋初值//
    repeat
  repeat
  for k←1 to n do           //k控制A^k(i,j) //
    for i←1 to n do
      for j←1 to n do
        A (i,j) ← min{A (i,j) , A (i,k) + A (k,j)}
      repeat
    repeat
  repeat
end ALL-PATHS
```

例6.8 有向图如图所示



	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

求图中所有结点间最短路径的成本矩阵

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

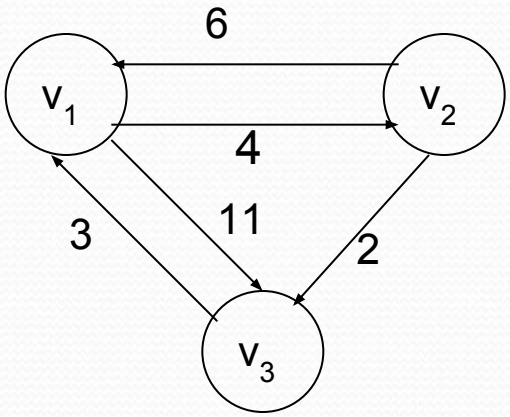
A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

注：A(i,j) = ∞表明G中从i到j没有有向路径

例6.8 有向图如图所示



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

算法的设计说明(1)：

$A^k(i,k) = A^{k-1}(i,k)$

$A^k(k,j) = A^{k-1}(k,j)$

即：在由第k-1到第k次的迭代过程中，A的第k行、第k列元素保持不变

$A^1(2,3)=\min\{A^0(2,3),A^0(2,1)+A^0(1,3)\}$

$=\min\{2, 6+11\}$

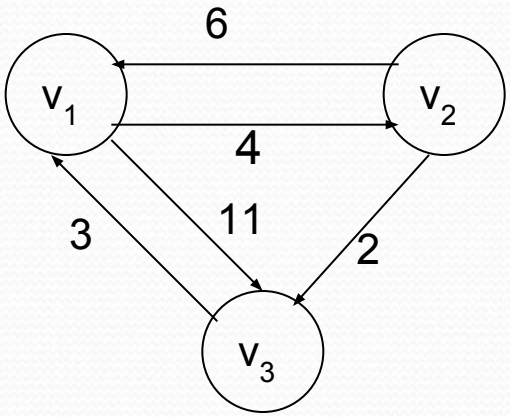
$=2$

$A^1(3,2)=\min\{A^0(3,2),A^0(3,1)+A^0(1,2)\}$

$=\min\{\infty, 3+4\}$

$=7$

例6.8 有向图如图所示



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

$A^2(1,3)=\min\{A^1(1,3),A^1(1,2)+A^1(2,3)\}$

$=\min\{11, 4+2\}$

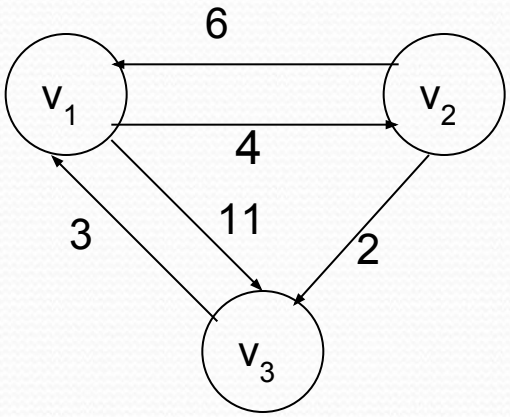
$=6$

$A^2(3,1)=\min\{A^1(3,1),A^1(3,2)+A^1(2,1)\}$

$=\min\{3, 7+6\}$

$=3$

例6.8 有向图如图所示



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

$A^3(1,2)=\min\{A^2(1,2),A^2(1,3)+A^2(3,2)\}$

$=\min\{4, 6+7\}$

$=4$

$A^3(2,1)=\min\{A^2(2,1),A^2(2,3)+A^2(3,1)\}$

$=\min\{6, 2+3\}$

$=5$

算法的设计说明

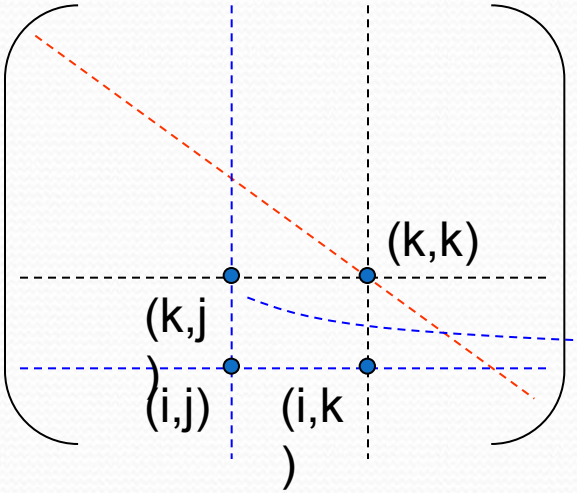
1)

∵ (1) $A^k(i,k) = A^{k-1}(i,k)$

$A^k(k,j) = A^{k-1}(k,j)$

在第k-1到第k次的迭代过程中，A的第k行、第k列元素不变

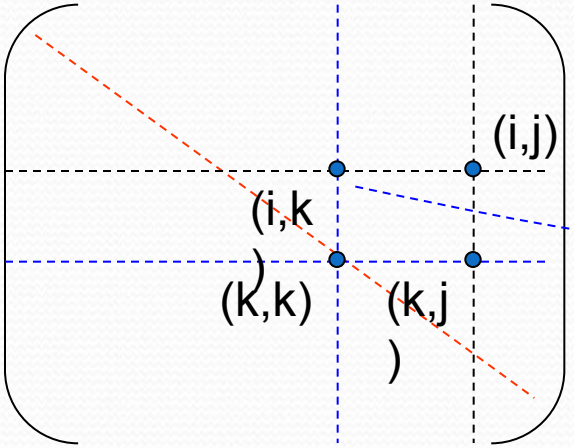
(2)



i>k且k>j,则有

$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^k(k,j)\}$

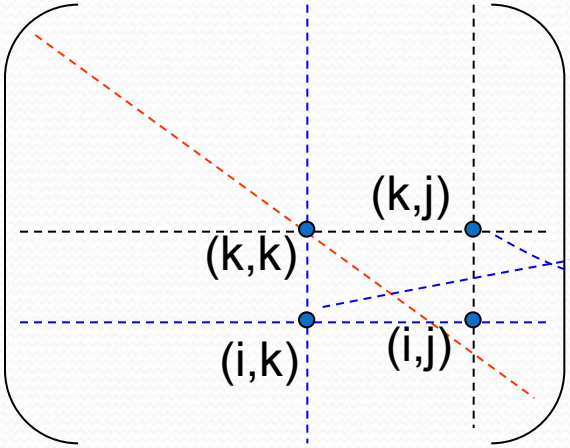
$A^{k-1}(k,j)$



k>i且j>k,则有

$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j), A^k(i,k) + A^{k-1}(k,j)\}$

$A^{k-1}(i,k)$



k<i且k<j,则有

$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j), A^k(i,k) + A^k(k,j)\}$

$A^{k-1}(i,k)$ $A^{k-1}(k,j)$

$$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j),\ A^k(i,k) + A^{k-1}(k,j)\}$$
$$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j),\ A^{k-1}(i,k) + A^k(k,j)\}$$
$$A^k(i,j) \leftarrow \min\{A^{k-1}(i,j),\ A^k(i,k) + A^k(k,j)\}$$
$$\equiv A^k(i,j) \leftarrow \min\{A^{k-1}(i,j),\ A^{k-1}(i,k) + A^{k-1}(k,j)\}$$

∴ 在算法的计算过程中取消了A的上标，并保证了每次计算的$A^k(i,j)$即为

$$\min\{A^{k-1}(i,j),\ A^{k-1}(i,k) + A^{k-1}(k,j)\}$$

2）如何求每对结点之间的路径？

性能分析

计算时间＝ $\Theta(n^3)$

注：该时间与A的值无关：

```
for k←1 to n do                          迭代n次
  for i←1 to n do                        迭代n次
    for j←1 to n do                      迭代n次
      A (i,j) ← min{A (i,j) , A (i,k) + A (k,j)}
    repeat
  repeat
repeat
```

第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- 6.3 每对结点之间的最短路径
- 6.4 最优二分检索树
- 6.5 0/1背包问题

6.4 最优二分检索树

1.二分检索树及其性能

二分检索树T是一棵二元树，它或者为空，或者其每个结点含有一个可以比较大小的数据元素，且有：

·T的左子树的所有元素比根结点中的元素小；

·T的右子树的所有元素比根结点中的元素大；

·T的左子树和右子树也是二分检索树。

注：

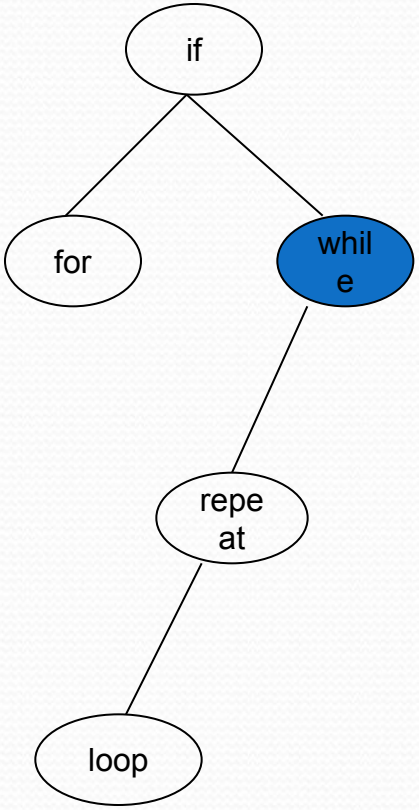
二分检索树要求树中所有结点的元素值互异

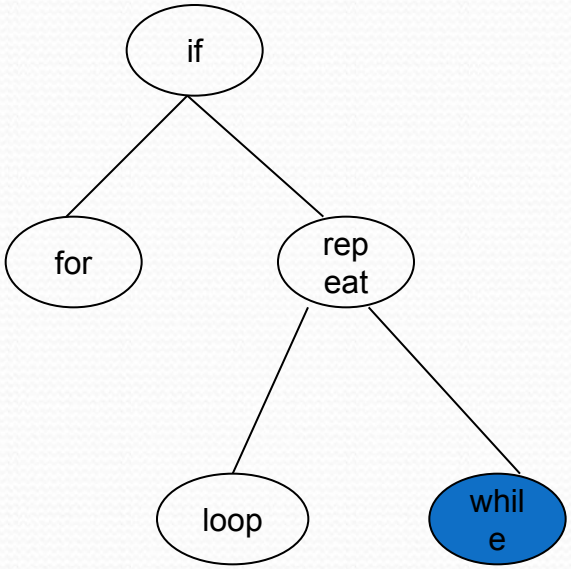
二分检索树的检索算法

算法6.4 SEARCH(T,X,i)

```
