



## 第七章

# *Tree Searching Strategies*

骆吉洲

计算机科学与工程系



## 7.1 *Motivation of Tree Searching*

## 7.2 *Basic Tree Searching Strategies*

## 7.3 *Optimal Tree Searching Strategies*

## 7.4 *Personnel Assignment Problem*

## 7.5 *Traveling Salesperson Optimization Problem*

## 7.6 *The A<sup>\*</sup> Algorithm*



# 参考资料



《计算机算法设计与分析》

- 第5章, 第6章

《网站资料》

- 第七章



## 7.1 *Motivation of Tree Searching*

很多问题可以表示成树。  
于是，这些问题可以使用树  
搜索算法来求解



# 布尔表达式可满足性问题

- 问题的定义

- 输入:  $n$  个布尔变量  $x_1, x_2, \dots, x_n$

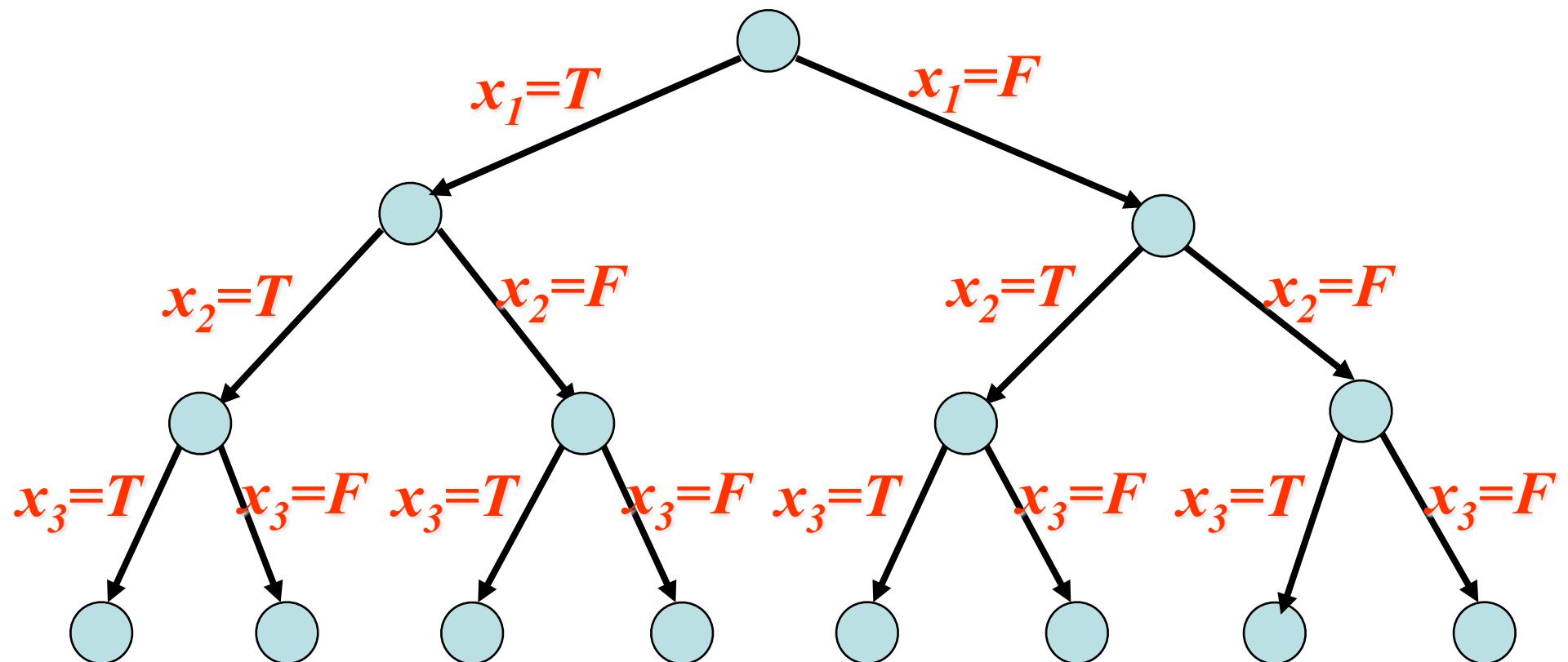
- 关于  $x_1, x_2, \dots, x_n$  的  $k$  个析取布尔式

- 输出: 是否存在一个  $x_1, x_2, \dots, x_n$  的一种赋值使得所有  $k$  个布尔析取式皆为真

- 把问题表示为树

- 通过不断地为赋值集合分类来建立树

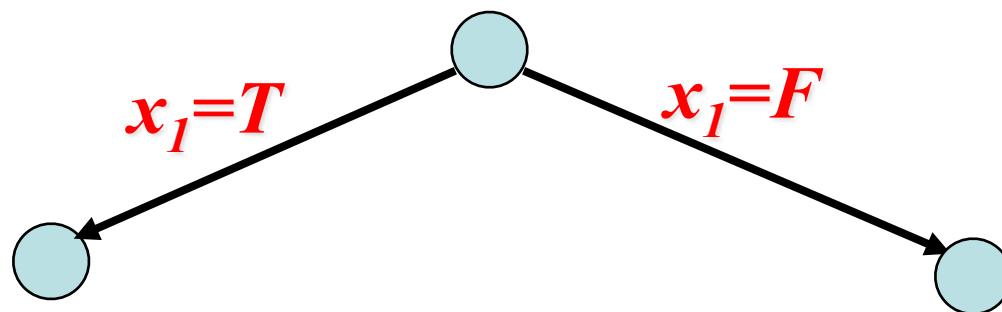
- (以三个变量( $x_1, x_2, x_3$ )为例)





- 求解问题

- 设有布尔式:  $\neg x_1, x_1, x_2 \vee x_5, x_3, \neg x_2$





- 问题的定义

- 输入：具有8个编号小方块的魔方

2	3	
5	1	4
6	8	7

- 输出：移动系列，经过这些移动，魔方达如下状态

1	2	3
8		4
7	6	5



## • 转换为树搜索问题

2	3	
5	1	4
6	8	7

2		3
5	1	4
6	8	7

2	3	4
5	1	
6	8	7



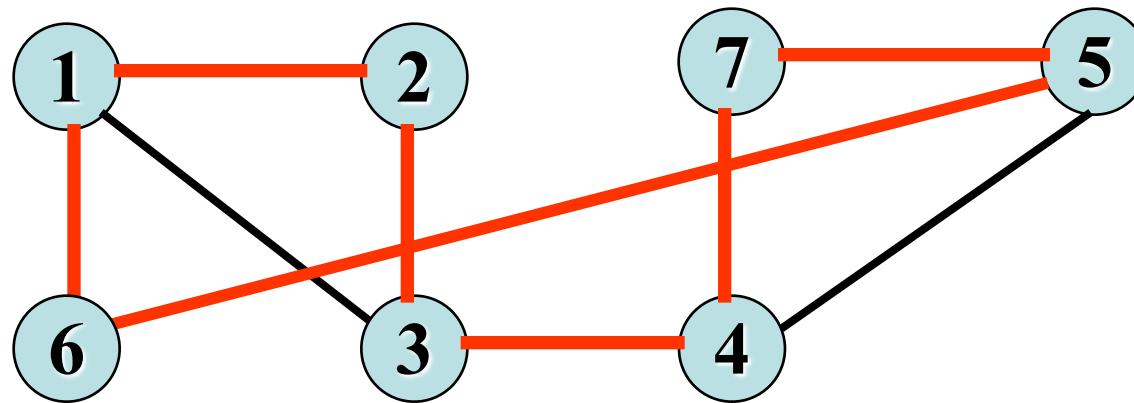
## ● 问题定义

- 输入: 具有n个节点的连通图 $G=(V, E)$
- 输出: G中是否具有Hamiltonian环

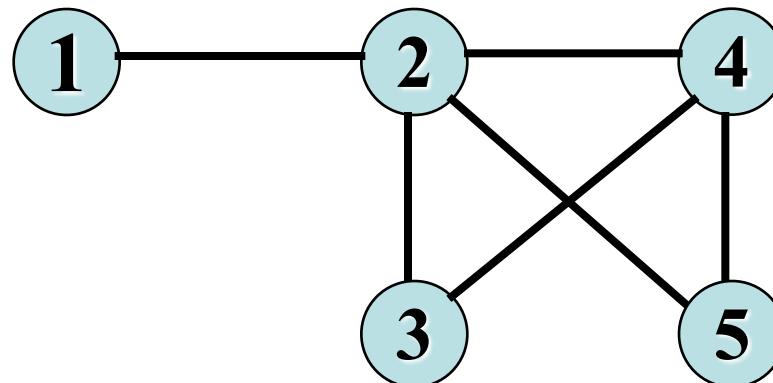
沿着 $G$ 的 $n$ 条边经过每个节点一次，并回到起始节点的环称为 $G$ 的一个Hamiltonian环。



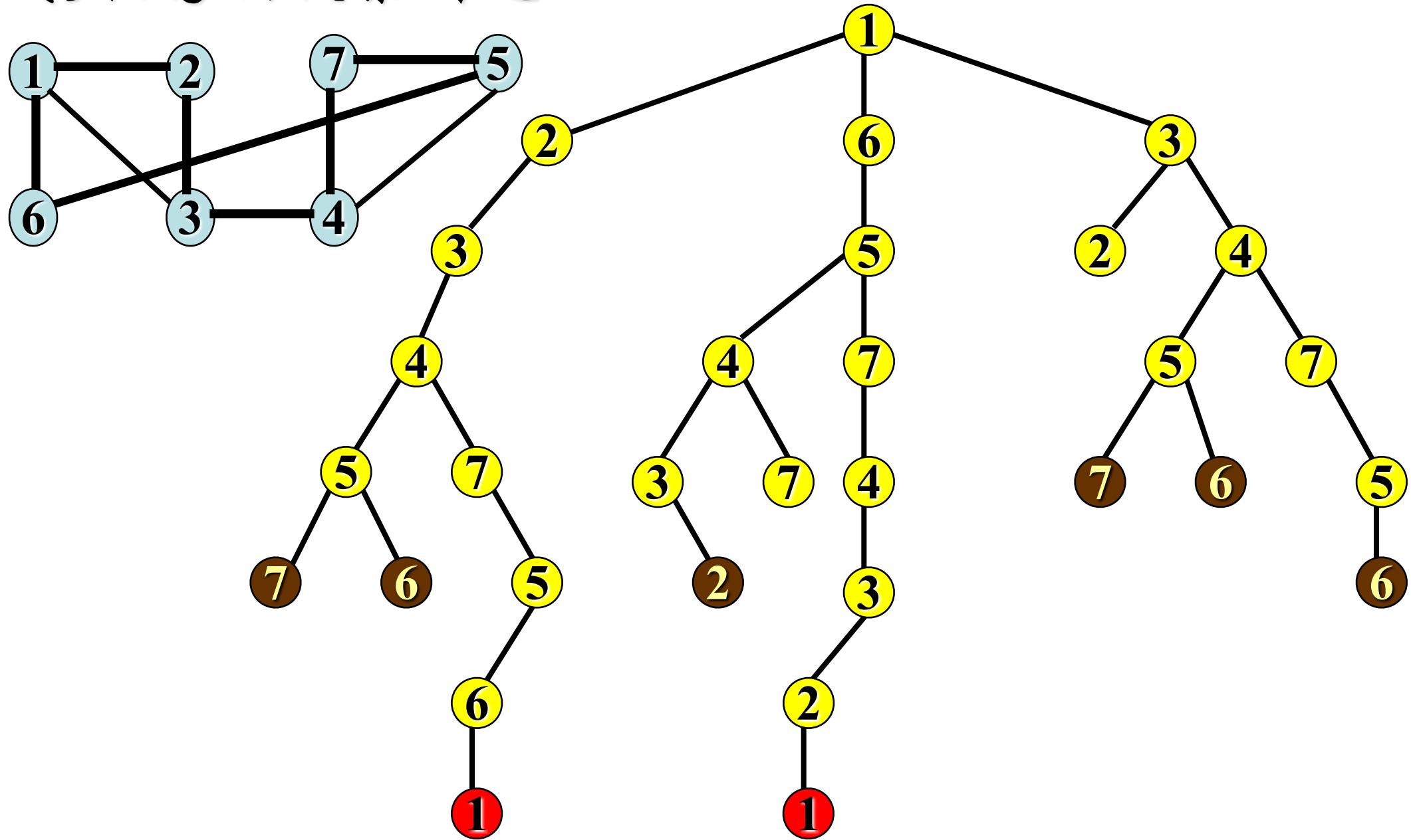
有 Hamiltonian 环图：

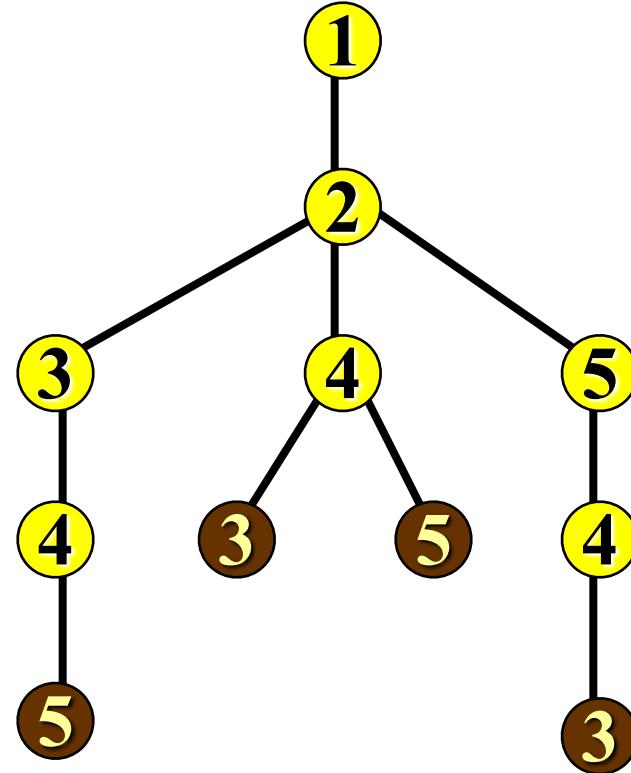
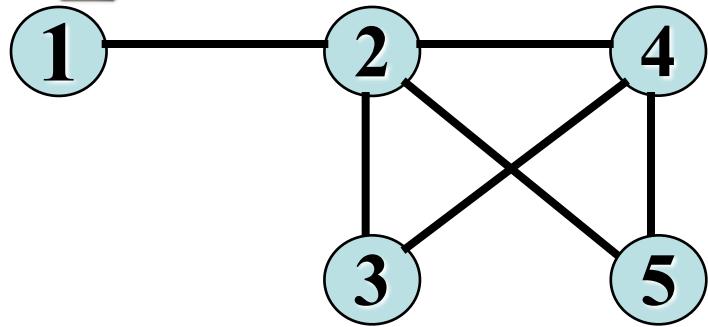