//在二分检索树T中检索X。如果X不在T中，则置i=o；否则i有IDENT(i)=X//
i←T
while i≠o do
  case
    :X＜IDENT(i):i←LCHILD(i) //若X小于IDENT(i)，则在左子树中继续查找//
    :X＝IDENT(i):return       //X等于IDENT(i)，则返回//
    :X＞IDENT(i):i←RCHILD(i) //若X大于IDENT(i)，则在右子树中继续查找//
  endcase
repeat
end SEARCH
```

对于一个给定的元素集合，以不同的元素当作根结点就会有不同形态的二分检索树，不同形态的二分检索树检索性能是不一样。以标识符集合为例:



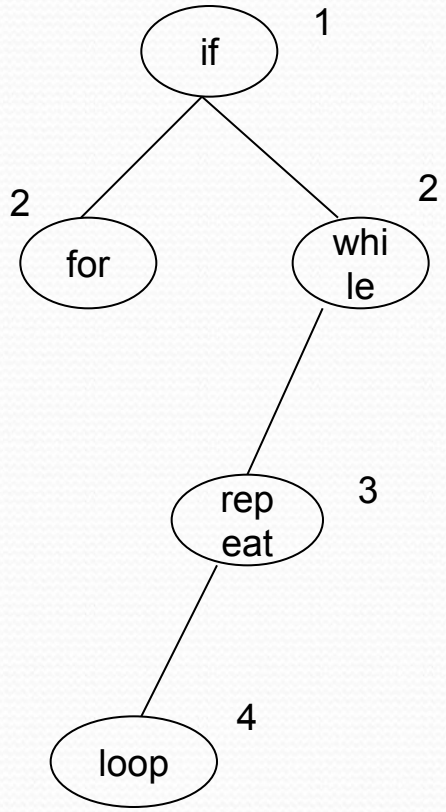


标识符集合：集合中的每个元素都是字符串。

应用实例：编译器

字符串的大小：字典顺序

如：for＜if＜loop＜repeat＜while



二分检索树的检索性能

区分成功检索和不成功检索两种情况：

➢成功检索：X恰为标识符集合中的一个元素。

►成功检索的情况共有n种，分别代表n个标识符之一。

➢不成功检索：X不是标识符集合中的任何元素。

►不成功检索的情况有n+1种（类似于二分检索）

引入检索的概率：每种情况都有被检索的概率。

成功检索的概率记为P_i，i=1,2,…,n

不成功检索的概率记为 Q_i,i=0,1,…,n

这里，所有成功检索和不成功检索是对同一标识符集合的所有检索需求的全集而言，即

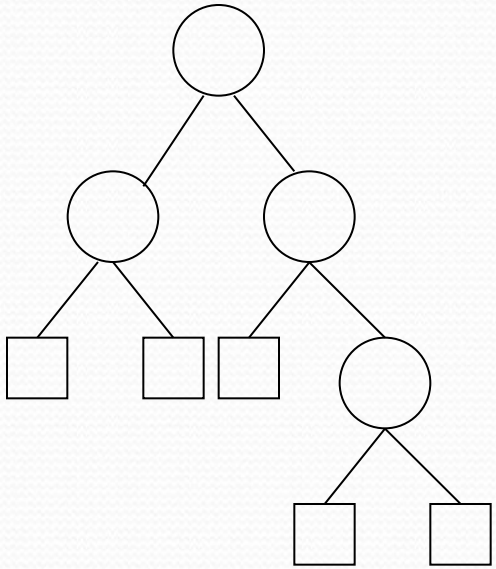
$$\sum_{i=1}^{n} P_i + \sum_{i=0}^{n} Q_i = 1$$

其中，$\sum_{i=1}^{n} P_i$：表示成功检索的总概率

$\sum_{i=0}^{n} Q_i$：表示不成功检索的总概率

每种检索情况所需的比较次数：

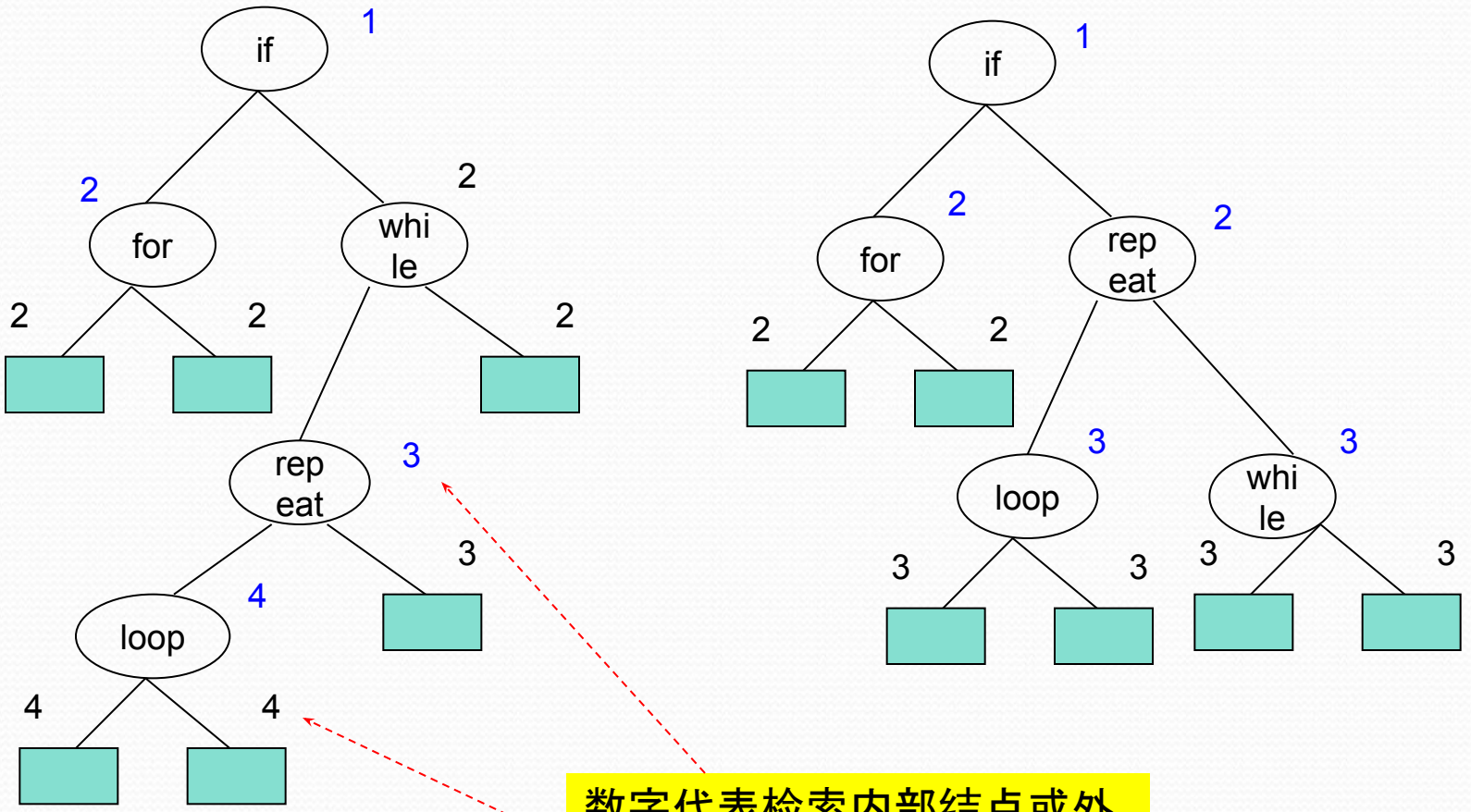
扩展的二分检索树表示：在二分检索树中加入代表不成功检索的外部结点。如图所示。



则在一棵二分检索树中，刚好有n个**内部结点**，代表n种成功检索情况；刚好有n＋1个**外部结点**，代表n+1种不成功检索的情况。并且，

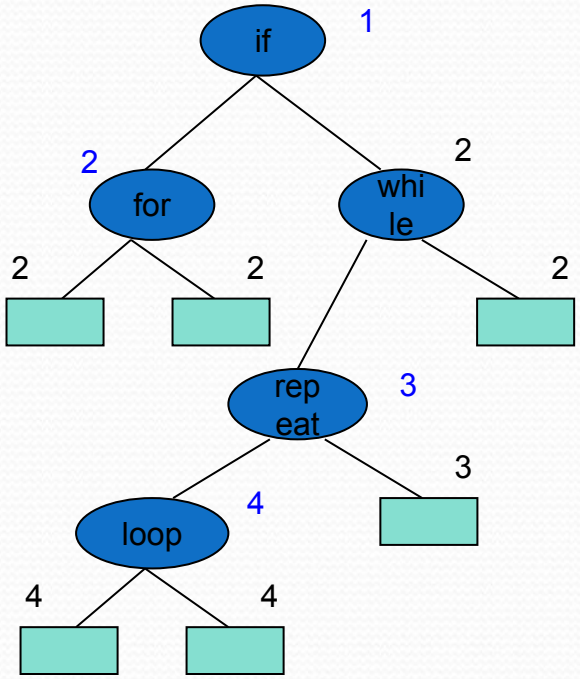
- 成功检索的比较次数等于结点在二分检索树中的级数，或根到该结点的距离+1。
- 不成功检索的比较次数等于外部结点在二分检索树中的级数-1，或根到该外部结点的距离。

例：



数字代表检索内部结点或外部结点时所需的比较次数

不同形态的二分检索树的检索性能不同，如：



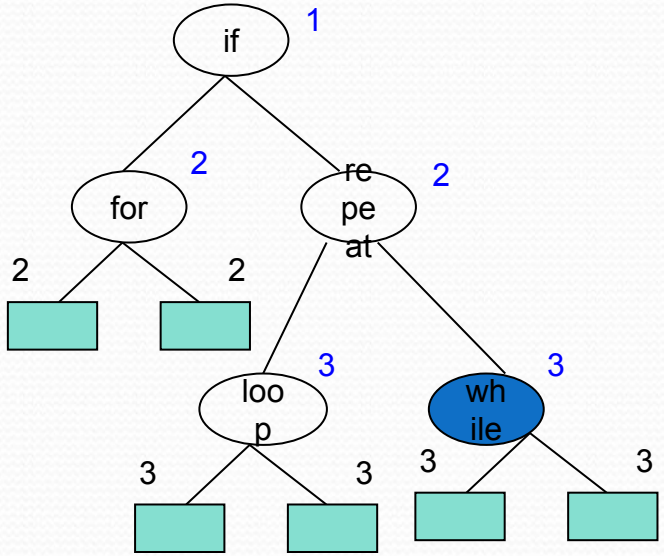
最坏：标识符loop需要做4次比较

平均：

$Avg_{成功}$=(1+2+2+3+4)/5=2.4

$Avg_{不成功}$=(2+2+2+3+4+4)/6=2.83

Avg=(1+2+2+3+4)+(2+2+2+3+4+4)/11=2.64



最坏：标识符loop/while需要做3次比较

平均：

$Avg_{成功}$=(1+2+2+3+3)/5=2.2

$Avg_{不成功}$=(2+2+3+3+3+3)/6=2.67

Avg=(1+2+2+3+3)+(2+2+3+3+3+3)/11=2.45

通过以上分析看出:对同一标识符集合而言，不同形态的二分检索树的性能是不同的。

- 如何衡量二分检索树的性能？
- 什么样的二分检索树性能最好？
- 何谓最优二分检索树？

最优二分检索树

给定的n个标识符的集合$\{a_1,a_2,...,a_n\}$, 假定$a_1<a_2<...<a_n$, 其最优二分检索树定义为具有最小**加权平均检索次数**的二分检索树。

二分检索树的**加权平均检索次数**:又称为该树的**预期成本**, 定义如下:

记

- P(i)为标识符a_i的检索概率, 即成功检索概率。
  (1≤i≤n, 即内部结点检索概率)
- Q(i)是第i种不成功检索情况的概率, 即待检索标识符X恰好处于:$a_i<X<a_{i+1}$, (设$a_0=-\infty,a_{n+1}=+\infty$)时的概率。
  (0≤i≤n, 即外部结点检索概率)。

二分检索树的预期成本(加权平均比较次数)

＝Σ每种情况出现的概率×该情况下所需的比较次数

注:

(1)平均检索成本的构成:成功检索部分＋不成功检索部分

(2)这里考虑所有可能的成功和不成功检索,共2n+1种可能的情况，有:

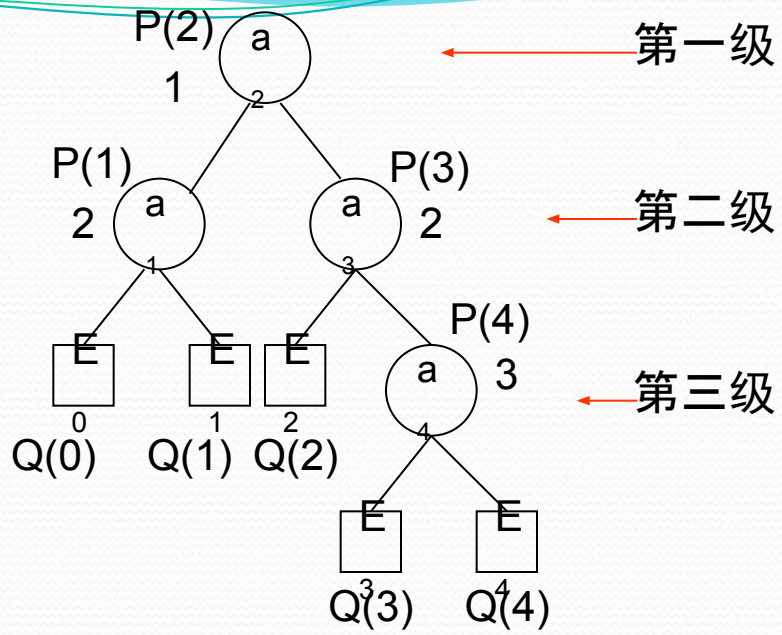
$\sum_{1\le i\le n} P(i)$ 表示所有成功检索的概率

$\sum_{0\le i\le n} Q(i)$ 表示所有不成功检索的概率

而所有情况下有: $\sum_{1\le i\le n} P(i) + \sum_{0\le i\le n} Q(i) = 1$

情况分析

(1)**成功检索**：在第l级的内结点终止的成功检索，需要做l次比较运算。



则，内部结点a_i在平均成本中的成本分担额为：

$$P(i)*level(a_i)$$

其中，P(i)为a_i的检索概率，

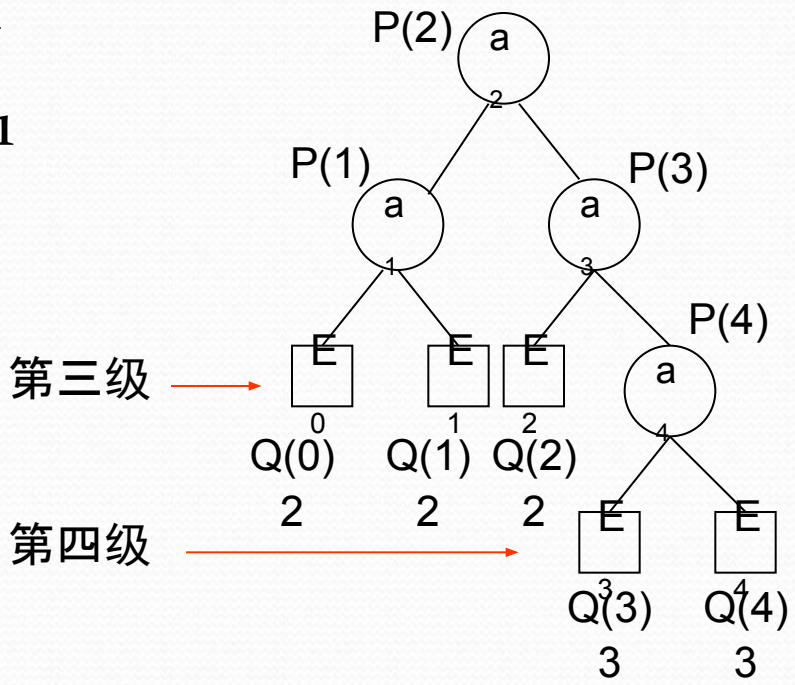
$level(a_i)$ = 结点a_i 的级数 $(1 \le i \le n)$。

(2)不成功检索：不成功检索的每一种情况定义了一个等价类，共有n+1个 这样的等价类$E_i(0 \le i \le n)$。其中，

$E_0=\{X|X<a_1\}$

$E_i=\{X|a_i<X<a_{i+1}(1 \le i<n)\}$

$E_n=\{X|X>a_n\}$



在二分检索树中用E_i表示各外部结点。若E_i处于第l级，则算法需作l-1次比较。则E_i在平均成本中的成本分担额为：

$$Q(i)*(level(E_i)-1)$$

其中，Q(i)为E_i的检索概率，$level(E_i)$ = 结点E_i 的级数 $(0 \le i \le n)$

平均检索成本的构成：

成功检索部分的成本＋不成功检索部分的成本

则，预期成本公式表示如下：

$$\mathrm{COST}(T)=\sum_{1\le i\le n}\mathrm{P(i)}*level(a_i)+\sum_{0\le i\le n}\mathrm{Q(i)}*(level(E_i)-1)$$

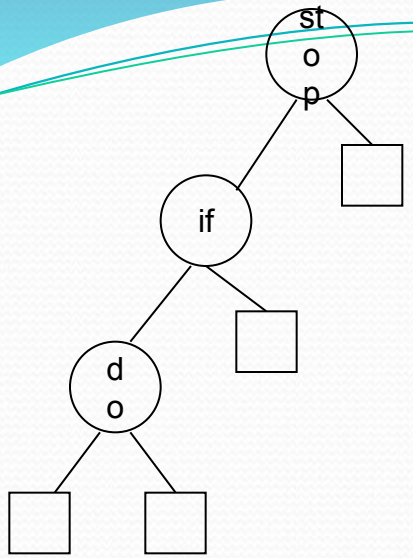
最优二分检索树问题：

求一棵具有最小预期成本的二分检索树

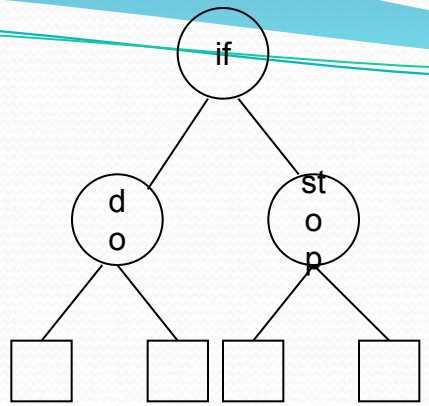
例6.9 已知标识符集合(a_1,a_2,a_3)=(do,if,stop)。