无 Hamiltonian 环图：



• 转换为树搜索问题







## 7.2 Basic Tree Searching Strategies

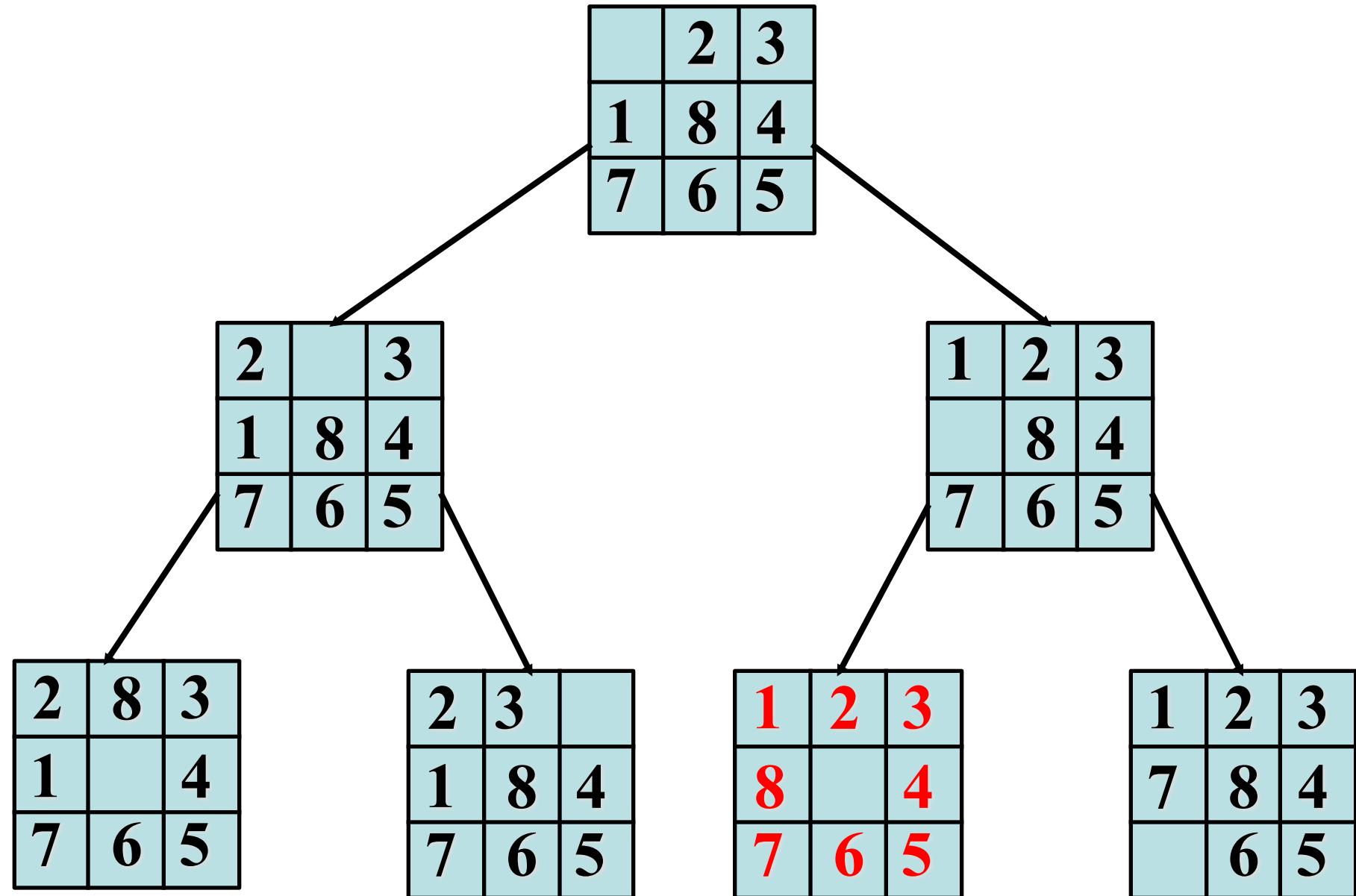
- **Breadth-First Search**
- **Depth-First Search**



## ● 算法

1. 构造由根组成的队列  $Q$ ;
2. If  $Q$  的第一个元素  $x$  是目标节点 Then 停止;
3. 从  $Q$  中删除  $x$ , 把  $x$  的所有子节点加入  $Q$  的末尾;
4. If  $Q$  空 Then 失败 Else goto 2.

## ● 例：求解 8-Puzzle 问题





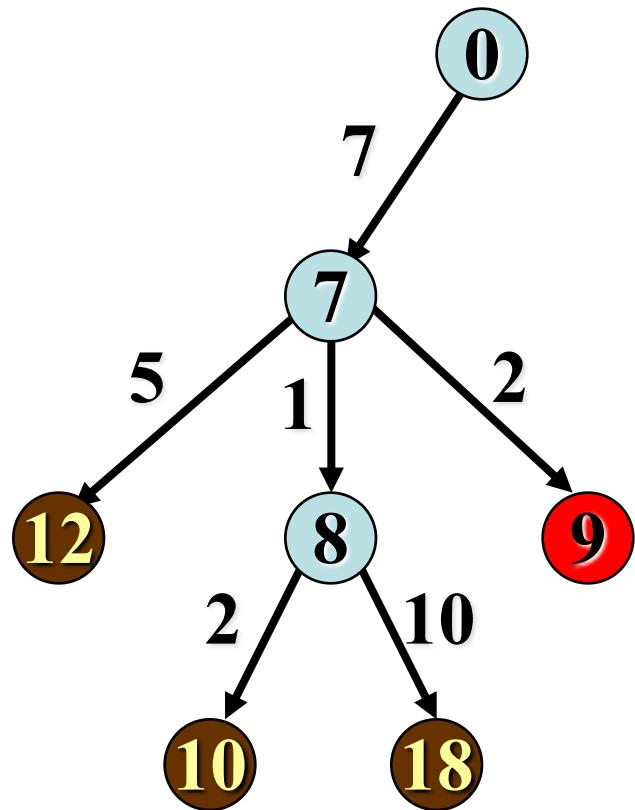
- 算法

1. 构造一个由根构成的单元素栈 $S$ ;
2. If  $Top(S)$ 是目标节点 Then 停止;
3.  $Pop(S)$ , 把 $Top(S)$ 的所有子节点压入栈顶;
4. If  $S$ 空 Then 失败 Else goto 2.

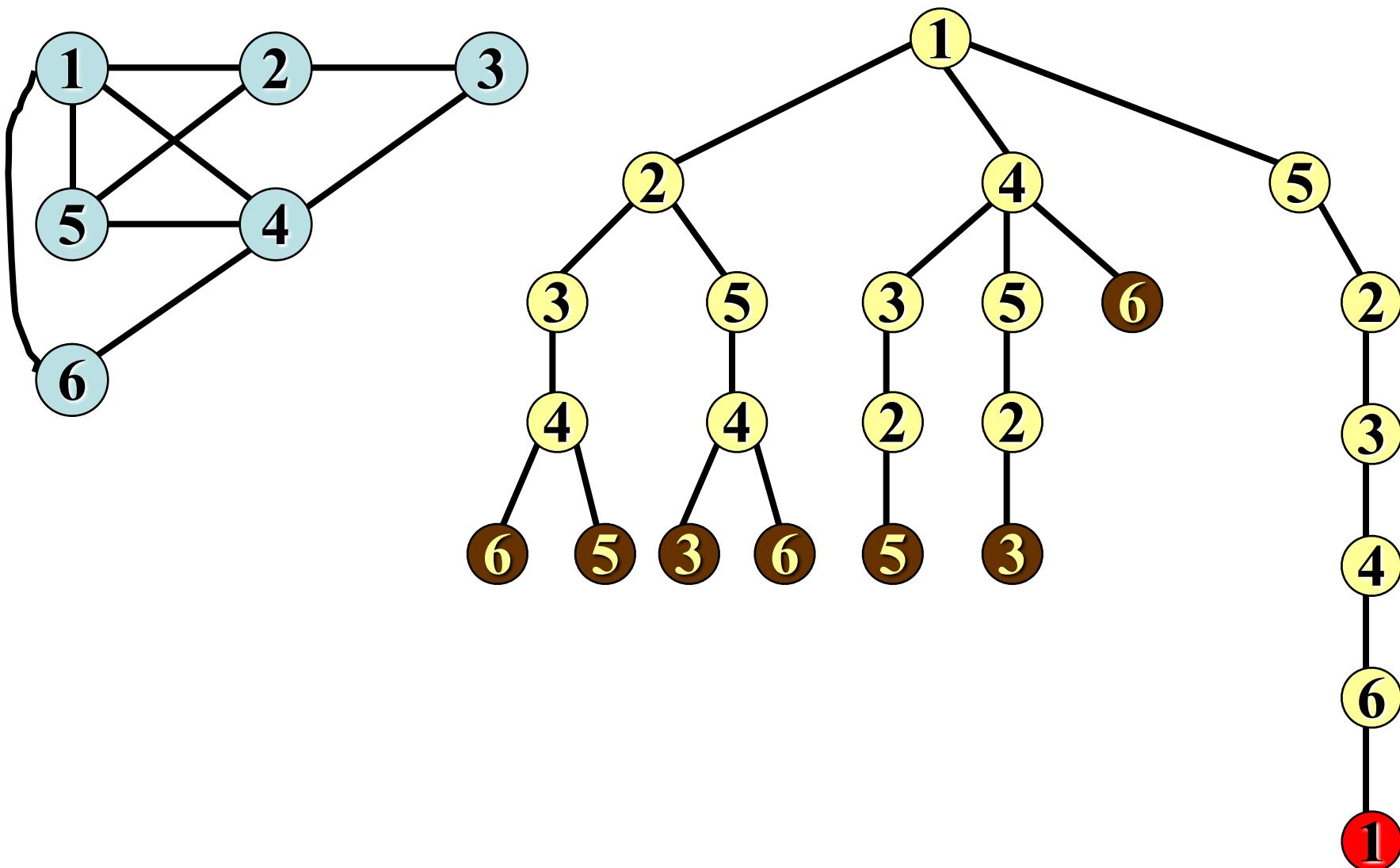
● 例1. 求解子集合和问题

输入:  $S=\{7, 5, 1, 2, 10\}$

输出: 是否存在  $S' \subseteq S$ , 使得  $\text{Sum}(S')=9$



• 例2. 求解 Hamiltonian 环问题





## 7.3 Optimal Tree Searching Strategies

- **Hill Climbing**
- **Best-First Search Strategy**
- **Branch-and-Bound Strategy**



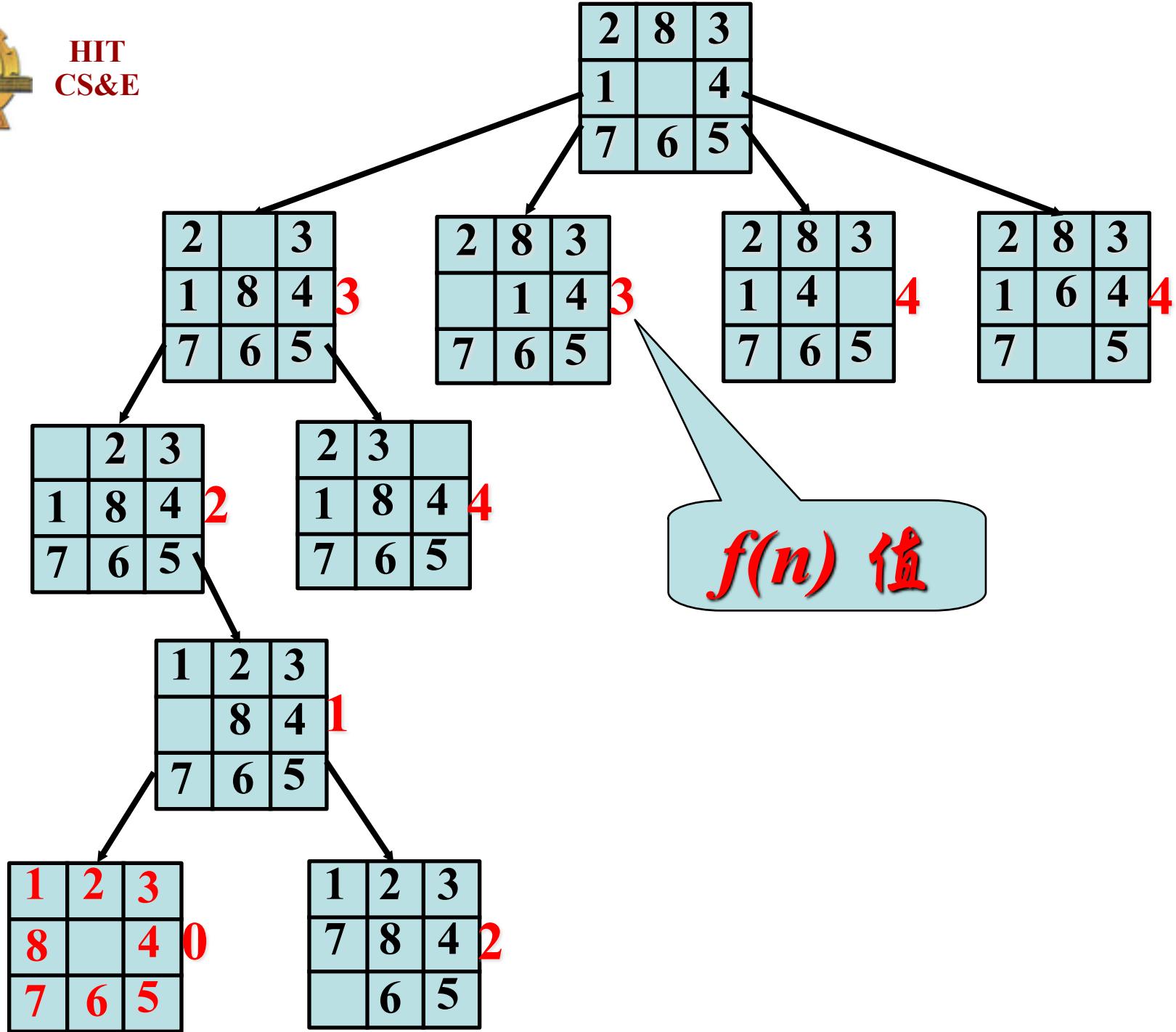
- 基本思想

- 在深度优先搜索过程中，我们经常遇到多个节点可以扩展的情况，首先扩展哪个？
- 爬山策略使用贪心方法确定搜索的方向，是优化的深度优先搜索策略
- 爬山策略使用启发式测度来排序节点扩展的顺序



- 用 8-Puzzle 问题来说明爬山策略的思想
  - 启发式测度函数:  $f(n)=W(n)$ ,  $W(n)$  是节点  $n$  中处于错误位置的方块数.
  - 例如, 如果节点  $n$  如下, 则  $f(n)=3$ , 因为方块 1、2、8 处于错误位置.

2	8	3
1		4
7	6	5



- Hill Climbing 算法

1. 构造由根组成的单元素栈  $S$ ;
2. If  $\text{Top}(S)$  是目标节点 Then 停止;
3.  $\text{Pop}(S)$ ;
4.  $S$  的子节点按照其启发测度由大到小的顺序压入  $S$ ;
5. If  $S$  空 Then 失败 Else goto 2.



# Best-First Search Strategy

- 基本思想

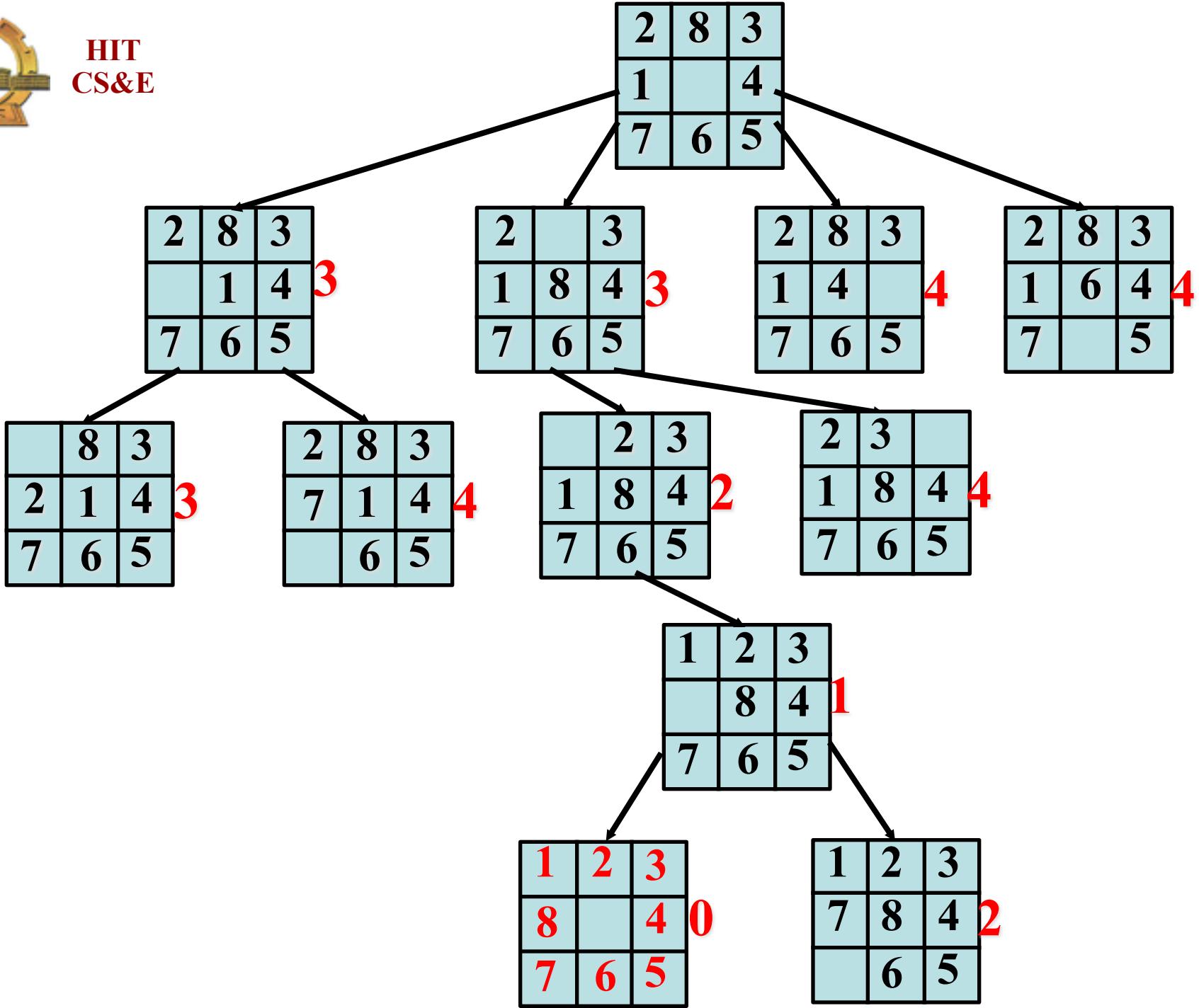
- 结合深度优先和广度优先的优点
- 根据一个评价函数，在目前产生的所有节点中选择具有最小评价函数值的节点进行扩展.
- 具有全局优化观念，而爬山策略仅有局部优化观念.



- BesT-First Search 算法

1. 使用评价函数构造一个堆  $H$ , 首先构造由根组成的单元素堆;
2. If  $H$  的根  $r$  是目标节点 Then 停止;
3. 从  $H$  中删除  $r$ , 把  $r$  的子节点插入  $H$ ;
4. If  $H$  空 Then 失败 Else goto 2.

- 8-Puzzle 问题实例





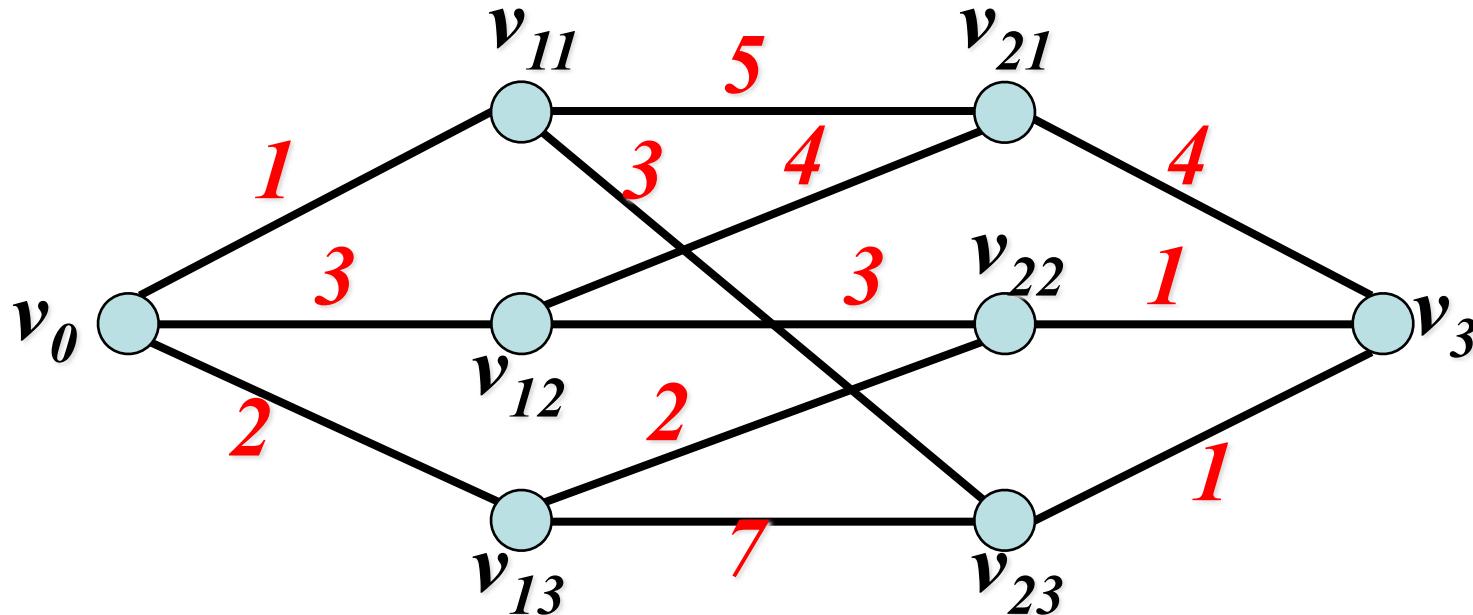
# Branch-and-Bound Strategy

- 基本思想
  - 上述方法很难用于求解优化问题
  - 分支界限策略可以有效地求解组合优化问题
  - 发现优化解的一个界限
  - 缩小解空间, 提高求解的效率
- 举例说明分支界限策略的原理



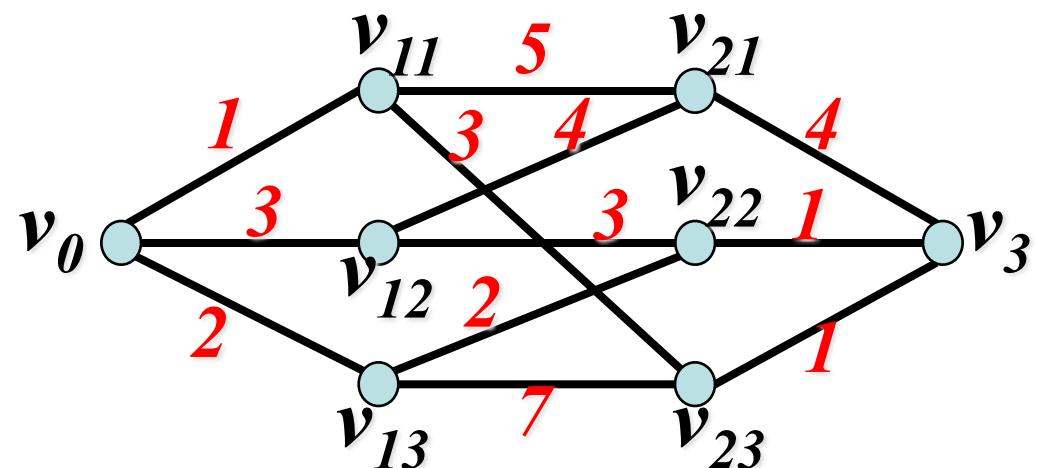
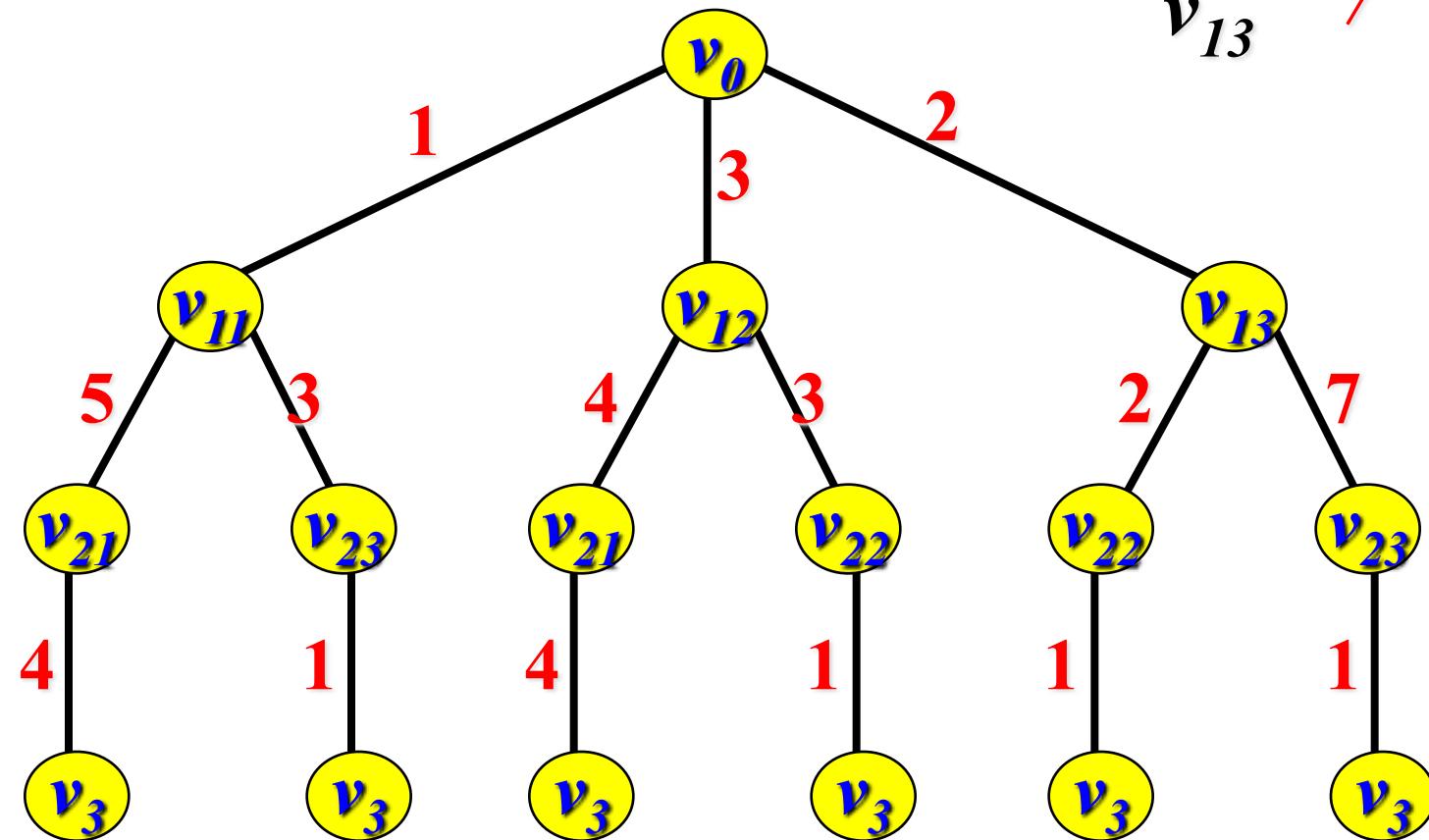
- 多阶段图搜索问题

- 输入: 多阶段图

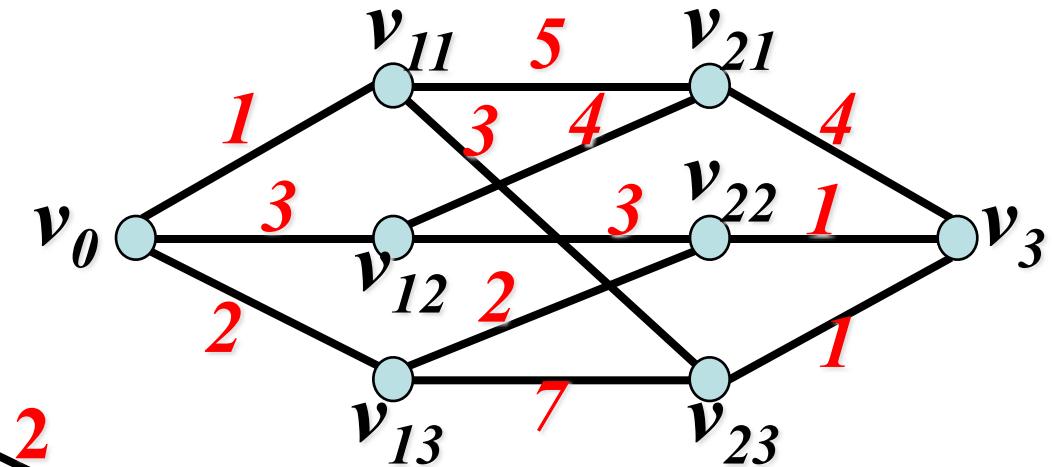
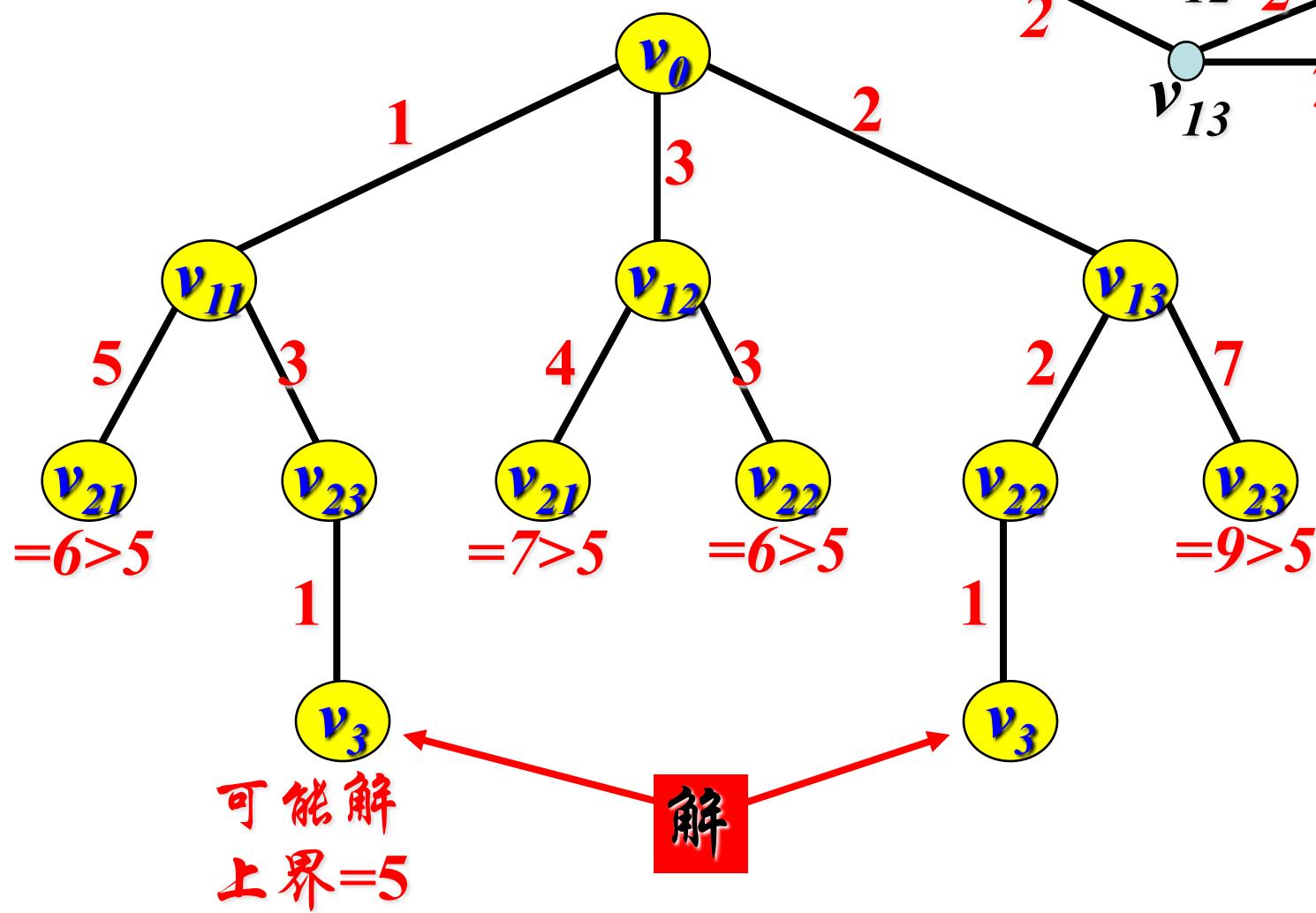


- 输出: 从 $v_0$ 到 $v_3$ 的最短路径

## • 问题的树表示



- 使用爬山策略进行分枝界限搜索





- 分支界限策略的原理

- 产生分支的机制(使用前面的任意一种策略)
- 产生一个界限(可以通过发现可能解)
- 进行分支界限搜索, 即剪除不可能产生优化解的分支.