成 功 检 索:3种
不成功情况:4种

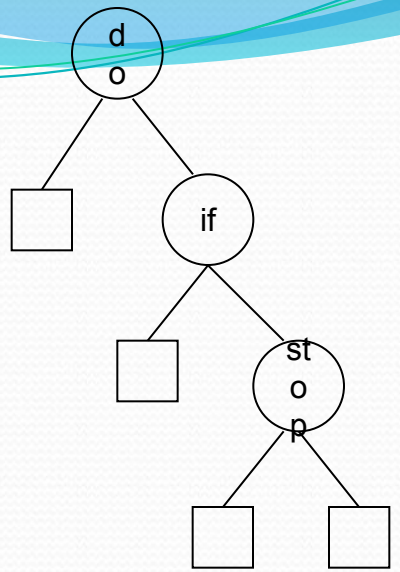
可能的二分检索树如下所示。



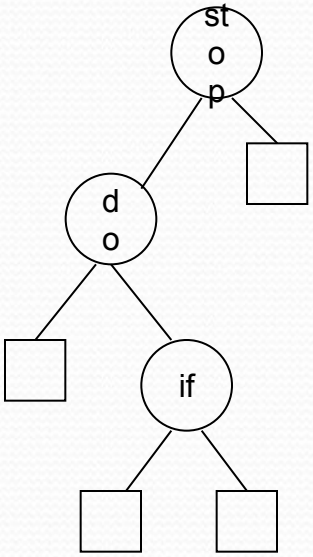
(a)



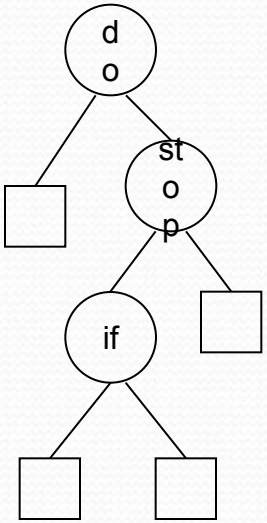
(b)



(c)



(d)



(e)

1) **等概率**：P(i)=Q(i)=1/7

cost(a) = 15/7

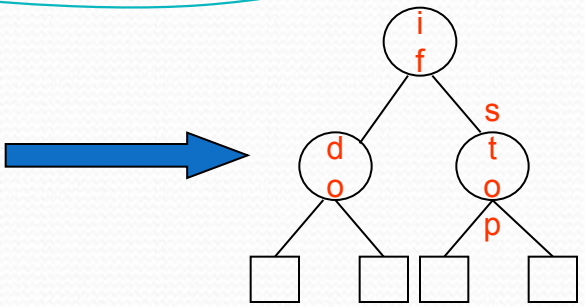
cost(b) = 13/7

cost(c) = 15/7

cost(d) = 15/7

cost(e) = 15/7

最优



(b)

2) **不等概率**：

P(1)=0.5　P(2)=0.1　P(3)=0.05

Q(0)=0.15　Q(1)=0.1　Q(2)=0.05　Q(3)=0.05

cost(a) = 2.65

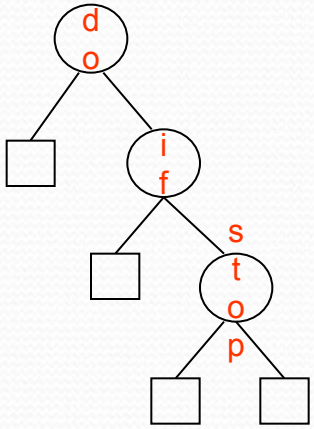
cost(b) = 1.9

cost(c) = 1.5

cost(d) = 2.15

cost(e) = 1.6

最优



(c)

cost(c) = 0.5X1+0.1X2+0.05X3+0.15X1+0.1X2+0.05X3+0.05X3=1.5

cost(b) = 0.5X2+0.1X1+0.05X2+0.15X2+0.1X2+0.05X2+0.05X2=1.9

对于给定的标识符集合，已知成功与不成功检索的概率，什么形态的二分检索树才是最优的二分检索树？

枚举法：列举各种形态的二分检索树，分析比较，找出最优的。

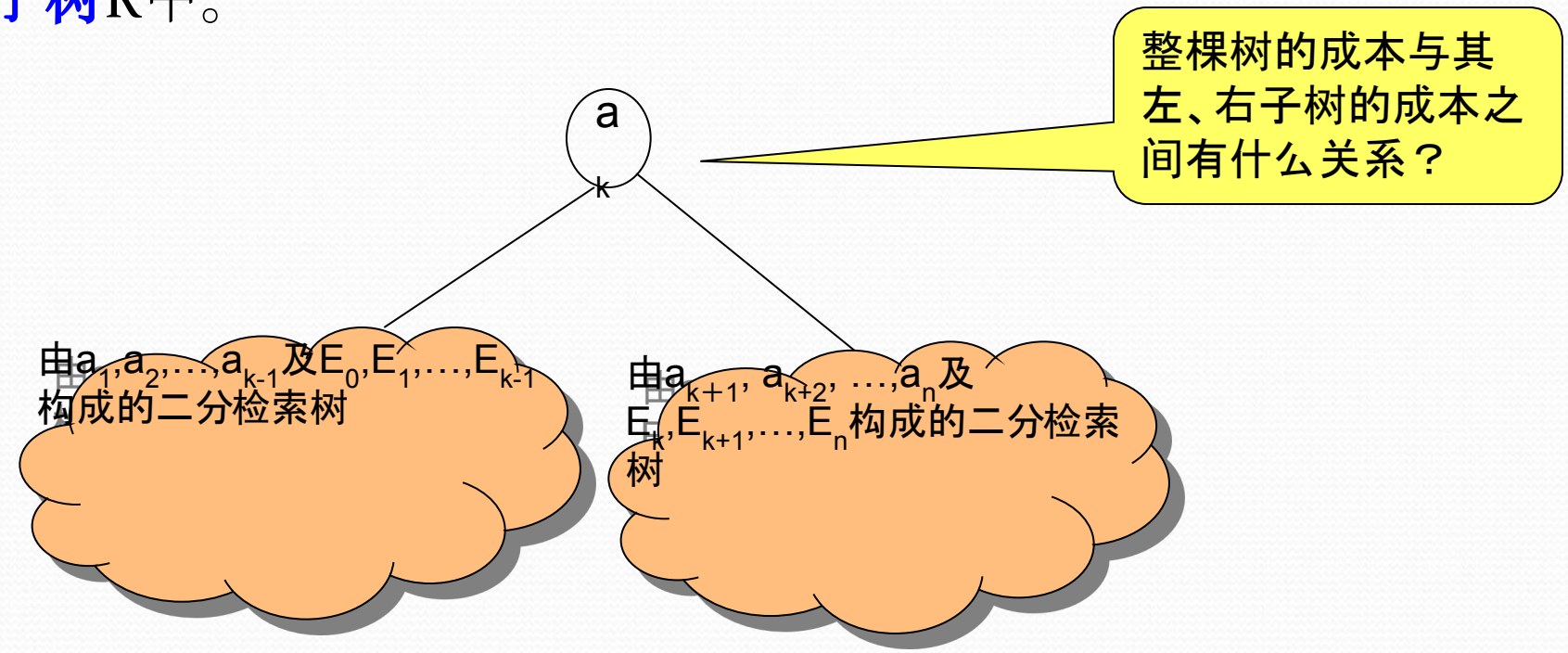
动态规划策略：把构造二分检索树的过程看成一系列决策的结果。

决策什么？

2. 多阶段决策过程

决策的对象：**树根**。

对$\{a_1,a_2,\ldots,a_n\}$ $(a_1<a_2<\ldots<a_n)$的二分检索树，若$\mathbf{a_k}$是该树的根，则内结点$a_1,a_2,\ldots,a_{k-1}$和外部结点$E_0,E_1,\ldots,E_{k-1}$将位于根a_k的**左子树**L中，而其余的结点 a_{k+1}, a_{k+2}, …,a_n及$E_k,E_{k+1},\ldots,E_n$将位于**右子树**R中。



定义：

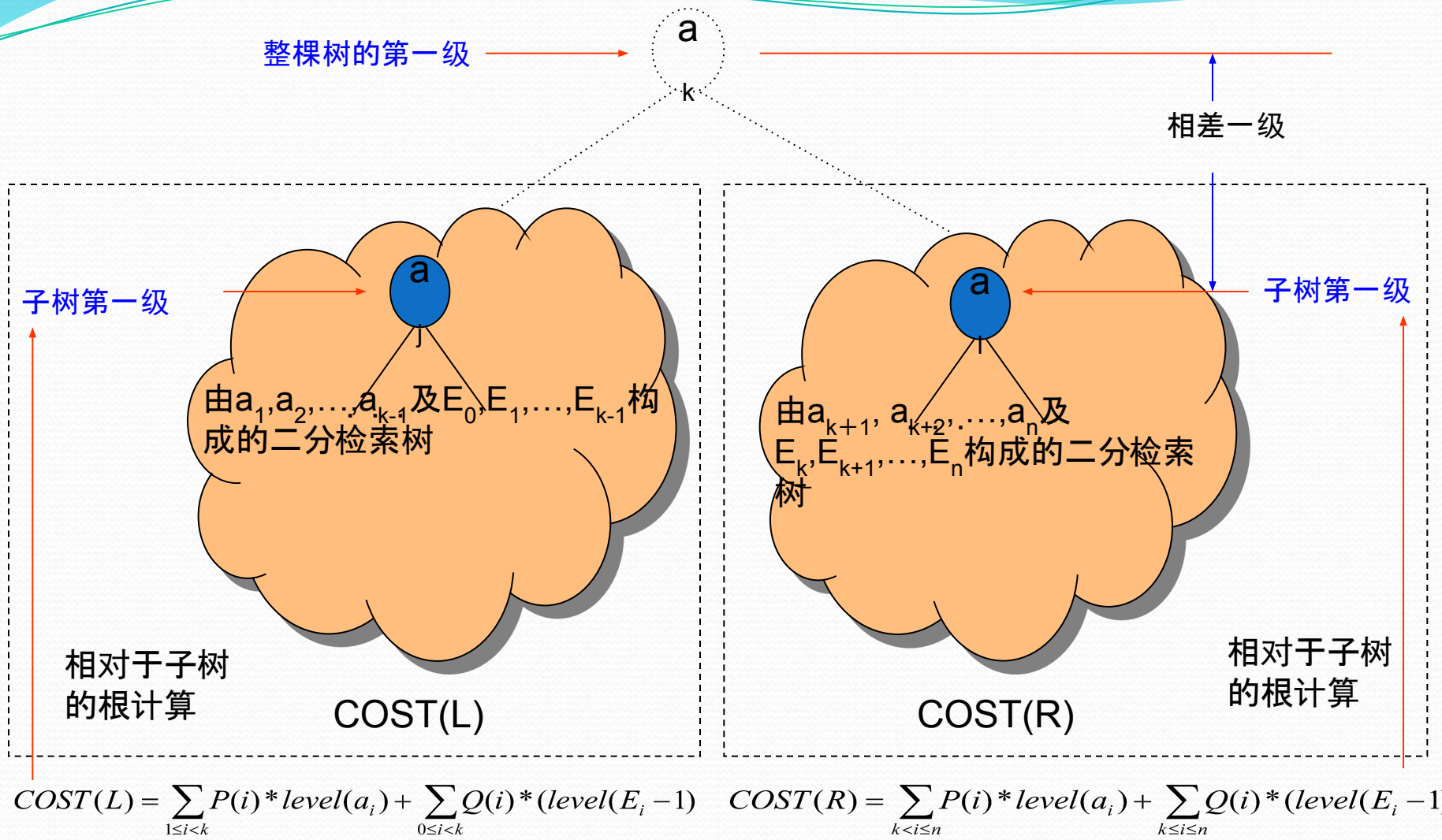
左子树的预期成本

$$COST(L)=\sum_{1\le i<k}P(i)*level(a_i)+\sum_{0\le i<k}Q(i)*(level(E_i)-1)$$

$$COST(R)=\sum_{k<i\le n}P(i)*level(a_i)+\sum_{k\le i\le n}Q(i)*(level(E_i)-1)$$

右子树的预期成本

含义：**左、右子树的预期成本**。这里，将左、右子树的根作为第一级来统计左、右子树中所有结点的级数，该级数等于相对原树根的级数－1。



$$COST(L)=\sum_{1\le i<k} P(i)*level(a_i)+\sum_{0\le i<k} Q(i)*(level(E_i-1)$$

$$COST(R)=\sum_{k<i\le n} P(i)*level(a_i)+\sum_{k\le i\le n} Q(i)*(level(E_i-1)$$

若：$level_T(X)$为结点X在树T中的级数，level(X)为X在T的左/右子树中的级数，则：$level_T(X) = level(X)+1$

$$COST(T) = \sum_{1\le i\le n} P(i) * level_T(a_i) + \sum_{0\le i\le n} Q(i) * (level_T(E_i) - 1)$$

$$= P(k) + \sum_{\substack{a_i \in \square\square\square \\ 1\le i\le k-1}} P(i) * (level(a_i) + 1) + \sum_{\substack{E_i \in \square\square\square \\ 0\le i\le k-1}} Q(i) * (level\ (E_i) + 1) - 1)$$

$$+ \sum_{\substack{a_i \in \square\square\square \\ k+1\le i\le n}} P(i) * (level(a_i) + 1) + \sum_{\substack{E_i \in \square\square\square \\ k\le i\le n}} Q(i) * (level\ (E_i) + 1) - 1)$$

$$= P(k) + \sum_{1\le i\le k-1} P(i) * level(a_i) + \sum_{0\le i\le k-1} Q(i) * (level(E_i) - 1)$$

$$+ \sum_{1\le i\le k-1} P(i) + \sum_{0\le i\le k-1} Q(i)$$

$$+ \sum_{k+1\le i\le n} P(i) * level(a_i) + \sum_{k\le i\le n} Q(i) * (level(E_i) - 1)$$

$$+ \sum_{k+1\le i\le n} P(i) + \sum_{k\le i\le n} Q(i)$$

根

左子树

右子树

$$\mathrm{COST}(T)=\sum_{1\le i\le n}\mathrm{P(i)}*level_T(a_i)+\sum_{0\le i\le n}\mathrm{Q(i)}*(level_T(E_i)-1)$$

$$=P(k)+\sum_{\substack{a_i\in\square\square\square\\ 1\le i\le k-1}}\mathrm{P(i)}*(level(a_i)+1)+\sum_{\substack{E_i\in\square\square\square\\ 0\le i\le k-1}}\mathrm{Q(i)}*(level\ (E_i)+1)-1)$$

$$+\sum_{\substack{a_i\in\square\square\square\\ k+1\le i\le n}}\mathrm{P(i)}*(level(a_i)+1)+\sum_{\substack{E_i\in\square\square\square\\ k\le i\le n}}\mathrm{Q(i)}*(level\ (E_i)+1)-1)$$