## 7.4 Personnel Assignment Problem

- 问题的定义
- 转换为树搜索问题
- 求解问题的分支界限搜索算法



## ● 输入

- 人的集合  $P = \{P_1, P_2, \dots, P_n\}$ ,  $P_1 < P_2 < \dots < P_n$

人与人的关系： $P_i \rightarrow P_j$  表示  $P_i$  可以帮助  $P_j$ ，或  $P_i$  是  $P_j$  的上级。

例. 给定  $P = \{P_1, P_2, P_3\}$ ,  $J = \{J_1, J_2, J_3\}$ ,  $J_1 \leq J_3$ ,  $J_2 \leq J_3$ .

$P_1 \rightarrow J_1$ 、 $P_2 \rightarrow J_2$ 、 $P_3 \rightarrow J_3$  是可能的解.

$P_1 \rightarrow J_1$ 、 $P_2 \rightarrow J_3$ 、 $P_3 \rightarrow J_2$  不可能是解.

人与人的关系： $P_i \rightarrow P_j$  表示  $P_i$  可以帮助  $P_j$ ，或  $P_i$  是  $P_j$  的上级。

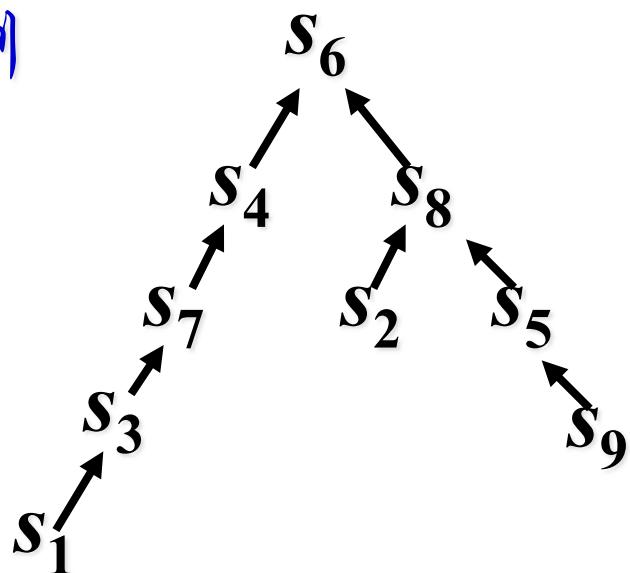
- 如果  $f(P_i) \leq f(P_j)$ , 则  $P_i \leq P_j$



## • 拓朴排序

- 输入: 偏序集合  $(S, \leq)$
- 输出:  $S$  的拓朴序列是  $\langle s_1, s_2, \dots, s_n \rangle$ ,  
满足: 如果  $s_i \leq s_j$ , 则  $s_i$  排在  $s_j$  的前面.

- 例



拓朴排序:

$s_1 \ s_3 \ s_7 \ s_4 \ s_9 \ s_5 \ s_2 \ s_8 \ s_6$

## ● 问题的解空间

命题1.  $P_1 \rightarrow J_{k1}$ 、 $P_2 \rightarrow J_{k2}$ 、...、 $P_n \rightarrow J_{kn}$ 是一个可能解，当且仅当  $J_{k1}$ 、 $J_{k2}$ 、...、 $J_{kn}$  必是一个拓扑排序的序列。

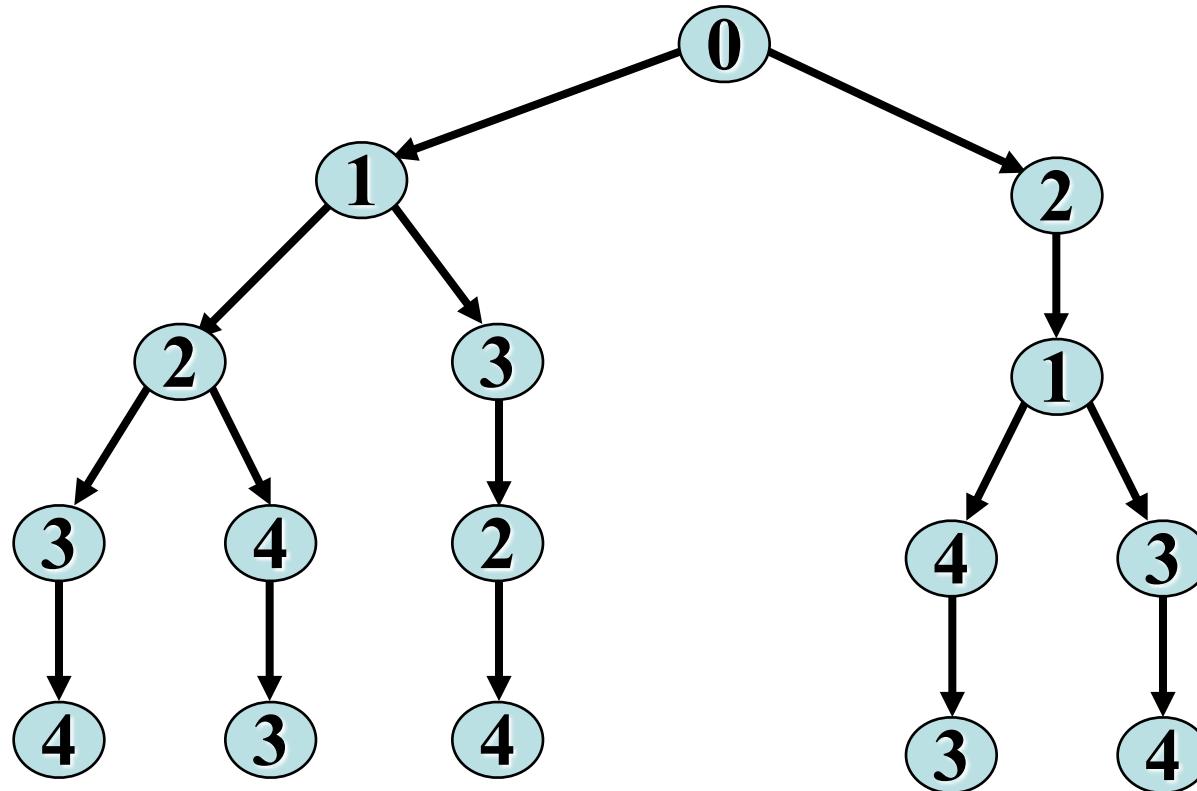
问题的解空间是所有拓扑排序的序列集合，  
每个序列对于一个可能的解

则  $(J_1, J_2, J_3, J_4)$ 、 $(J_1, J_2, J_4, J_3)$ 、 $(J_1, J_3, J_2, J_4)$ 、  
 $(J_2, J_1, J_3, J_4)$ 、 $(J_2, J_1, J_4, J_3)$  是拓扑排序序列

$(J_1, J_2, J_4, J_3)$  对应于  $P_1 \rightarrow J_1$ 、 $P_2 \rightarrow J_2$ 、 $P_3 \rightarrow J_4$ 、 $P_4 \rightarrow J_3$



- 问题的树表示(即用树表示所有拓扑排序序列)





- 拓朴序列树的生成算法

输入：偏序集合 $S$ , 树根 $root$ .

输出：由 $S$ 的所有拓朴排序序列构成的树.

1. 生成树根 $root$ ;

2. 选择偏序集中没有前序元素的所有元素，作为 $root$ 的子节点；

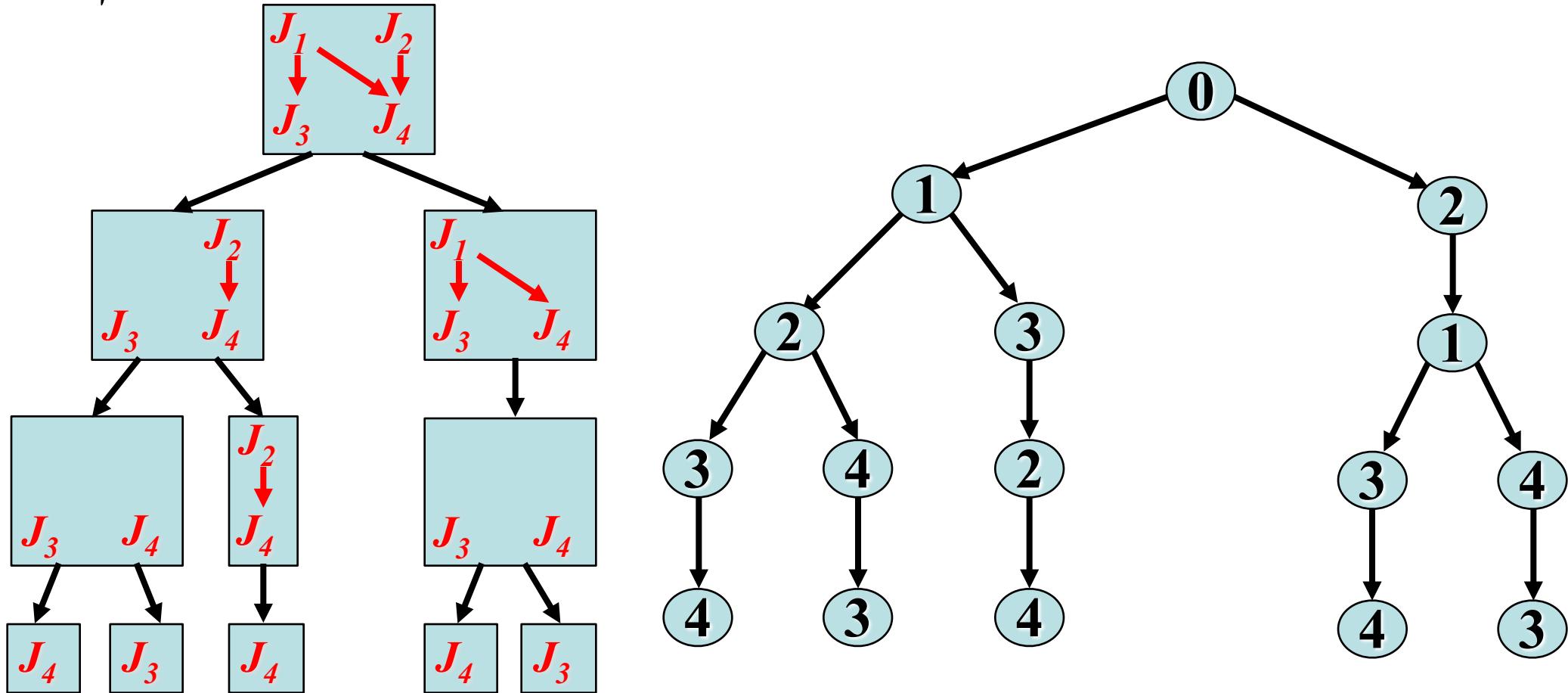
3. For  $root$ 的每个字节点 $v$  Do

4.  $S=S-\{v\}$ ;

5. 把 $v$ 作为根，递归地处理 $S$ .



• 例





- 计算解的代价的下界

**命题2.** 把代价矩阵某行(列)的各元素减去同一个数, 不影响优化解的求解.

- 代价矩阵的每行(列)减去同一个数(该行或列的最小数), 使得每行和每列至少有一个零, 其余各元素非负.
- 每行(列)减去的数的和即为解的下界.



例.