$$=\mathrm{P(k)}+\sum_{1\le i\le k-1}\mathrm{P(i)}*level(a_i)+\sum_{0\le i\le k-1}\mathrm{Q(i)}*(level(E_i)-1)$$

COST(L)

$$+\sum_{1\le i\le k-1}\mathrm{P(i)}+\sum_{0\le i\le k-1}\mathrm{Q(i)}\ \Rightarrow\ \mathrm{Q(0)}+\sum_{1\le i\le k-1}(\mathrm{P(i)}+\mathrm{Q(i)})$$

COST(R)

$$+\sum_{k+1\le i\le n}\mathrm{P(i)}*level(a_i)+\sum_{k\le i\le n}\mathrm{Q(i)}*(level(E_i)-1)$$

$$+\sum_{k+1\le i\le n}\mathrm{P(i)}+\sum_{k\le i\le n}\mathrm{Q(i)}\ \Rightarrow\ \mathrm{Q(k)}+\sum_{k+1\le i\le n}(\mathrm{P(i)}+\mathrm{Q(i)})$$

$$\mathrm{COST}(T) = \sum_{1\le i\le n} \mathrm{P(i)} * level_T(a_i) + \sum_{0\le i\le n} \mathrm{Q(i)} * (level_T(E_i) - 1)$$

$$= \mathrm{P(k)} + \sum_{1\le i\le k-1} \mathrm{P(i)} * level(a_i) + \sum_{0\le i\le k-1} \mathrm{Q(i)} * (level(E_i) - 1)$$

$$+ \sum_{1\le i\le k-1} \mathrm{P(i)} + \sum_{0\le i\le k-1} \mathrm{Q(i)}$$

$$+ \sum_{k+1\le i\le n} \mathrm{P(i)} * level(a_i) + \sum_{k\le i\le n} \mathrm{Q(i)} * (level(E_i) - 1)$$

$$+ \sum_{k+1\le i\le n} \mathrm{P(i)} + \sum_{k\le i\le n} \mathrm{Q(i)}$$

$$= \mathrm{P(k)} + \mathrm{COST(L)} + \sum_{1\le i\le k-1} \mathrm{P(i)} + \sum_{0\le i\le k-1} \mathrm{Q(i)}$$

$$+ \mathrm{COST(R)} + \sum_{k+1\le i\le n} \mathrm{P(i)} + \sum_{k\le i\le n} \mathrm{Q(i)}$$

记：$W(i,j)=Q(i)+\sum_{i+1}^{j}(Q(l)+P(l))$

则有：

$$W(0,k-1)=\sum_{1\le i\le k-1}\mathrm{P(i)}+\sum_{0\le i\le k-1}\mathrm{Q(i)}$$

$$W(k,n)=\sum_{k+1\le i\le n}\mathrm{P(i)}+\sum_{k\le i\le n}\mathrm{Q(i)}$$

左子树“成本差额”

右子树“成本差额”

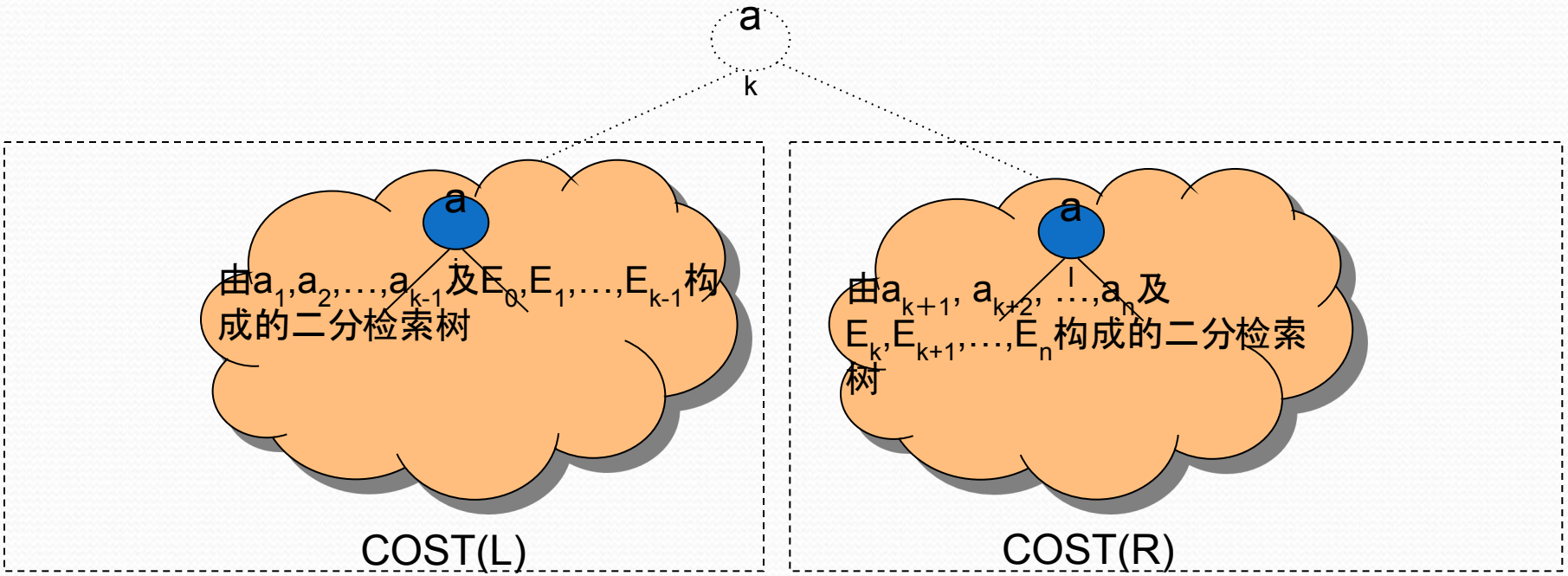
则，原二分检索树的预期成本可表示为：

$$\mathrm{COST}(T)=\mathrm{P(k)}+\mathrm{COST(L)}+W(0,k-1)+\mathrm{COST(R)}+W(k,n)$$

最优二分检索树：COST(T)有最小值

证明**最优性原理**对于上述问题是成立的：

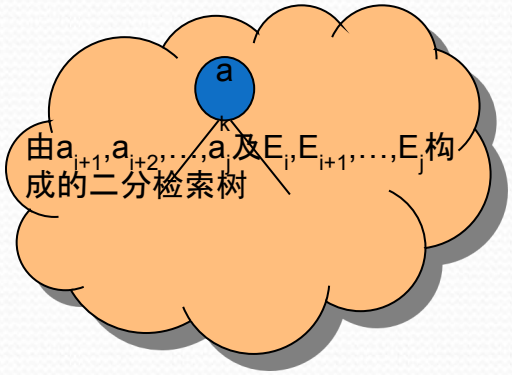
若以a_k为根的二分检索树T是关于集合$\{a_1,a_2,\ldots,a_n\}$的一棵最优二分检索树，L、R分别是a_k的左子树和右子树，L包含结点$a_1,a_2,\ldots,a_{k-1}$和$E_0,E_1,\ldots,E_{k-1}$、R包含结点及$a_{k+1}, a_{k+2}, \ldots,a_n$和$E_k,E_{k+1},\ldots,E_n$，则L和R必是相应集合的的最优二分检索树。COST(L)和COST(R)是其预期成本。



记由$a_{i+1},a_{i+2},\ldots,a_j$和$E_i,E_{i+1},\ldots,E_j$构成的**最优二分检索树**的预期成本为C(i,j)。

设二分检索树T的根结点为a_k, 则“最优”时有： COST(L) = C(o,k-ı)

COST(R) = C(k,n)



而,

$$C(0,n)=\min_{1\le k\le n}\{C(0,k-1)+C(k,n)+\underbrace{P(k)+W(0,k-1)+W(k,n)}_{W(0,n)}\}$$

对任何C(i,j)有,

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)+\underline{P(k)+W(i,k-1)+W(k,j)}\}$$
$$=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+\underline{W(i,j)}$$

递推过程如下：

★ 首先计算所有j-i=1的C(i,j)

★ 然后依次计算j-i=2,3,…,n的C(i,j)。

★ C(0,n)=最优二分检索树的成本。

初始值　C(i,i) = 0

$$W(i,i) = Q(i),\ 0 \le i \le n$$

最优二分检索树的构造方法：

- 在计算C(i,j)的过程中，记下使C(i,j)取得最小值的k值，记为R(i,j)。k对应的结点即是子树T_{ij}的根，。
- 依据R(0,n)…，推导树的形态。

例6.10 设n=4,且(a_1,a_2,a_3,a_4)=(do,if,read,while)。

设P(1:4) = (3,3,1,1)，Q(0:4) = (2,3,1,1,1) (概率值“扩大”了16倍)

初始：W(i,i)=Q(i)

C(i,i)=0

R(i,i)=0

且有，W(i,j)=P(j)+Q(j)+W(i,j-1)

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+W(i,j)$$

j-i=1阶段的计算：

W(0,1)=P(1)+Q(1)+W(0,0) = 3+3+2 = 8

①C(0,1)=W(0,1)+min{C(0,0)+C(1,1)} = 8

R(0,1)=1

W(1,2)=P(2)+Q(2)+W(1,1) = 7

②C(1,2)=W(1,2)+min{C(1,1)+C(2,2)} = 7