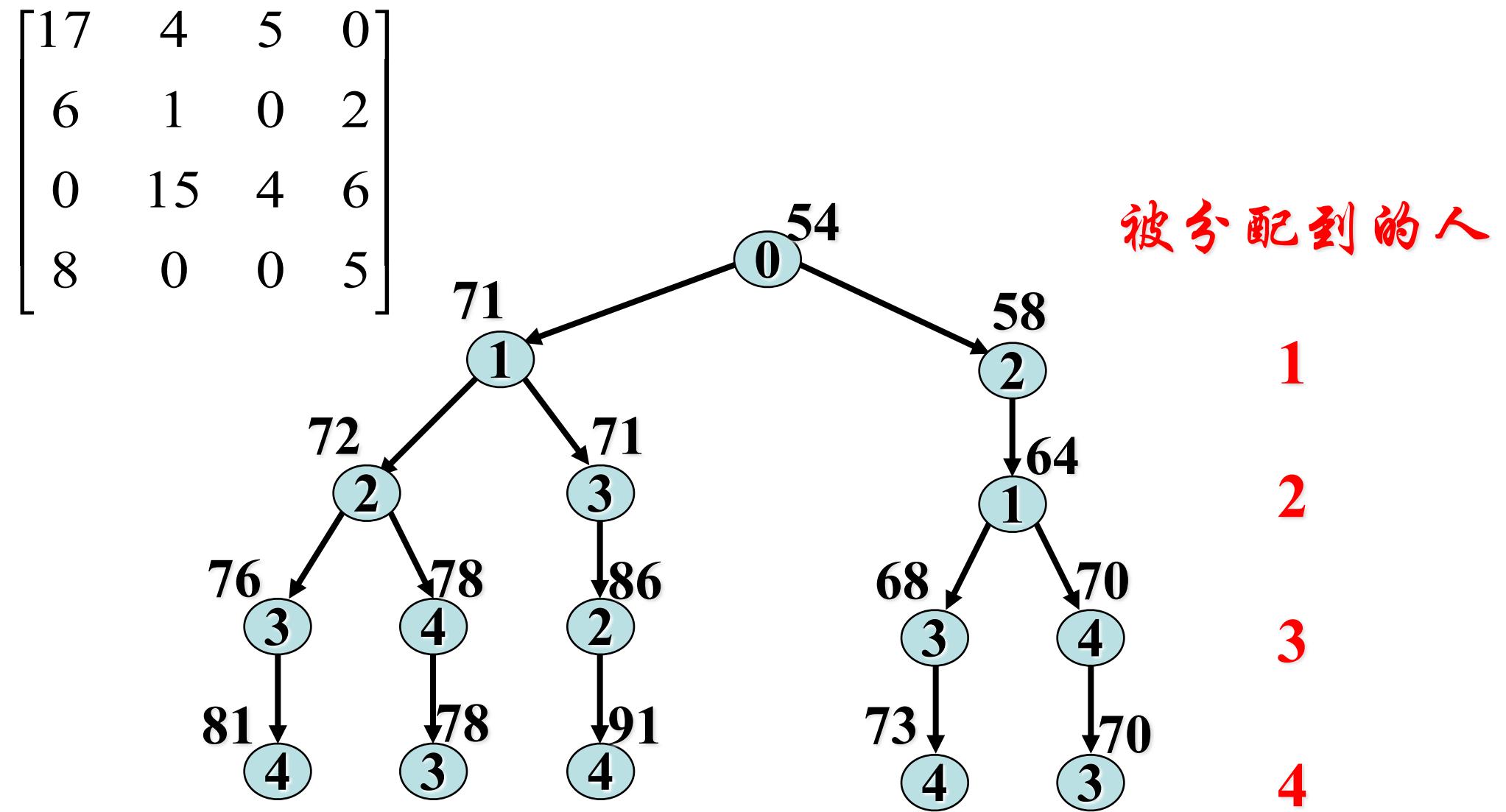
$$\begin{array}{l} J_1 \quad J_2 \quad J_3 \quad J_4 \\ P_1 \left[ \begin{array}{cccc} 29 & 19 & 17 & 12 \end{array} \right] -12 \\ P_2 \left[ \begin{array}{cccc} 32 & 30 & 26 & 28 \end{array} \right] -26 \\ P_3 \left[ \begin{array}{cccc} 3 & 21 & 7 & 9 \end{array} \right] -3 \\ P_4 \left[ \begin{array}{cccc} 18 & 13 & 10 & 15 \end{array} \right] -10 \\ \qquad \qquad \qquad -3 \end{array}$$



$$\left[ \begin{array}{cccc} 17 & 4 & 5 & 0 \\ 6 & 1 & 0 & 2 \\ 0 & 15 & 4 & 6 \\ 8 & 0 & 0 & 5 \end{array} \right]$$

解代价下界=12+26+3+10+3=54

# 解空间的加权树表示

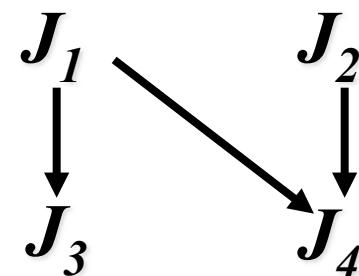


## • 分支界限搜索(使用爬山法)算法

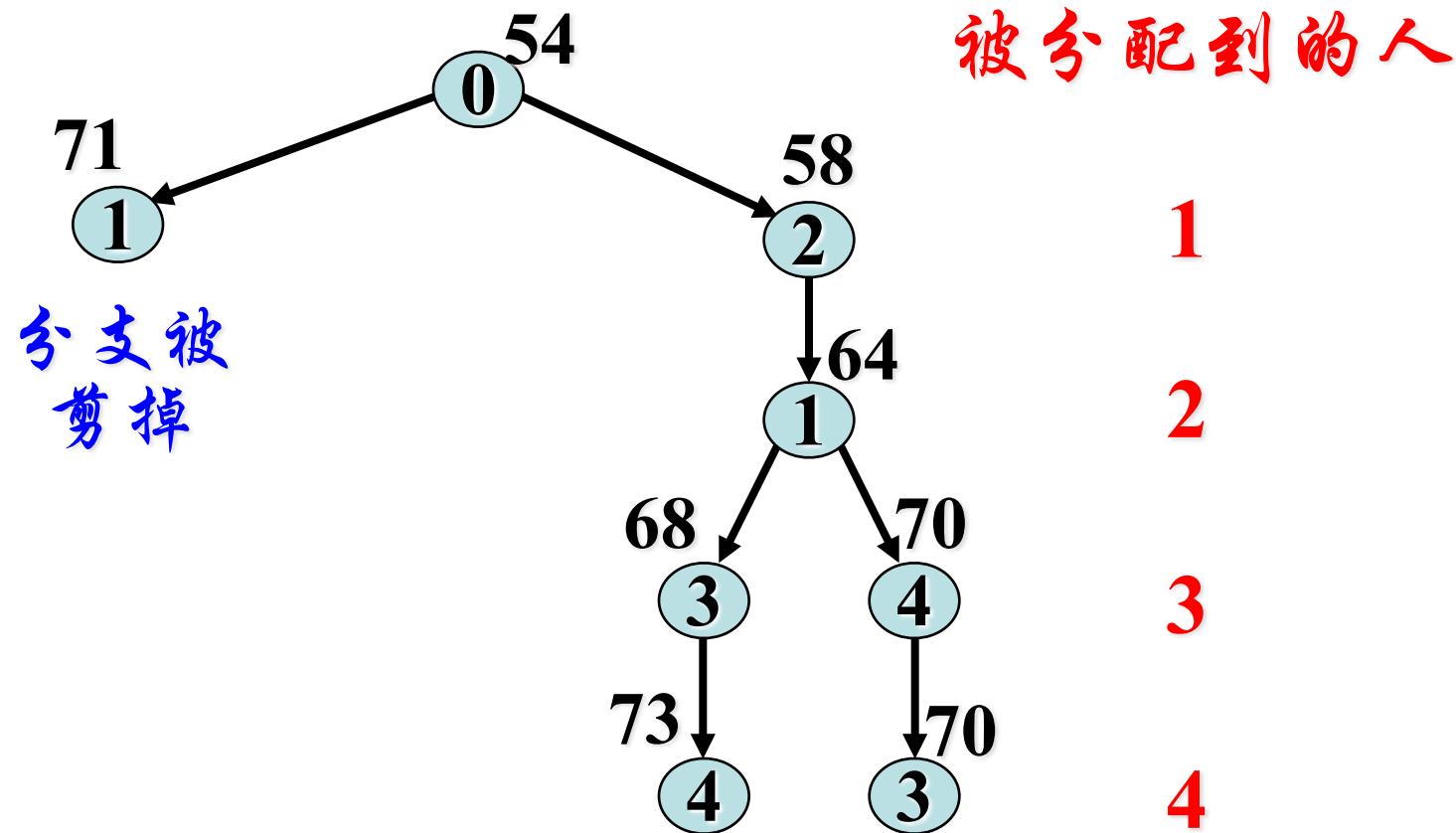
1. 建立根节点，其权值为解代价下界；
2. 使用爬山法，类似于拓朴排序序列树生成算法求解问题，每产生一个节点，其权值为加工后的代价矩阵对应元素加其父节点权值；
3. 一旦发现一个可能解，将其代价作为界限，循环地进行分支界限搜索：剪掉不能导致优化解的子解，使用爬山法继续扩展新增节点，直至发现优化解。

● 例

$$\{P_1, P_2, P_3, P_4\}$$



$$\begin{bmatrix} 17 & 4 & 5 & 0 \\ 6 & 1 & 0 & 2 \\ 0 & 15 & 4 & 6 \\ 8 & 0 & 0 & 5 \end{bmatrix}$$





## 7.5 *Traveling Salesperson Optimization Problem*

- 问题的定义
- 转换为树搜索问题
- 分支界限搜索算法



输入：无向连通图  $G=(V, E)$ ，  
每个节点都没有到自身的边，  
每对节点之间都有一条非负加权边。

输出：一条由任意一个节点开始  
经过每个节点一次  
最后返回开始节点的路径，  
该路径的代价(即权值之和)最小。



- 所有解集合作为树根，其权值由代价矩阵使用上节方法计算；
- 用爬山法递归地划分解空间，得到二叉树
- 划分过程：
  - 如下选择图上满足下列条件的边 $(i, j)$ 
    - $\text{Cost}(i, l) = \max\{\text{Cost}(k, l) \mid \forall k \in V\}$
    - $\text{Cost}(i, j) = 0$
  - 使右子树代价下界增加最大
  - 所有包含 $(i, j)$ 的解集合作为左子树
  - 所有不包含 $(i, j)$ 的解集合作为右子树
  - 计算出左右子树的代价下界



- 在上述二叉树建立算法中增加如下策略：
  - 发现优化解的上界 $\alpha$ ;
  - 如果一个子节点的代价下界超过 $\alpha$ , 则终止该节点的扩展.
- 下边我们用一个例子来说明算法

● 构造根节点，设代价矩阵如下

	$j = 1$	$2$	$3$	$4$	$5$	$6$	$7$	
$i=1$	$\infty$	3	93	13	33	9	57	-3
2	4	$\infty$	77	42	21	16	34	-4
3	45	17	$\infty$	36	16	28	25	-16
4	39	90	80	$\infty$	56	7	91	-7
5	28	46	88	33	$\infty$	25	57	-25
6	3	88	18	46	92	$\infty$	7	-3
7	44	26	33	27	84	39	$\infty$	-26
				-7	-1			-4

- 根节点为所有解的集合
- 计算根节点的代价下界

➤ 得到如下根节点及其代价下界

所有解的集合

L.B=96

➤ 变换后的代价矩阵为

		$j = 1$	2	3	4	5	6	7	
		$i=1$	$\infty$	0	83	9	30	6	50
		2	0	$\infty$	66	37	17	12	26
		3	29	1	$\infty$	19	0	12	5
		4	32	83	66	$\infty$	49	0	80
		5	3	21	56	7	$\infty$	0	28
		6	0	85	8	42	89	$\infty$	0
		7	18	0	0	0	58	13	$\infty$

- 构造根节点的两个子节点

- 选择使子节点代价下界增加最大的划分边(4, 6)
- 建立根节点的子节点：

- ✓ 左子节点为包括边(4, 6)的所有解集合
- ✓ 右子节点为不包括边(4, 6)的所有解集合

$\infty$	0	83	9	30	6	50
0	$\infty$	66	37	17	12	26
29	1	$\infty$	19	0	12	5
32	83	66	$\infty$	49	0	80
3	21	56	7	$\infty$	0	28
0	85	8	42	89	$\infty$	0
18	0	0	0	58	13	$\infty$

所有解的集合 L.B=96

包括边(4, 6)  
的所有解集合

不包括边(4, 6)  
的所有解集合

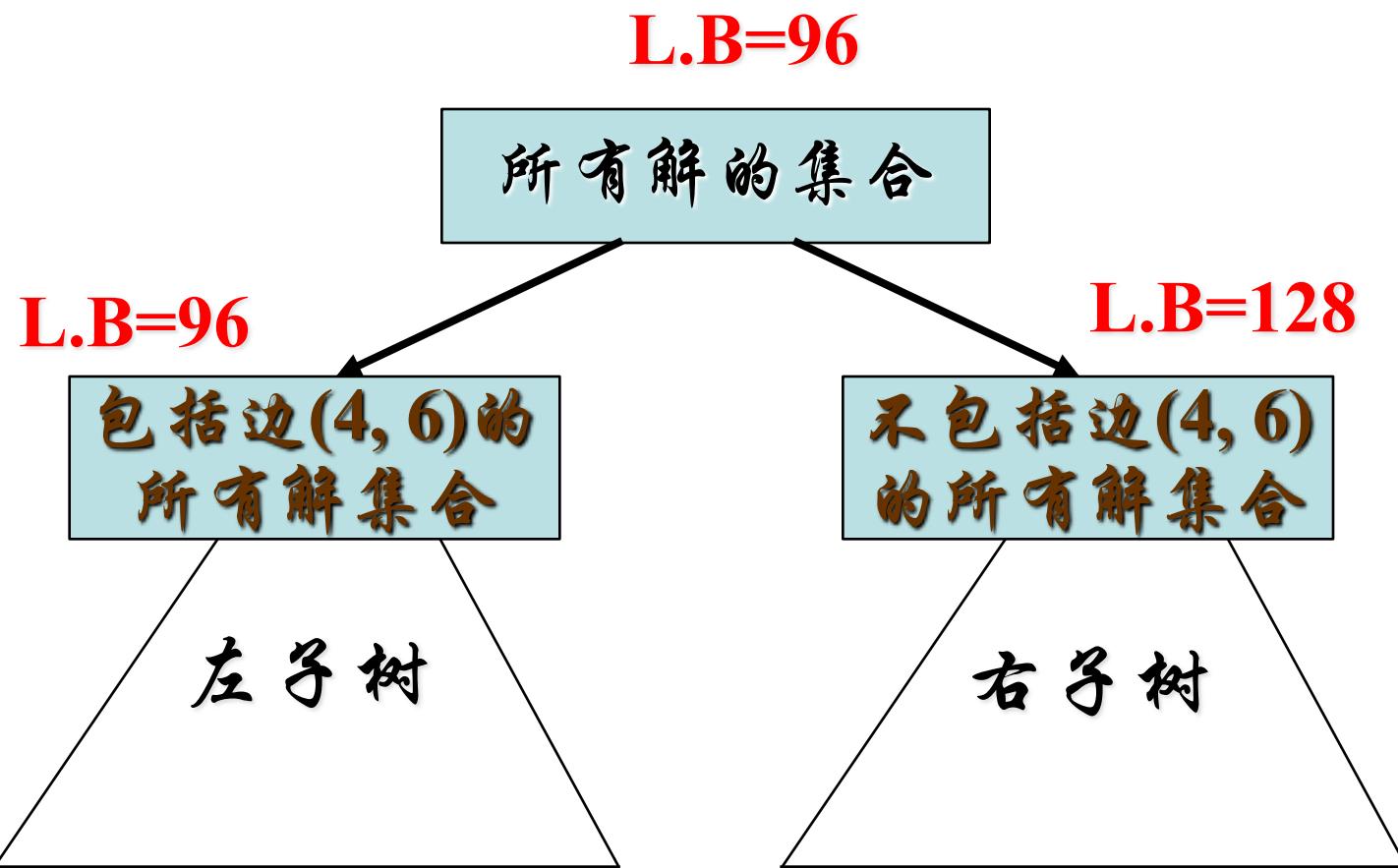


## ➤ 计算左右子节点的代价下界

- ✓  $(4, 6)$  的代价为 0, 所以左节点代价下界仍为 **96**.
- ✓ 我们来计算右节点的代价下界:
  - ◆ 如果一个解不包含  $(4, 6)$ , 它必包含一条从 4 出发的边和进入节点 6 的边.
  - ◆ 由变换后的代价矩阵可知, 具有最小代价由 4 出发的边为  $(4, 1)$ , 代价为 32.
  - ◆ 由变换后的代价矩阵可知, 具有最小代价进入 6 的边为  $(5, 6)$ , 代价为 0.
  - ◆ 于是, 右节点代价下界为:  **$96+32+0=128$** .



➤ 目前的树为



## ● 递归地构造左右子树

### ➤ 构造左子树根对应的代价矩阵

✓ 左子节点为包括边(4, 6)的所有解集合, 所以矩阵的第4行和第6列应该被删除

✓ 由于边(4, 6)被使用, 边(6, 4)不能再使用, 所以代价矩阵的元素 $C[6, 4]$ 应该设置为 $\infty$ .