R(1,2)=2

W(2,3)=P(3)+Q(3)+W(2,2) = 3

③C(2,3)=W(2,3)+min{C(2,2)+C(3,3)} = 3

R(2,3)=3

W(3,4)=P(4)+Q(4)+W(3,3) = 3

④C(3,4)=W(3,4)+min{C(3,3)+C(4,4)} = 3

R(3,4)=4

仅有一个内结点

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+W(i,j)$$

j-i=2阶段的计算：

W(0,2)=P(2)+Q(2)+W(0,1) = 12

C(0,2)=W(0,2)+min{C(0,0)+C(1,2), C(0,1)+C(2,2)}

=12 + min{0+7, 8+0}=19

R(0,2) = 1

W(1,3)=

C(1,3)=

R(1,3) =

W(2,4)=

C(2,4)=

R(2,4) =

j-i=2阶段的计算：

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+W(i,j)$$

W(0,2)=P(2)+Q(2)+W(0,1) = 12

C(0,2)=W(0,2)+min{C(0,0)+C(1,2), C(0,1)+C(2,2)}

=12 + min{0+7, 8+0}=19

R(0,2) = 1

W(1,3)=P(3)+Q(3)+W(1,2) = 9

C(1,3)=W(1,3)+min{C(1,1)+C(2,3), C(1,2)+C(3,3)} = 12

R(1,3) = 2

W(2,4)=P(4)+Q(4)+W(2,3) = 5

C(2,4)=W(2,4)+min{C(2,2)+C(3,4), C(2,3)+C(4,4)} = 8

R(2,4) = 3

j-i=3阶段的计算：

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+W(i,j)$$

W(0,3)=P(3)+Q(3)+W(0,2) = 14

C(0,3)=W(0,3)+min{C(0,0)+C(1,3), C(0,1)+C(2,3),
C(0,2)+C(3,3)}=14 + min{0+12, 8+3,19+0}=25

R(0,3) = 2

W(1,4)=

C(1,4)=

R(1,4) =

j-i=3阶段的计算：

$$C(i,j)=\min_{i<k\le j}\{C(i,k-1)+C(k,j)\}+W(i,j)$$

W(0,3)=P(3)+Q(3)+W(0,2) = 14

C(0,3)=W(0,3)+min{C(0,0)+C(1,3), C(0,1)+C(2,3),
C(0,2)+C(3,3)}=14 + min{0+12, 8+3,19+0}=25

R(0,3) = 2

W(1,4)=P(4)+Q(4)+W(1,3) = 11

C(1,4)=W(1,4)+min{C(1,1)+C(2,4), C(1,2)+C(3,4) ,
C(1,3)+C(4,4)} = 19

R(1,4) = 2

j-i=4阶段的计算：

W(0,4)=P(4)+Q(4)+W(0,3) = 1+1+14=16

C(0,4)=W(0,4)+min{C(0,0)+C(1,4),C(0,1)+C(2,4),
C(0,2)+C(3,4), C(0,3)+C(4,4)}

= 16 + min{0+19, 8+8,19+3, 25+0}

= 32

R(0,4) = 2

W(i,j), C(i,j), R(i,j)计算结果

j-i \ i	0	1	2	3	4
0	2,0,0	3,0,0	1,0,0	1,0,0	1,0,0
1	8,8,1	7,7,2	3,3,3	3,3,4	
2	12,19,1	9,12,2	5,8,3		
3	14,25,2	11,19,2			
4	16,32,2			W(i,j),C(i,j),R(i,j)	

则，C(0,4)=32

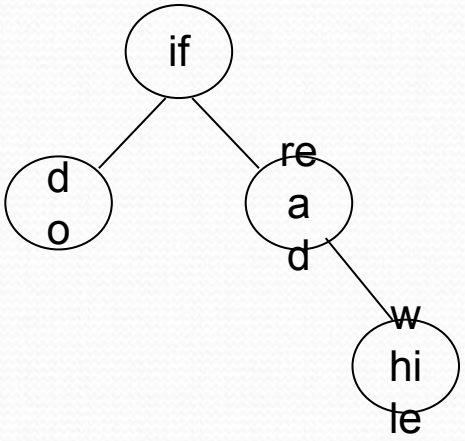
二分检索树：

$T_{04}=2 \Rightarrow T_{01}$(左子树)，$T_{24}$(右子树)

$T_{01}=1 \Rightarrow T_{00}$(左子树)，$T_{11}$(右子树)

$T_{24}=3 \Rightarrow T_{22}$(左子树)，$T_{34}$(右子树)

$T_{34}=4 \Rightarrow T_{33}$(左子树)，$T_{44}$(右子树)



3.计算时间分析

- 当j-i=m时，有n-m+1个C(i,j)要计算
- C(i,j)的计算：O(m)

  每个C(i,j)要求找出m个量中的最小值。则，n-m+1个C(i,j)的计算时间：

  $O(nm-m^2)$

对所有可能的m，总计算时间为：

$$\sum_{1\le m\le n}(nm-m^2)=O(n^3)$$

一种改进措施（克努特）：最优的$k\in[R(i,j-1),R(i+1,j)]$

Donald E. Knuth, Optimum binary search trees,Acta Information 1971

4.算法描述

```
procedure OBST(P,Q,n)
 //给定n个标识符a1＜a2＜…＜an。已知成功检索的概率P(i),不成功检索概率Q(i),  0≤i≤n。算法对于标识符ai+1, …, aj计算最优二分检索树Tij的成本C(i,j)、根 R(i,j)、权W(i,j) //
  real P(1:n),Q(1:n),C(0:n,0:n),W(0:n,0:n);integer R(0:n,0:n)
  for i←0 to n-1 do
    (W(i,i), R(i,i), C(i,i))←(Q(i),0,0)  //置初值//
    (W(i,i+1), R(i,i+1), C(i,i+1))←(Q(i)+Q(i+1)+P(i+1),i+1, Q(i)+Q(i+1)+P(i+1))
  repeat //含有一个结点的最优树//
  (W(n,n), R(n,n), C(n,n))←(Q(n),0,0)
  for m←2 to n do
    for i←0 to n-m do
      j←i+m
      W(i,j) ←W(i,j-1)+P(j)+Q(j)
      k←区间[R(i,j-1), R(i+1,j)]中使{C(i,l-1)+C(l,j)}取最小值的l值  //Knuth结论//
      C(i,j) ←W(i,j)+C(i,k-1)+C(k,j)
      R(i,j)←k
    repeat
  repeat
end OBST
```

OBST算法的计算时间:$O(n^2)$

第六章 动态规划

- 6.1 一般方法
- 6.2 多段图