✓ 结果矩阵如下

$j = 1$	2	3	4	5	7	
$i = 1$	$\infty$	0	83	9	30	50
2	0	$\infty$	66	37	17	26
3	29	1	$\infty$	19	0	5
5	3	21	56	7	$\infty$	28
6	0	85	8	$\infty$	89	0
7	18	0	0	0	58	$\infty$

## > 计算左子树根的代价下界

✓ 矩阵的第5行不包含0

✓ 第5行元素减3, 左子树根代价下界为:  $96+3=99$

✓ 结果矩阵如下

$j = 1$	2	3	4	5	7	
$i=1$	$\infty$	0	83	9	30	50
2	0	$\infty$	66	37	17	26
3	29	1	$\infty$	19	0	5
5	0	18	53	4	$\infty$	25
6	0	85	8	$\infty$	89	0
7	18	0	0	0	58	$\infty$

## ➤ 构造右子树根对应的代价矩阵

- ✓ 右子节点为不包括边(4, 6)的所有解集合, 只需要把  $C[4, 6]$  设置为  $\infty$
- ✓ 结果矩阵如下

		$j = 1$	2	3	4	5	6	7
$i=1$	$\infty$	0	83	9	30	6	50	
2	0	$\infty$	66	37	17	12	26	
3	29	1	$\infty$	19	0	12	5	
4	32	83	66	$\infty$	49	$\infty$	80	
5	3	21	56	7	$\infty$	0	28	
6	0	85	8	42	89	$\infty$	0	
7	18	0	0	0	58	13	$\infty$	

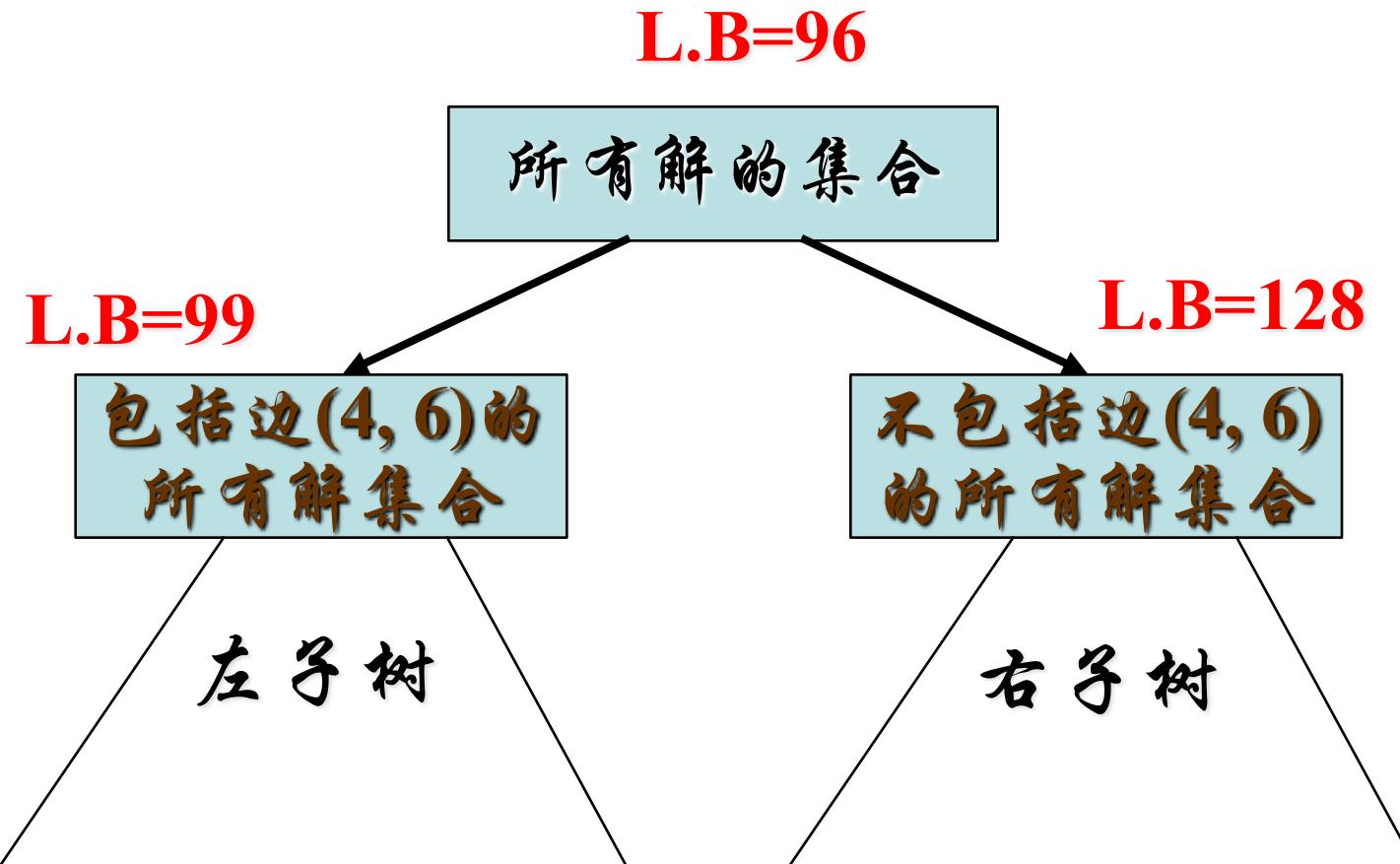
## ➤ 计算右子树根的代价下界

- ✓ 矩阵的第4行不包含0
- ✓ 第4行元素减32, 右子树根代价下界为: 128
- ✓ 结果矩阵如下

	$j = 1$	2	3	4	5	6	7
$i=1$	$\infty$	0	83	9	30	6	50
2	0	$\infty$	66	37	17	12	26
3	29	1	$\infty$	19	0	12	5
4	0	51	34	$\infty$	17	$\infty$	48
5	3	21	56	7	$\infty$	0	28
6	0	85	8	42	89	$\infty$	0
7	18	0	0	0	58	13	$\infty$



➤ 目前的树为





## ➤ 使用爬山策略扩展左子树根

- ✓ 这样边使子节点代价下界增加最大的划分边(3, 5)
- ✓ 左子节点为包括边(3, 5)的所有解集合
- ✓ 右子节点为不包括边(3, 5)的所有解集合
- ✓ 计算左、右子节点的代价下界: 99和117

## ➤ 目前树扩展为:



L.B=96

所有解集合

L.B=99

包括边(4, 6)

由于右子树代价  
下界=160>126  
停止扩展

L.B=99

包括边(3, 5)

不包括边(5, 5)

L.B=117

L.B=112

包括边(2, 1)

不包括边(2, 1)

L.B=125

L.B=126

包括边(1, 4)

不包括边(1, 4)

L.B=153

L.B=126

包括边(6, 7)

不包

解:

1-4-6-7-3-5-2-1

优化解代  
价的上界

L.B=126

包括边(5, 2)

不包

代价: 126

L.B=126

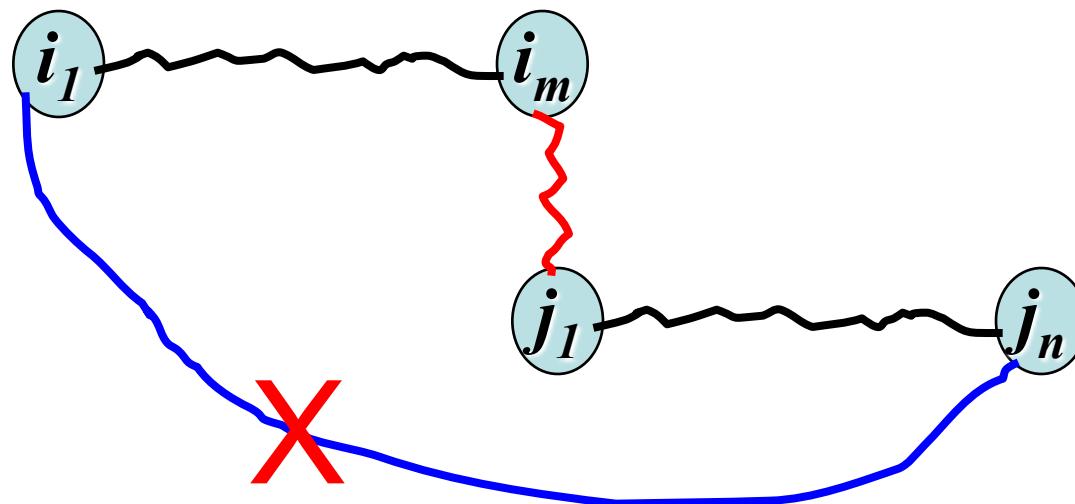
包括边(7, 3)

不包括边(7, 3)

空集合

## 注意

如果  $i_1-i_2-\dots-i_m$  和  $j_1-j_2-\dots-j_m$  已被包含在一个正在构造的路径中,  $(i_m, j_1)$  被加入, 则必须避免  $j_n$  到  $i_1$  的路径被加入. 否则出现环.





## 7.6 *The A<sup>★</sup> Algorithm*

- A<sup>\*</sup>算法的基本思想
- A<sup>\*</sup>算法的规则
- 应用A<sup>\*</sup>算法求解最短路径问题



- A\*算法与分支界限策略的比较
  - 分支界限策略是为了剪掉不能达到优化解的分支
  - 分支界限策略的关键是“界限”
  - A\*算法的核心是告诉我们在某些情况下，我们得到的解一定是优化解，于是算法可以停止
  - A\*算法试图尽早地发现优化解
  - A\*算法经常使用 Best-first 策略求解优化问题

- A\*算法关键——代价函数

- 对于任意节点 $n$

- $g(n)$ =从树根到 $n$ 的代价

- $h^*(n)$ =从 $n$ 到目标节点的优化路径的代价

- $f^*(n)=g(n)+h^*(n)$ 是节点 $n$ 的代价

- What is the value of  $h^*(n)$ ?

- 不知道!

- 于是,  $f^*(n)$ 也不知道

- 估计 $h^*(n)$

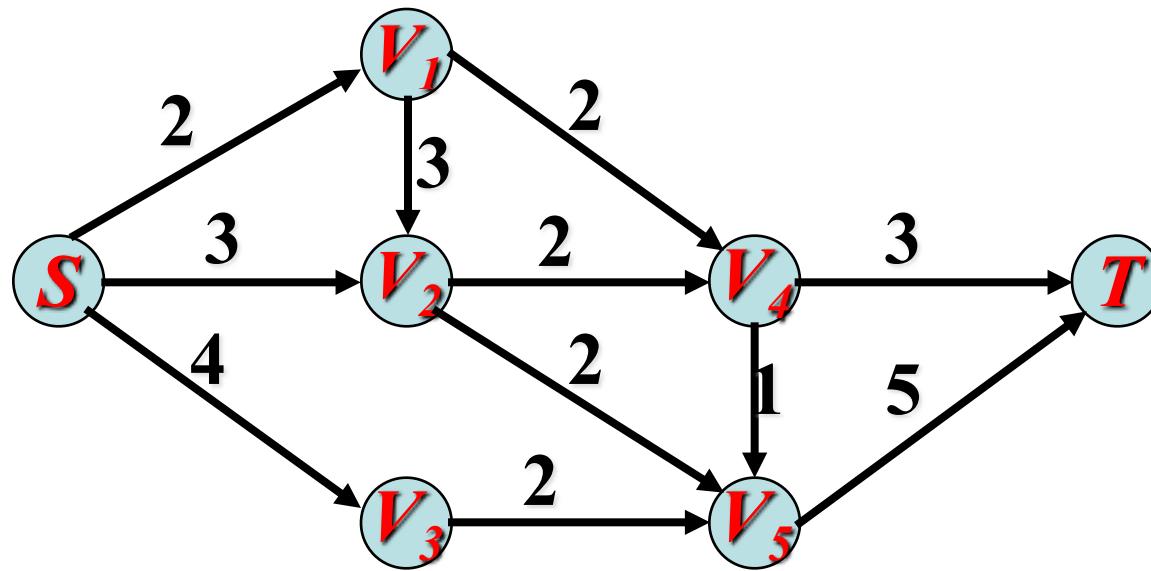
- 使用任何方法去估计 $h^*(n)$ , 用 $h(n)$ 表示 $h^*(n)$ 的估计

- $h(n) \leq h^*(n)$ 总为真

- $f(n)=g(n)+h(n) \leq g(n)+h^*(n)=f^*(n)$ 定义为 $n$ 的代价

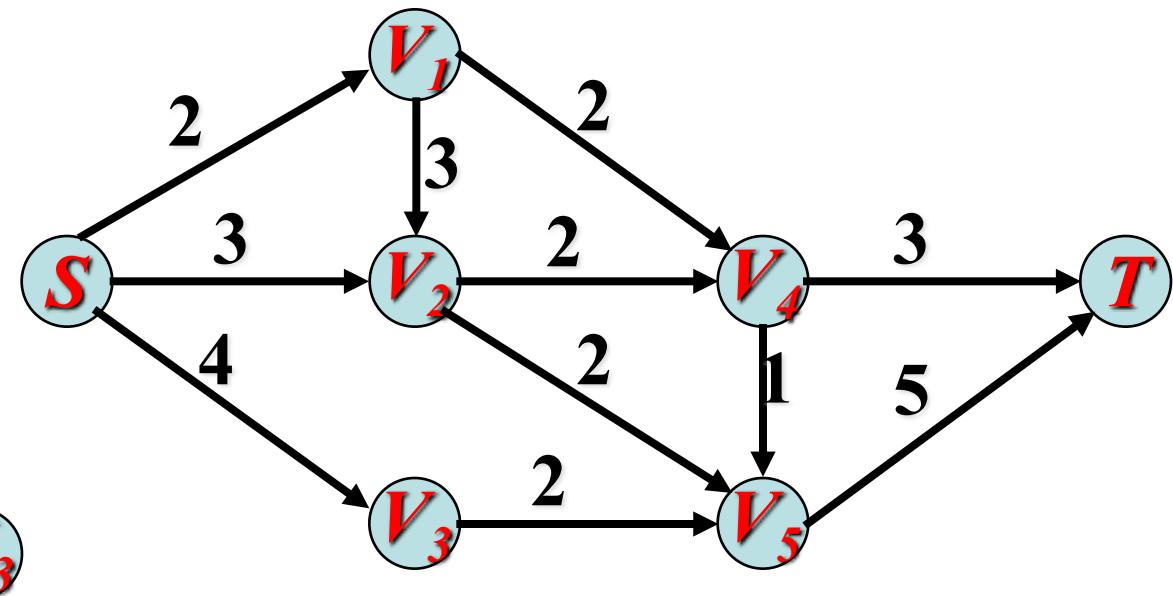
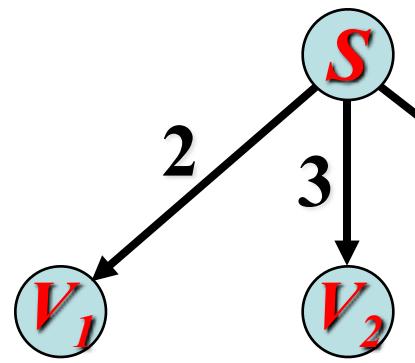
# 例1. 最短路径问题：

— 输入：



— 输出：发现一个从  $S$  到  $T$  的最短路径

## - 求解树的第一阶段



$$g(V_1) = 2, \quad g(V_2) = 3, \quad g(V_3) = 4$$

~~$$h^*(V_1) = 5, \quad f^*(V_1) = g(V_1) + h^*(V_1) = 7$$~~

## - 估计 $h^*(n)$

- 从  $V_1$  出发有两种可能：代价为 2，代价为 3，最小者为 2
- $h^*(V_1) \geq 2$ ，选择  $h(n)=2$  为  $h^*(V_1)$  的估计值
- $f(V_1) = g(v_1) + h(V_1) = 4$  为  $V_1$  的代价

• A\*算法本质——已经发现的解是优化解

**定理1.** 使用 Best-first 策略搜索树, 如果 A\* 选择的节点是目标节点, 则该节点表示的解是优化解.

**证明.**

令  $n$  是任意扩展到的节点,  $t$  是途中目标节点.