- 6.3 每对结点之间的最短路径
- 6.4 最优二分检索树
- 6.5 0/1背包问题

6.5 0/1背包问题

1. 问题描述

1) KNAP(1,j,X)

目标函数：

$$\sum_{1\le i\le j} p_i x_i$$

约束条件：

$$\sum_{1\le i\le j} w_i x_i \le X$$

$$x_i = 0\text{或}1, p_i > 0, w_i > 0, 1 \le i \le j$$

0/1背包问题：KNAP(1,n,M)

最优性原理对于0/1背包问题成立

求解策略：向前递推、向后递推

2) 向后的递推求解

记$f_j(X)$是KNAP(1,j,X)的最优解，则$f_n(M)$有

$$f_n(M) = \max\{f_{n-1}(M), f_{n-1}(M-w_n)+p_n\}$$

对于任意的$f_i(X)$，$i>0$,有

$$f_i(X) = \max\{f_{i-1}(X), f_{i-1}(X-w_i)+p_i\}$$

递推过程：

★ 初始值

$$f_0 = \begin{cases} 0 & X \geq 0 \\ -\infty & X < 0 \end{cases}$$

★ 求出f_i在所有可能的X取值情况下的值。

★ $f_n(M)$=KNAP(1,n,M)

例6.11 背包问题: $n=3,(w_1,w_2,w_3)=(2,3,4),(p_1,p_2,p_3)=(1,2,5)$, $M=6$

递推计算过程

$$f_0(X)=\begin{cases}-\infty & X<0\\ 0 & X\geq 0\end{cases}$$

$$f_1(X)=\begin{cases}-\infty & X<0\\ \max\{0,-\infty+1\}=0 & 0\leq X<2\\ \max\{0,0+1\}=1 & X\geq 2\end{cases}$$

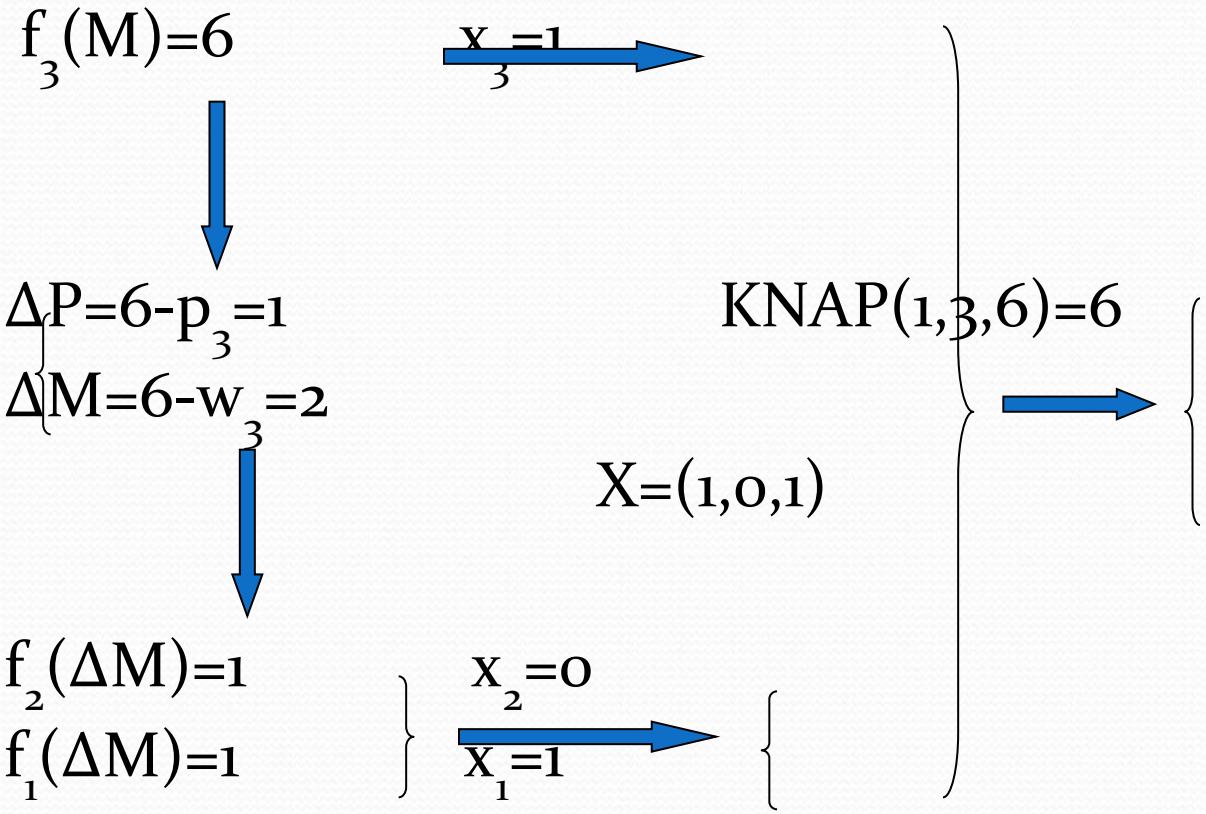
$$f_2(X)=\begin{cases}-\infty & X<0\\ \max\{0,-\infty+2\}=0 & 0\leq X<2\\ \max\{1,-\infty+2\}=1 & 2\leq X<3\\ \max\{1,0+2\}=2 & 3\leq X<5\\ \max\{1,1+2\}=3 & X\geq 5\end{cases}$$

$f_3(M)=\max\{3,1+5\}=6 \quad M=6$

$f_i(X)$是关于X的一个分段函数

尽管X≥0,但还不足以装下w_1

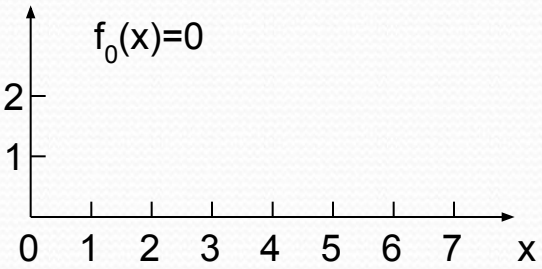
解向量的推导:根据特定情况下f_i取值的选择情况决定x_i的值

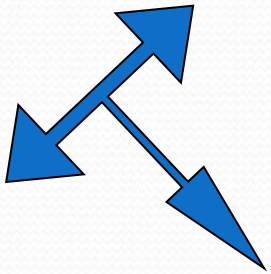


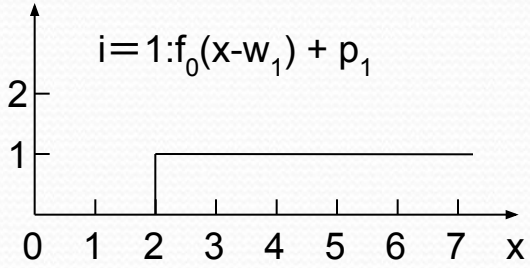
f_1, f_2, f_3计算过程的图解

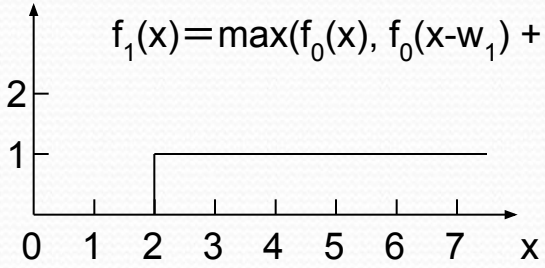
$i: f_{i-1}(x-w_i) + p_i$（装下i物品的情况）

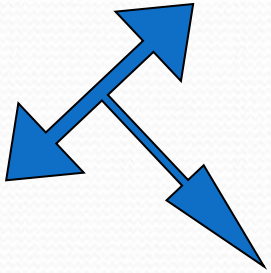
i=0:函数不存在

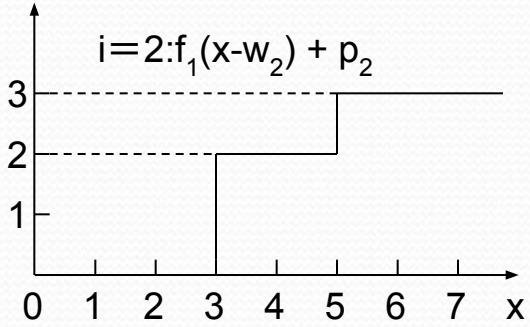


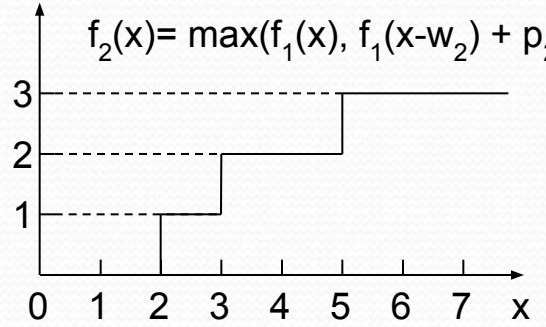




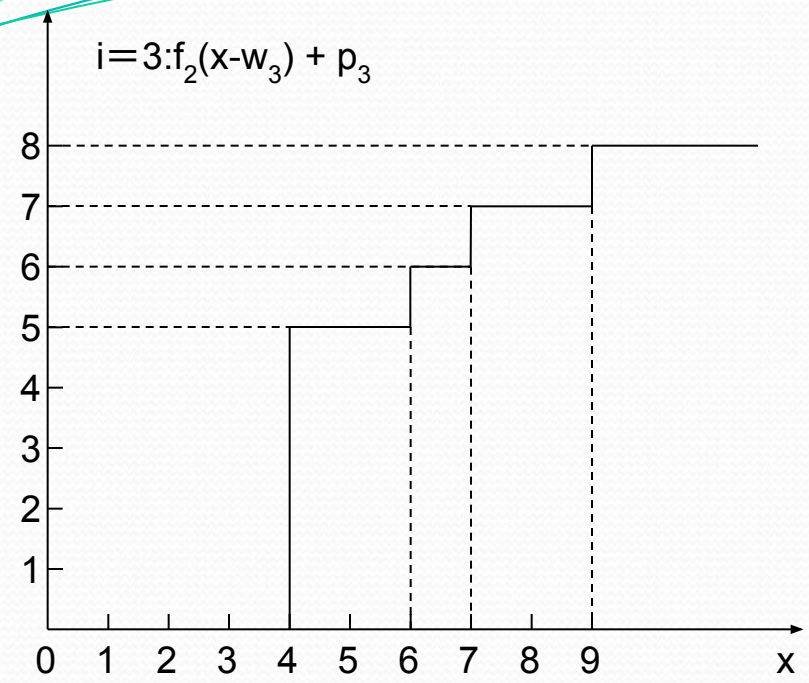


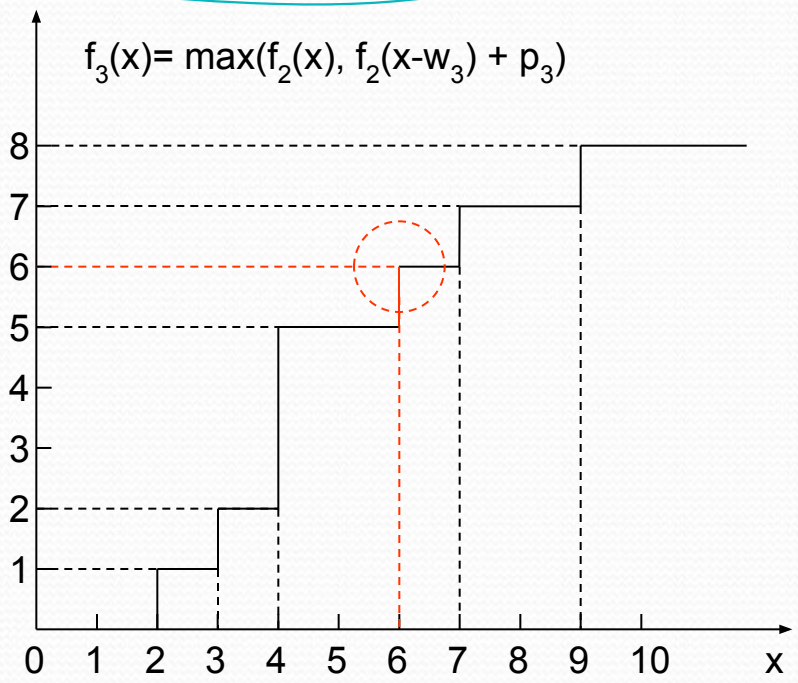






取大值原则





注：

- $f_{i-1}(X-w_i)+p_i$曲线的构造：将$f_{i-1}(X)$的曲线在X轴上右移w_i个单位，然后上移p_i个单位而得到；
- $f_i(X)$曲线的构造：由$f_{i-1}(X)$ 和$f_{i-1}(X-w_i)+p_i$的曲线按X相同时f取大值的规则归并而成

2. 用序偶表示法求0/1背包问题

f_i是关于X的阶跃函数，阶跃点是f_i的关键点。每个阶跃点用其对应坐标表示——称为一个序偶，f_i阶跃点的集合称为 f_i的序偶集合，即：

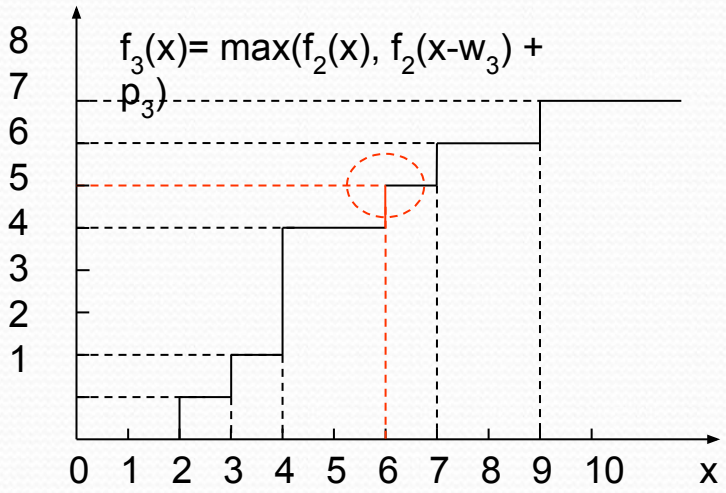
$S^i = \{(P_j,W_j) | W_j$是f_i曲线中使得f_i产生一次阶跃的X值，$P_j=f_i(W_j)$，

$0 \le j \le r$，r是阶跃点个数 $\}$

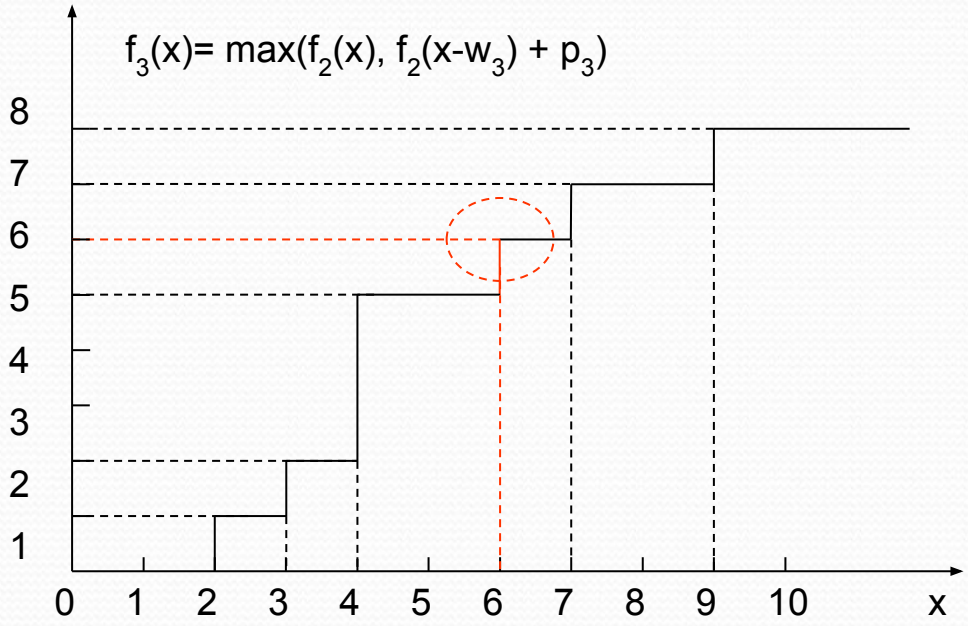
- $(P_0,W_0)=(0,0)$
- 若共有r个阶跃值，则分别对应r个(P_j,W_j)序偶，$1 \le j \le r$
- 若$W_j < W_{j+1}$，则$P_j < P_{j+1}$，$0 \le j < r$

  即f_i是关于X的单调递增函数
- 若$W_j \le X < W_{j+1}$，$f_i(X)=f_i(W_j)$

  即具有阶跃特点
- 若$X \ge W_r$，$f_i(X)=f_i(W_r)$



★ S^i中的序偶是背包问题KNAP(1,i,X)在X各种取值情况下子问题的最优解



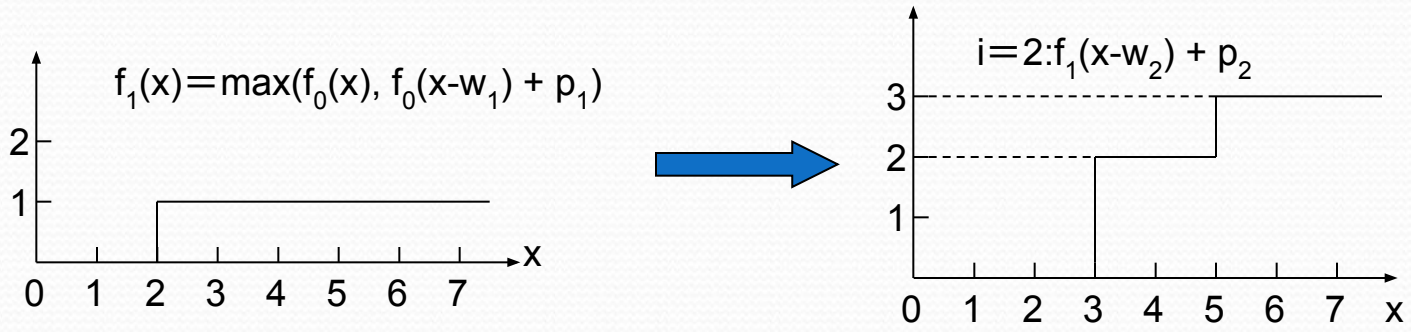
S^i的构造

即：在S^{i-1}的序偶分量上增加p_i、w_i生成

记 S_1^{i-1}是$f_{i-1}(X-w_i)+p_i$的所有序偶的集合，则

$$S_1^i = \{(P,W) \mid (P-p_i, W-w_i) \in S^{i-1}\}$$

其中，S^{i-1}是f_{i-1}的所有序偶的集合



S^i的构造：由S^{i-1}和 S_1^{i-1}按照支配规则合并而成。

支配规则：反映曲线合并过程中的取大值规则。即：

如果S^{i-1}和 S_1^{i-1}之一有序偶(P_j,W_j),另一有(P_k,W_k),

且有 $W_j \geq W_k$, $P_j \leq P_k$, 则序偶(P_j,W_j)将被舍弃。

例6.12 例6.11的序偶计算

$n=3,(w_1,w_2,w_3)=(2,3,4),(p_1,p_2,p_3)=(1,2,5),\ M=6$

$S^0=\{(0,0)\}$ $=\{(1,2)\}$ S_1^1

$S^1=\{(0,0),(1,2)\}$ $=\{(2,3),(3,5)\}$ S_1^2

$S^2=\{(0,0),(1,2),(2,3),(3,5)\}$

$=\{(5,4),(6,6),(7,7),(8,9)\}$ S_1^3

$S^3=\{(0,0),(1,2),(2,3),(5,4),(6,6),(7,7),(8,9)\}$

注：序偶(3,5)被(5,4)“支配”而删除

- 在S^i中，没有两个完全一样的序偶存在，即不存在i和k，使得(P_j,W_j)、$(P_k,W_k)\in S^i$且$W_j=W_k$且$P_j=P_k$，同时也不存在$W_j=W_k$或$P_j=P_k$。

- 若$W_j>W_k$则$P_j>P_k$，反之亦然，即序偶同时按照W_i和P_i递增有序。

如何求取决策序列？

分析S^i中序偶的来源：

S^i中的序偶或者来源于S^{i-1} 或者来源于 。 S_1^i

★ 若来源于S^{i-1}，则对当前的W计算$f_i(X)$时，表达式中$f_{i-1}(X)$的值大些，故第i件物品不装为好，即$x_i=0$。

否则

★ 来源于 ，S_1^{i-1}的效益值好些，第i件物品应该装入背包，$x_i=1$。

$$f_i(X) = \max\{f_{i-1}(X), f_{i-1}(X-w_i)+p_i\}$$

KNAP(1,n,M)问题的解——决策序列的求取

★ S^n是KNAP(1,n,X)在$0 \le X \le M$各种取值下的最优解

★ KNAP(1,n,M)的最优解由S^n的最后一对有效序偶(具有有效的最大W值的序偶)给出。

x_i的推导

x_n: 设S^n的最后一对有效序偶是(P_1,W_1), $W_1 \le M$, 则

(P_1,W_1)或者是S^{n-1}的最末一对有效序偶, 或者是

(P_j+p_n,W_j+w_n), 其中$(P_j,W_j) \in S^{n-1}$且W_j是S^{n-1}中满足$W_j+w_n \le M$的最大值。

- 若$(P_1,W_1) \in S^{n-1}$, 则$X_n=0$;否则,
- $(P_1-p_n,W_1-w_n) \in S^{n-1}$, $X_n=1$

x_{n-1}: 若$x_n=0$, 则判断S^{n-1}中(P_1,W_1)的来源, 以确定X_{n-1}的值

若$x_n=1$, 则判断S^{n-1}中(P_1-p_n,W_1-w_n)的来源, 以确定X_{n-1}的值

$x_{n-2},\ldots,x_1$将依次推导得出

例6.13（例6.12）

$S^0=\{(0,0)\}$

$S^1=\{(0,0),(1,2)\}$

$S^2=\{(0,0),(1,2),(2,3),(3,5)\}$

$S^3=\{(0,0),(1,2),(2,3),(5,4),(6,6),(7,7),(8,9)\}$

M=6, $f_3(6)$由S^3中的序偶(6,6)给出。

1) $\because (6,6) \notin S^2$

$\therefore x_3=1$

2) $\because (6-p_3,6-w_3)=(1,2)\in S^2$且$(1,2)\in S^1$

$\therefore x_2=0$

3) $\because (1,2) \notin S^0$

$\therefore x_1=1$

算法6.6 非形式的背包算法

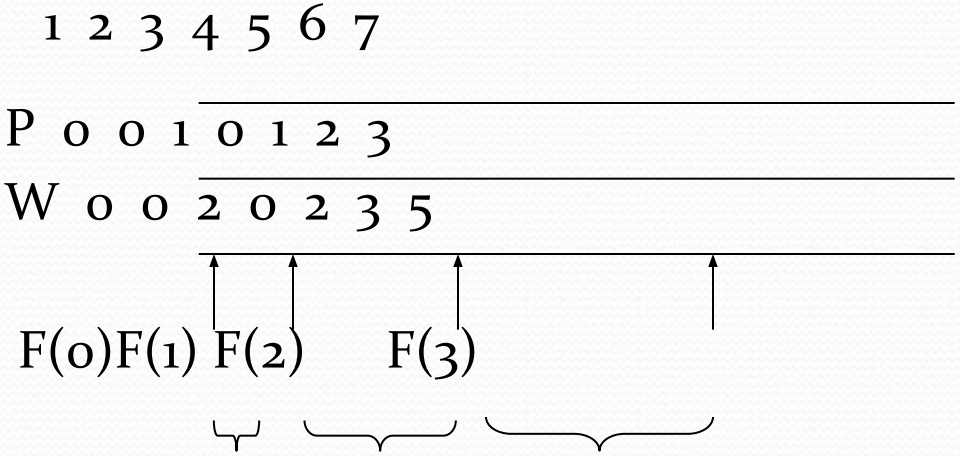
```
procedure DKP(p,w,n,M)
  S^0 ←{(0,0)}
  for i←1 to n-1 do
     ←S^i_1={(P_1,W_1)|(P_1-p_i,W_1-w_i)∈ S^{i-1} and W_1≤M} //生成   //
   S^i ←MERGE-PURGE(S^{i-1},   ) S^i_1/将S^{i-1}和   按支配规则归并为S^i//
  repeat
  (PX,WX)← S^{n-1}的最末一个有效序偶
  (PY,WY)←(P1＋p_n,W1+w_n), 其中, W1是S^{n-1}中使得W＋w_n≤M的
          所有序偶中取最大值得W
   //沿S^{n-1},..., S^1回溯确定x_n,x_{n-1},...,x_1的取值//
   if PX＞PY then x_n←0  //PX将是S^n的最末序偶//
         else  x_n←1  //PY将是S^n的最末序偶//
   endif
   回溯确定x_{n-1},...,x_1
 end DKP
```

3. DKP的实现

1)序偶集S^i的存储结构

使用两个一维数组P和W存放所有的序偶(P,W),其中P存放P值，W存放W值

- 序偶集S^0, S^1,..., S^{n-1}顺序、连续地存放于P和W中；
- 用指针F(i)表示S^i中第一个元素在数组P、W中的下标位置，$0 \le i \le n$;
- F(n)＝S^{n-1}中最末元素位置＋1



每一部分递增有序

2) 序偶的生成与合并

★ S^i的序偶将按照P和W的递增次序生成，并且

★ 中序偶的生成将与 和S_1^{i-1}的合并同时进行 S_1^i

设 生成的下一序偶是(PQ,WQ)，根据支配规则处理如下：

⑴ 将S^{i-1}中所有W值小于WQ的序偶直接计入S^i中；

⑵ 根据支配规则，若S^{i-1}中有W值小于WQ的序偶支配(PQ,WQ)，则(PQ,WQ)将被舍弃，否则(PQ,WQ)计入S^i中；

⑶ 清除S^{i-1}中所有可被(PQ,WQ)支配的序偶，这些序偶有W≥WQ且P≤PQ；

⑷ 对所有的(PQ,WQ)重复上述处理；

⑸ 将最后S^{i-1}中剩余的序偶直接计入S^i中（这是一些P和W均较大的序偶）；

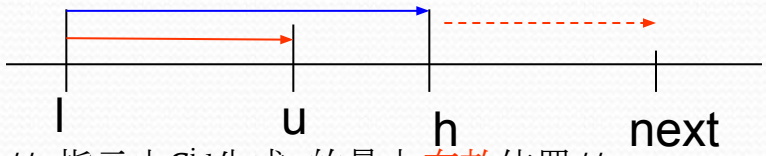
所有计入S^i中的新序偶依次存放到由F(i)指示的S^i的存放位置上。

注：不需要存放 的专用空间 S_1^{i}

3) 算法的实现

算法6.7 0/1背包问题的算法描述

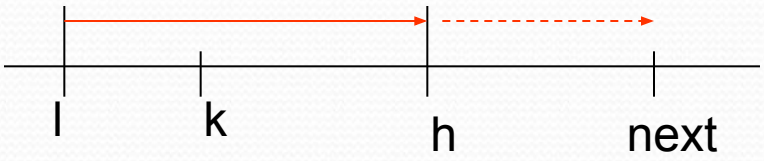
```
procedure DKNAP(p,w,n,M,m)
  real p(n), w(n), P(m), W(m),pp,ww,M;
  integer F(0:n),l,h,u,i, j,p,next
  F(0)←1;P(1)←W(1)←0    //S⁰//
  l←h←1     //S⁰ 的首端和末端;l是S^(i-1)的首端, h是S^(i-1)的末端//
  F(1)←next←2 //P和W中第一个空位; next指示P/W中可以存放(P,W)序偶的第一个位置//
  for i←1 to n-1 do //生成Sⁱ//
    k←l
    u←在l≤r≤h中使得W(r)+w_i≤M的最大r  //u指示由S^(i-1)生成 S_1^i 的最大有效位置//
    for j←l to u do //生成 S_1^i 同时进行归并//
      (pp,ww)←(P(j)+p_i,W(j)+w_i) //生成 S_1^i 中的下一个元素//
      while k≤h and W(k)＜ww do //从S^(i-1)中取元素并归并//