往证  $f(t) = g(t)$  是优化解代价.

(1). A\* 算法使用 Best-first 策略,  $f(t) \leq f(n)$ .

(2). A\* 算法使用  $h(n) \leq h^*(n)$  估计规则,  $f(t) \leq f(n) \leq f^*(n)$ .

(3).  $\{f^*(n)\}$  中必有一个为优化解的代价, 令其为  $f^*(s)$ .

我们有  $f(t) \leq f^*(s)$ .

(4).  $t$  是目标节点  $h(t) = 0$ , 所以  $f(t) = g(t) + h(t) = g(t) \leq f^*(s)$ .

(5).  $f(t) = g(t)$  是一个可能解,  $g(t) \geq f^*(s)$ ,  $f(t) = g(t) = f^*(s)$ .

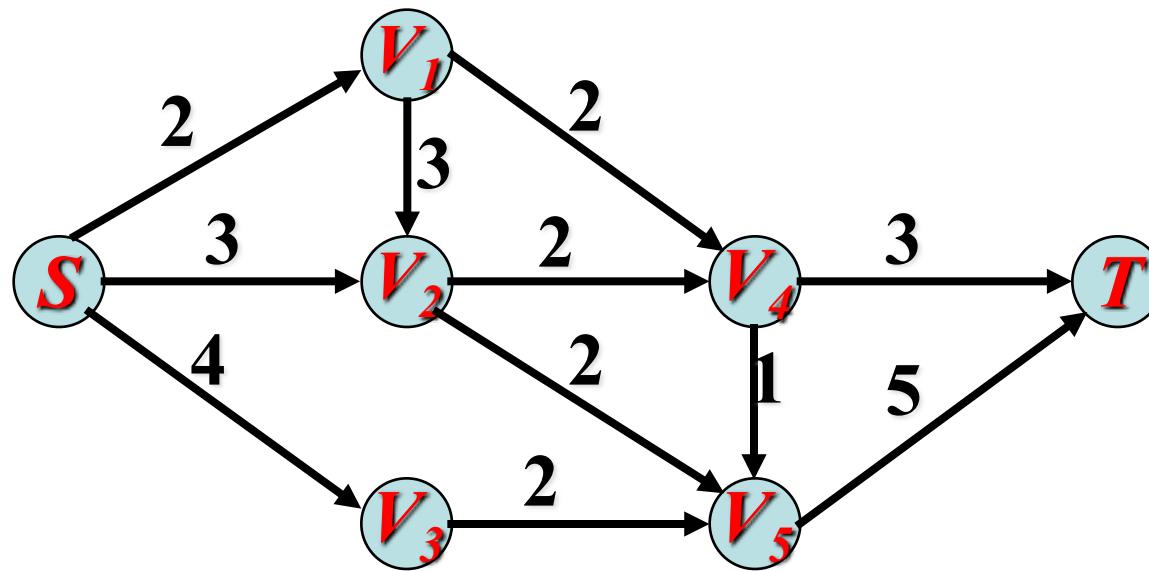


- (1). 使用 Best-first 策略搜索树;
- (2). 节点 $n$ 的代价函数为 $f(n)=g(n)+h(n)$ ,  $g(n)$ 是从根到 $n$ 的路径代价,  $h(n)$ 是从 $n$ 到某个目标节点的优化路径代价;
- (3). 对于所有 $n$ ,  $h(n) \leq h^*(n)$ ;
- (4). 当选择到的节点是目标节点时, 算法停止, 返回一个优化解.



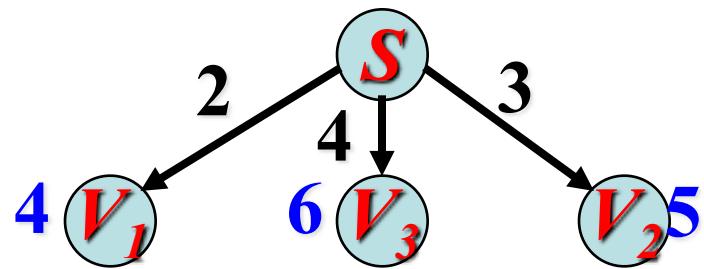
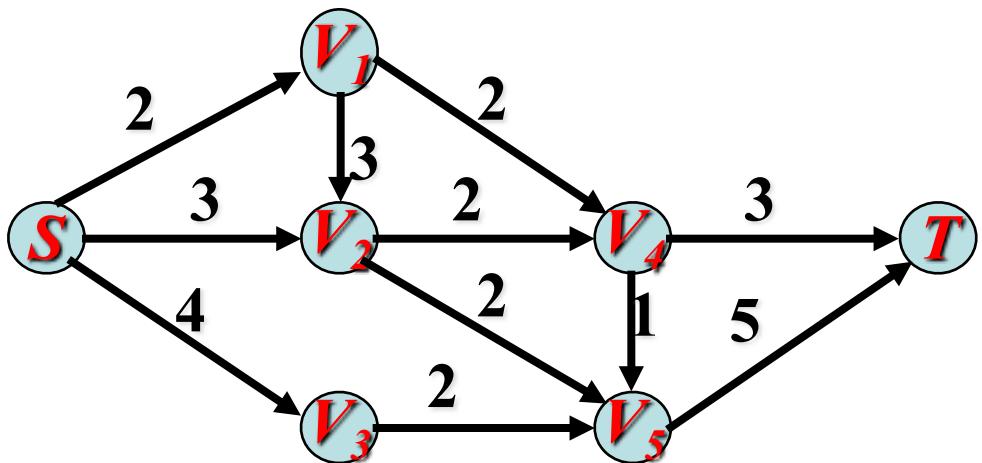
# 应用A\*算法求解最短路径问题

- 问题的输入:



- A\*算法的执行全过程

# Step 1



$$g(V_1) = 2$$

$$h(V_1) = \min\{2, 3\} = 2$$

$$f(V_1) = 2 + 2 = 4$$

$$g(V_3) = 4$$

$$h(V_3) = \min\{2\} = 2$$

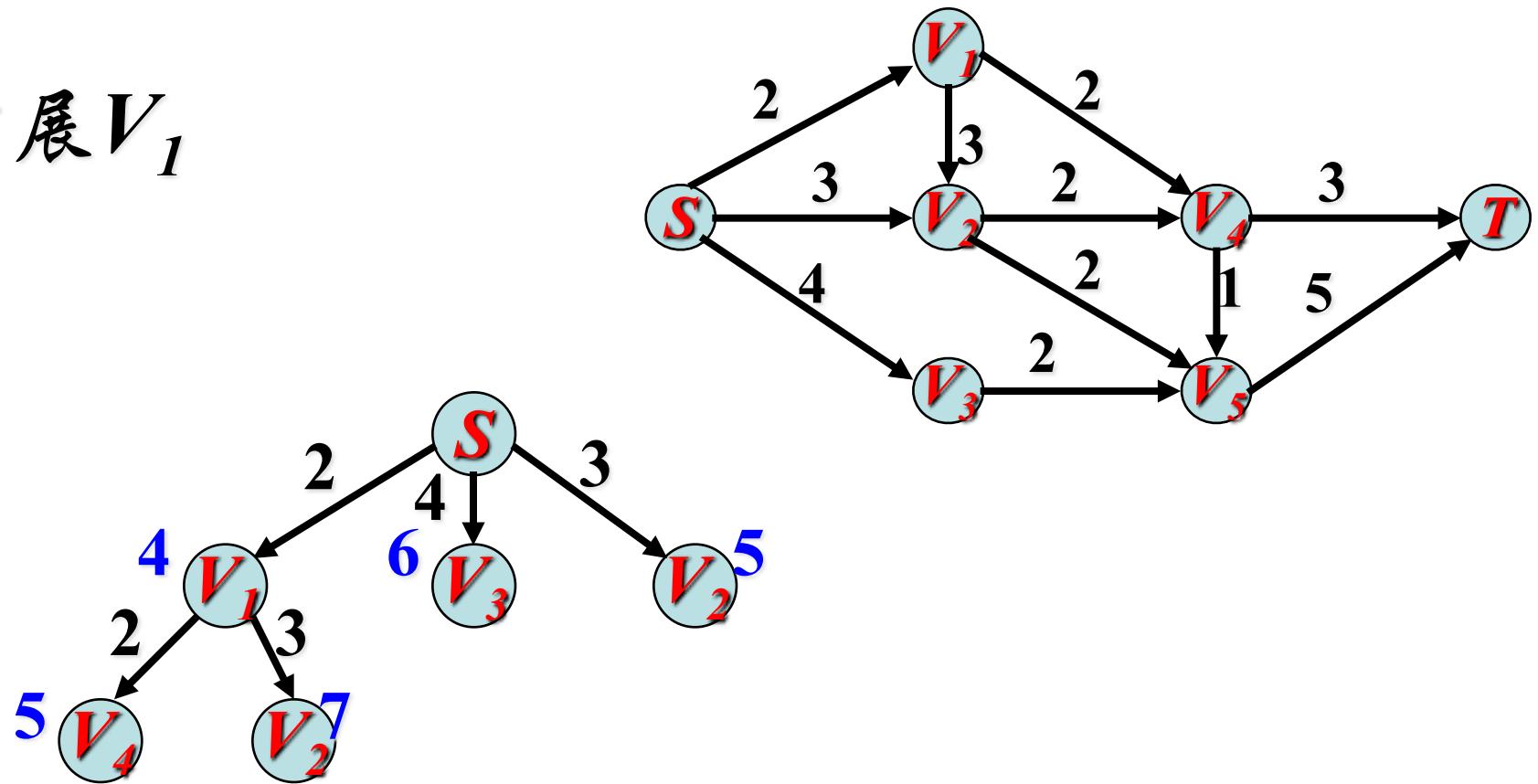
$$f(V_3) = 4 + 2 = 6$$

$$g(V_2) = 3$$

$$h(V_2) = \min\{2, 2\} = 2$$

$$f(V_2) = 2 + 2 = 5$$

## Step 2. 扩展 $V_1$



$$g(V_4) = 2 + 2 = 4$$

$$h(V_4) = \min\{3, 1\} = 1$$

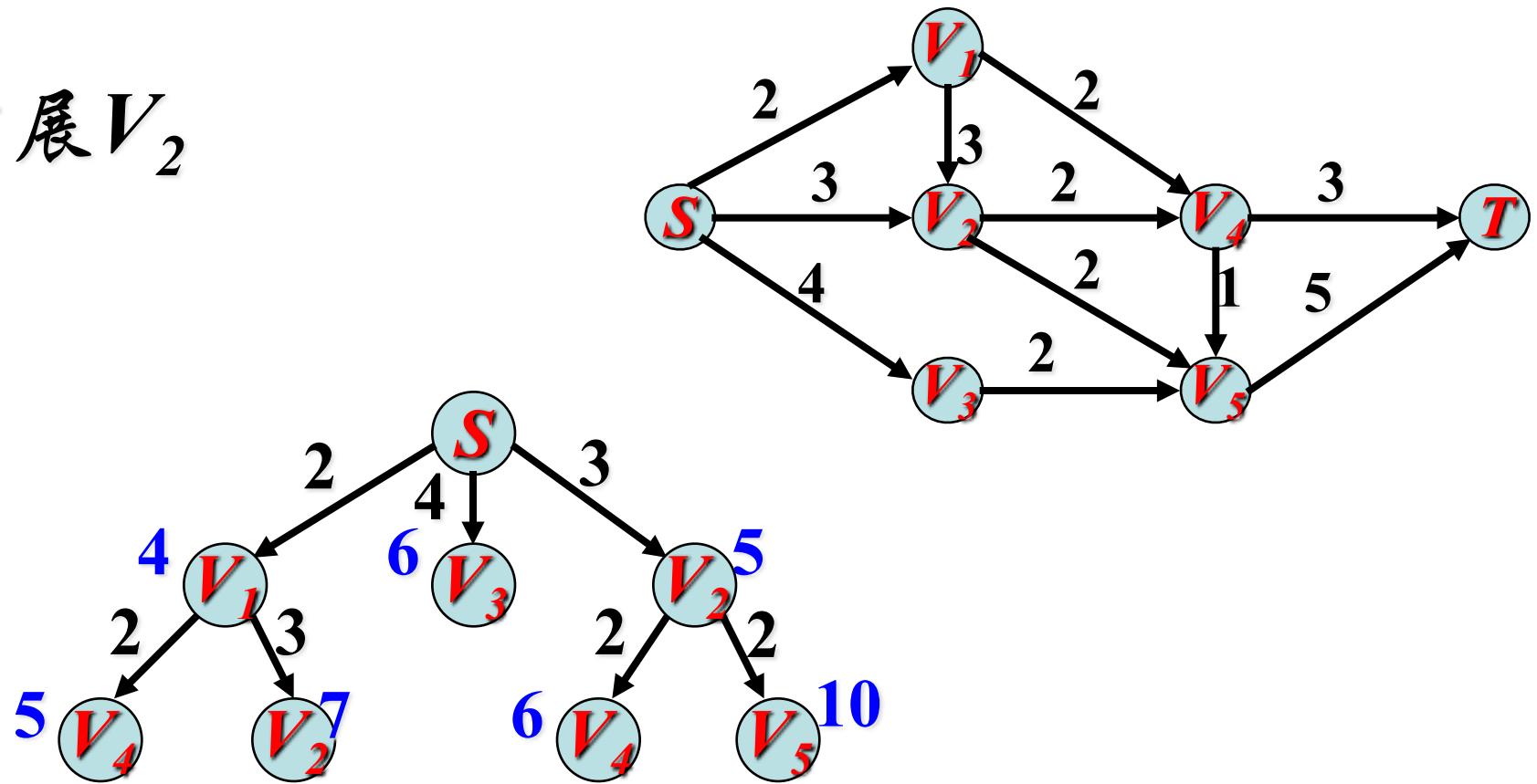
$$f(V_4) = 4 + 1 = 5$$

$$g(V_2) = 2 + 3 = 5$$

$$h(V_2) = \min\{2, 2\} = 2$$

$$f(V_2) = 5 + 2 = 7$$

### Step 3. 扩展 $V_2$



$$g(V_4) = 3 + 2 = 5$$

$$h(V_4) = \min\{3, 1\} = 1$$

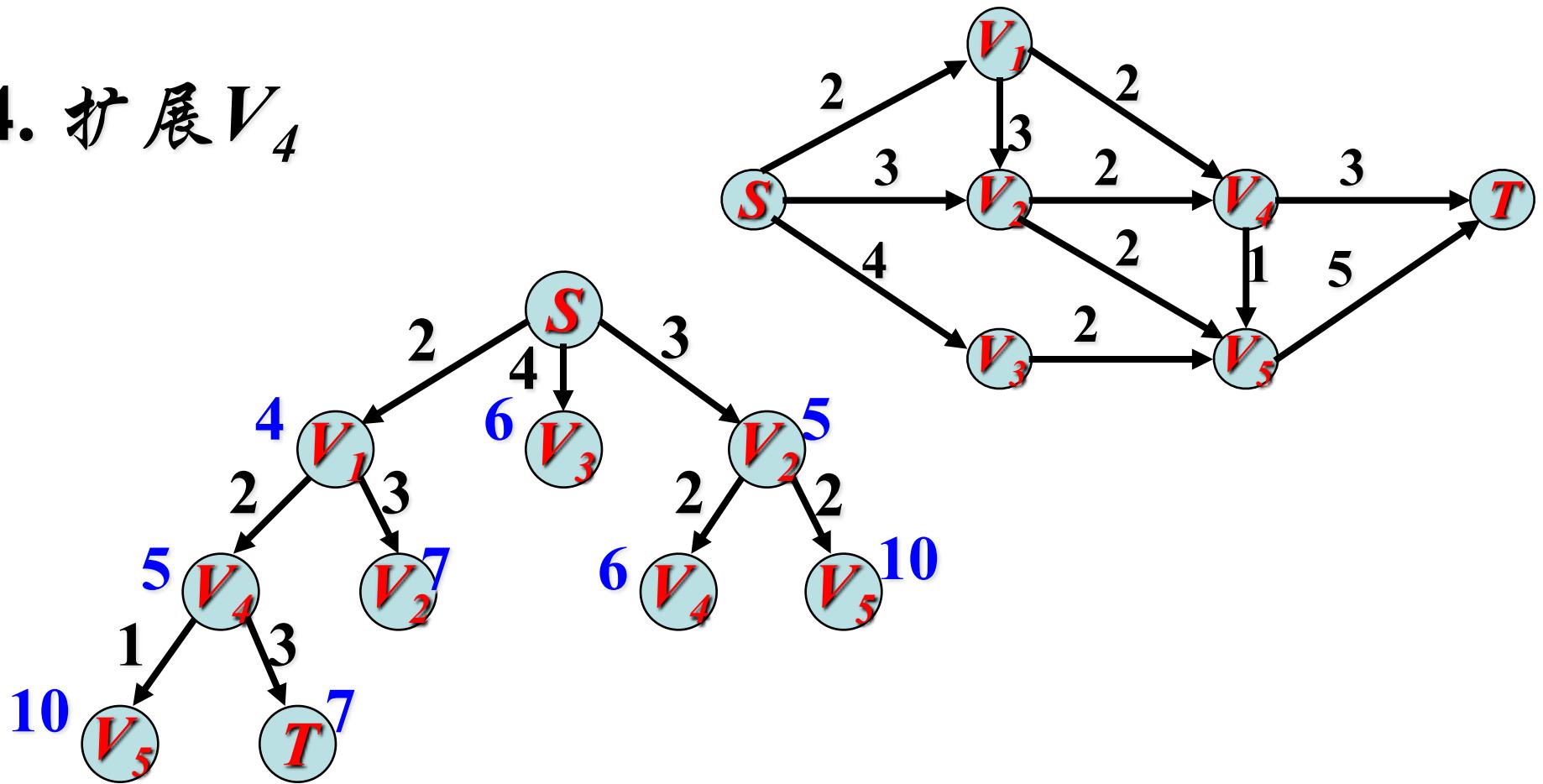
$$f(V_4) = 5 + 1 = 6$$

$$g(V_5) = 3 + 2 = 5$$

$$h(V_5) = \min\{5\} = 5$$

$$f(V_2) = 5 + 5 = 10$$

## Step 4. 扩展 $V_4$



$$g(V_5) = 2 + 2 + 1 = 5$$

$$g(T) = 2 + 2 + 3 = 7$$

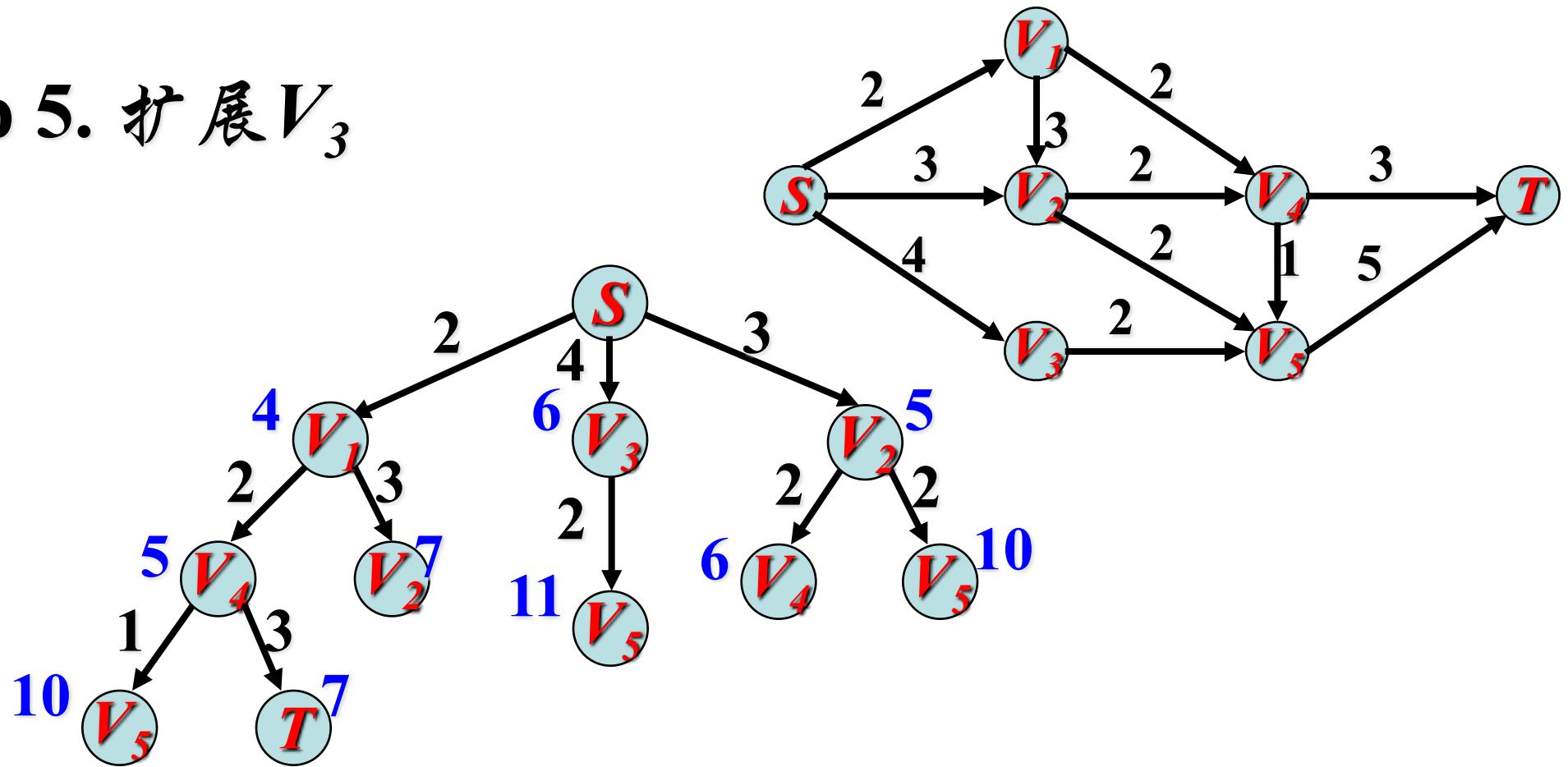
$$h(V_5) = \min\{5\} = 5$$

$$h(T) = 0$$

$$f(V_5) = 5 + 5 = 10$$

$$f(T) = 7 + 0 = 7$$

## Step 5. 扩展 $V_3$

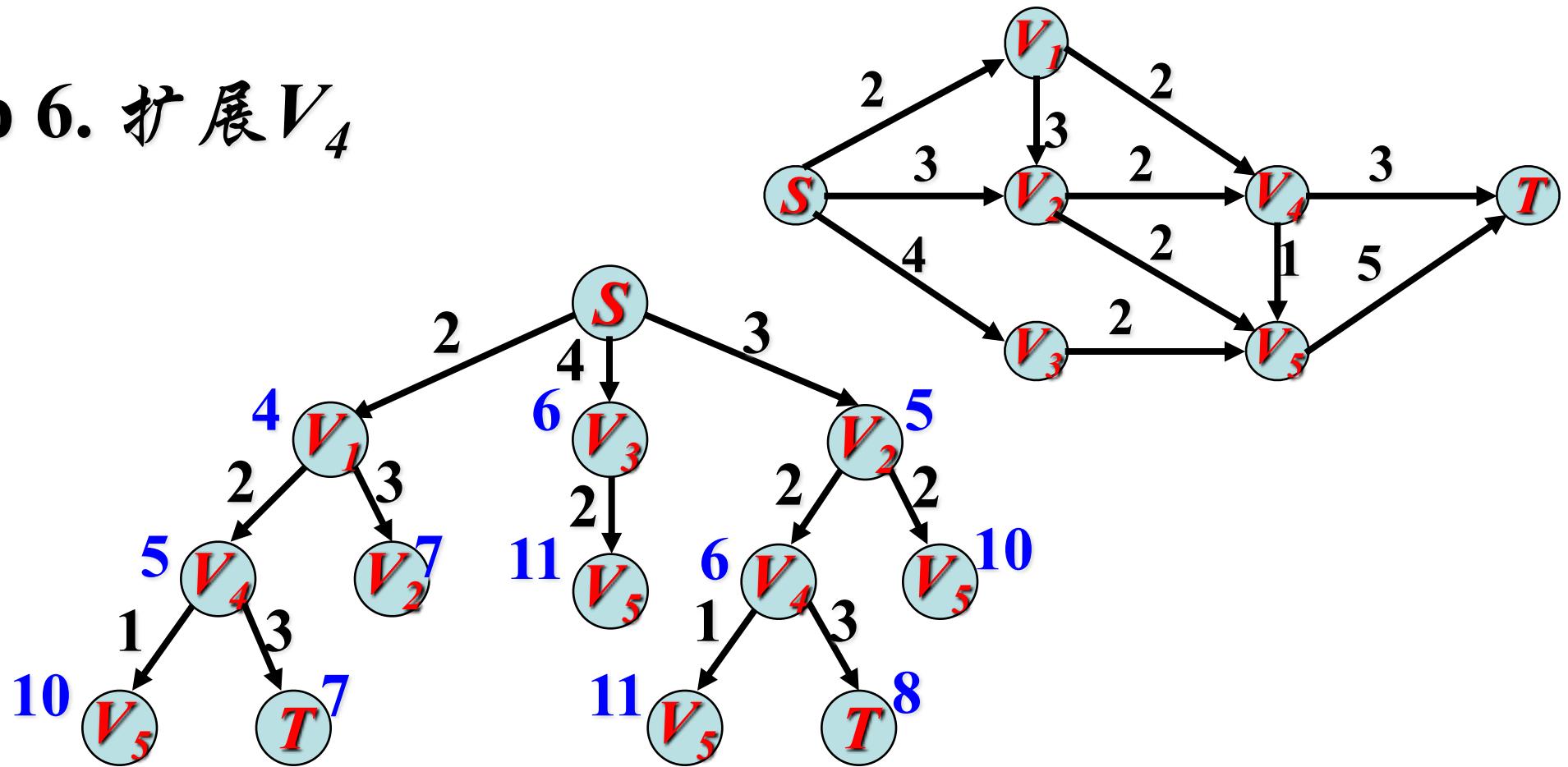


$$g(V_5) = 4 + 2 = 6$$

$$h(V_5) = \min\{5\} = 5$$

$$f(V_5) = 6 + 5 = 11$$

## Step 6. 扩展 $V_4$



$$g(V_5) = 3 + 2 + 1 = 6$$

$$g(T) = 3 + 2 + 3 = 8$$

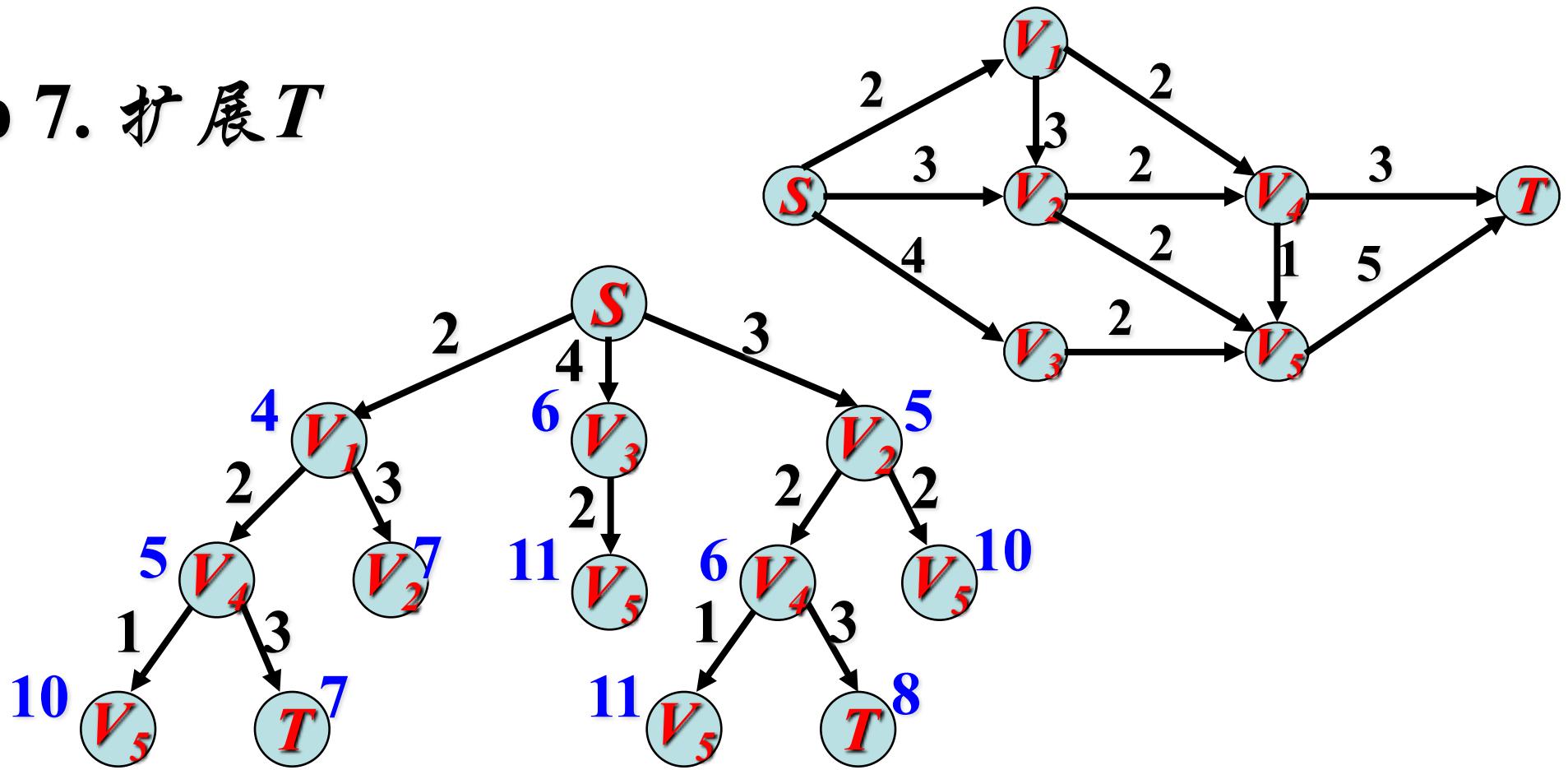
$$h(V_5) = \min\{5\} = 5$$

$$h(T) = 0$$

$$f(V_5) = 6 + 5 = 11$$

$$f(T) = 8 + 0 = 8$$

## Step 7. 扩展 $T$



因为  $T$  是目标节点, 所以我们得到解:

$$S \rightarrow V_1 \rightarrow V_4 \rightarrow T$$