        P(next)←P(k);W(next)←W(k) //所有W(k)＜ww 的序偶直接归并//
        next←next+1;k←k＋1
      repeat
```



```
    //按照支配规则考虑(pp,ww)及S^{i-1}中的序偶//
    if k≤h and W(k)=ww then
       pp←max(pp,P(k)) k←k+1
    endif
    if pp＞P(next-1) then (P(next), W(next))←(pp,ww)
       next←next+1
    endif
    //清除S^{i-1}中的序偶//
    while k≤h and P(k)≤P(next-1) do k←k+1 repeat
  repeat
  //在并入 中的所有序偶之后，将S^{i-1}中剩余的元素并入S^i //
  while k≤h do
     (P(next),W(next))←(P(k),W(k))
     next←next+1; k←k+1
  repeat
  //对S^{i+1}置初值 //
  l←h+1; h←next -1; F(i+1)←next
 repeat
 CALL PARTS  //递推求取X_n,X_{n-1},...,x_1//
END DKNAP
```

此时W(next-1)≤ww

这些序偶的W(k)≥ww



4. 过程DKNAP的分析

1) 空间复杂度

记S^i中的序偶数为:$|S^i|$

则有，$|S^i| \le |S^{i-1}| + |S_1^i|$

又，$|S_1^i| \le |S^{i-1}|$

所以，$|S^i| \le 2|S^{i-1}|$

最坏情况下，由S^{i-1}生成 以及S_1^i生成S^i时没有序偶被支配，则

$$\sum_{0 \le i \le n-1} |S^i| = \sum_{0 \le i \le n-1} 2^i = 2^n - 1$$

故，DKNAP所需的空间复杂度（P、W数组）为:$O(2^n)$

2) 时间复杂度

由S^{i-1}生成S^i的时间为：$O(|S^{i-1}|)$，$0 \le i \le n-1$

故，DKNAP计算所有的S^i所需的时间为：

$$\sum_{0 \le i \le n-1} |S^i| = O(2^n)$$

进一步考察：

若每件物品的重量w_i和效益值p_i均为整数，则S^i中每个序偶(P,W)的P值和W值也是整数，且有 $P, \le W \le M$ $\sum_{0 \le j \le i} |p_j|$

问题：在1～n范围内有多少个互异的整数？

答案：n个

而在任一S^i中，所有序偶具有互异P值和W值。

故有：

$$|S^i| \le 1 + \sum_{0 \le j \le i} |p_j| \qquad |S^i| \le 1 + \min\{\sum_{0 \le j \le i} |w_j|, M\}$$

至少有一个(0,0)

故，在所有w_j和p_j均为整数的情况下，DKNAP的时间和空间复杂度将为：

$$O(\min\{2^n, n\sum_{1\le i\le n}|p_i|, nM\})$$

5. 序偶集合的一种启发式生成策略

在由S^0生成S^n的过程中，有些序偶无论如何也不会导致问题的最优解——问题的最优解由最大有效序偶给出。

这些序偶最终也不会出现在任何最优决策序列中，故可以及时的舍去，以进一步降低计算量。

例：

$S^2=\{(0,0),(1,2),(2,3),(3,5)\}$

$p_3=2,\ w_3=3$

$(0,\ 0)\rightarrow(2,\ 3)$

$(1,\ 2)\rightarrow(3,\ 5)$

怎样预知背包的可能最好效益？

启发式原理如下：

设L是最优解的估计值，且$f_n(M) \geq L$

记PLEFT(i)= $\sum_{i<j\leq n} p_j$ 即p_{i+1}至n件物品的效益值之和

对正在生成的f_i的序偶(P,W)，若有P＋PLEFT(i)＜L，则(P,W)将不计入S^i中。

例：

$p_3=2$，$w_3=3$ →PLEFT(2) = 2

取L＝6

(0，0)→(2，3)，0+PLEFT(2)=2<6

(1，2)→(3，5)，1+PLEFT(2)=3<6

L的选择：

① 取S^i的最末有效序偶(P,W)的P作为L，$P \leq f_n(M)$。

② 将某些剩余物品的p值＋P作为L。

例6.15 0/1背包问题

设 n=6,$(p_1,p_2,p_3,p_4,p_5,p_6)=(w_1,w_2,w_3,w_4,w_5,w_6)=(100,50,20,10,7,3)$,
M=165

若不使用启发方法, 所得序偶集为:

$S^0=\{0\}$

$S^1=\{0,100\}$

$S^2=\{0,50,100,150\}$

$S^3=\{0,20,50,70,100,120,150\}$

$S^4=\{0,10,20,30,50,60,70,80,100,110,120,130,150,\ 160\}$

$S^5=\{0,7,10,17,20,27,30,37,50,57,60,67,70,77,80,87,100,$
$107,110,117,120,127,130,137,150,157,160\}$ (27个元素)

则, $f_6(165)=163$

这里对每对序偶(P,W)仅用单一量P(或W)表示

启发式规则求解

分析：将物品1,2,4,6装入背包，将占用163的重量并产生163的效益。

故，取期望值L＝163.

按照启发式生成规则，从S^i中删除所有P＋PLEFT(i)＜L的序偶，则有

$PLEFT(0)=p_1+p_2+p_3+p_4+p_5+p_6=190$

$S^0=\{0\} \xrightarrow{S_1} \{100\}$ $PLEFT(1)=p_2+p_3+p_4+p_5+p_6=90$

$S^1=\{100\} \xrightarrow{S_1} \{150\}$ $PLEFT(2)=p_3+p_4+p_5+p_6=40$

$S^2=\{150\} \xrightarrow{S_1} \varnothing$ $PLEFT(3)=p_4+p_5+p_6=20$ $(w_3=20)$

$S^3=\{150\} \xrightarrow{S_1} \{160\}$ $PLEFT(4)=p_5+p_6=10$

$S^4=\{160\} \xrightarrow{S_1} \varnothing$ $PLEFT(5)=p_6=3$ $(w_5=7)$

$S^5=\{160\}$ $PLEFT(6)=0$

$f_6(165)=160+3 = 163$

作业

6.3, 6.6

6.12*, 6.13*