搜索

东南大学计算机学院 方效林

本章内容

- 树的搜索
- 人员分配
- 旅行商问题
- A*算法

树的搜索

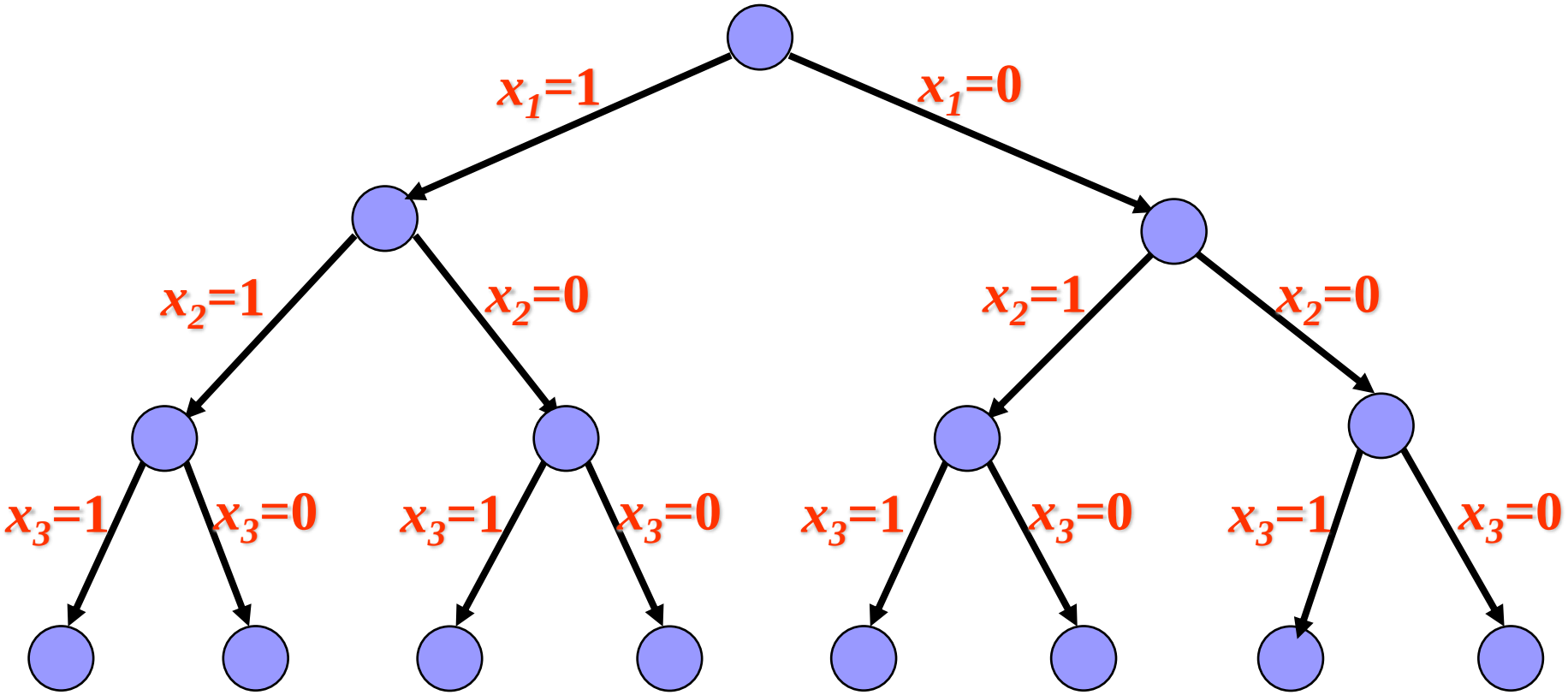
- 很多问题可以表示成为树。于是, 这些问题可以使用树搜索算法来求解

树的搜索

- CNF-SAT可满足问题
  - 给定n个布尔变量$x_1, x_2, \ldots, x_n$，和m个子句，子句是1个或多个个文字(变量或变量的非)的析取操作
    - 如：$(\overline{x}_1 \vee x_2)$，$(\overline{x}_3 \vee x_1 \vee x_5)$
  - 给定这m个子句的合取范式
    - $(\overline{x}_1 \vee x_2) \wedge (x_7 \vee x_9 \vee \overline{x}_{10}) \wedge \cdots \wedge x_5$
  - 问该合取范式的值可否为真？

树的搜索

- 把问题表示为树
  - 通过不断地为赋值集合分类来建立树



树的搜索

- 8数码问题
  - 输入：具有8个编号小方块的魔方

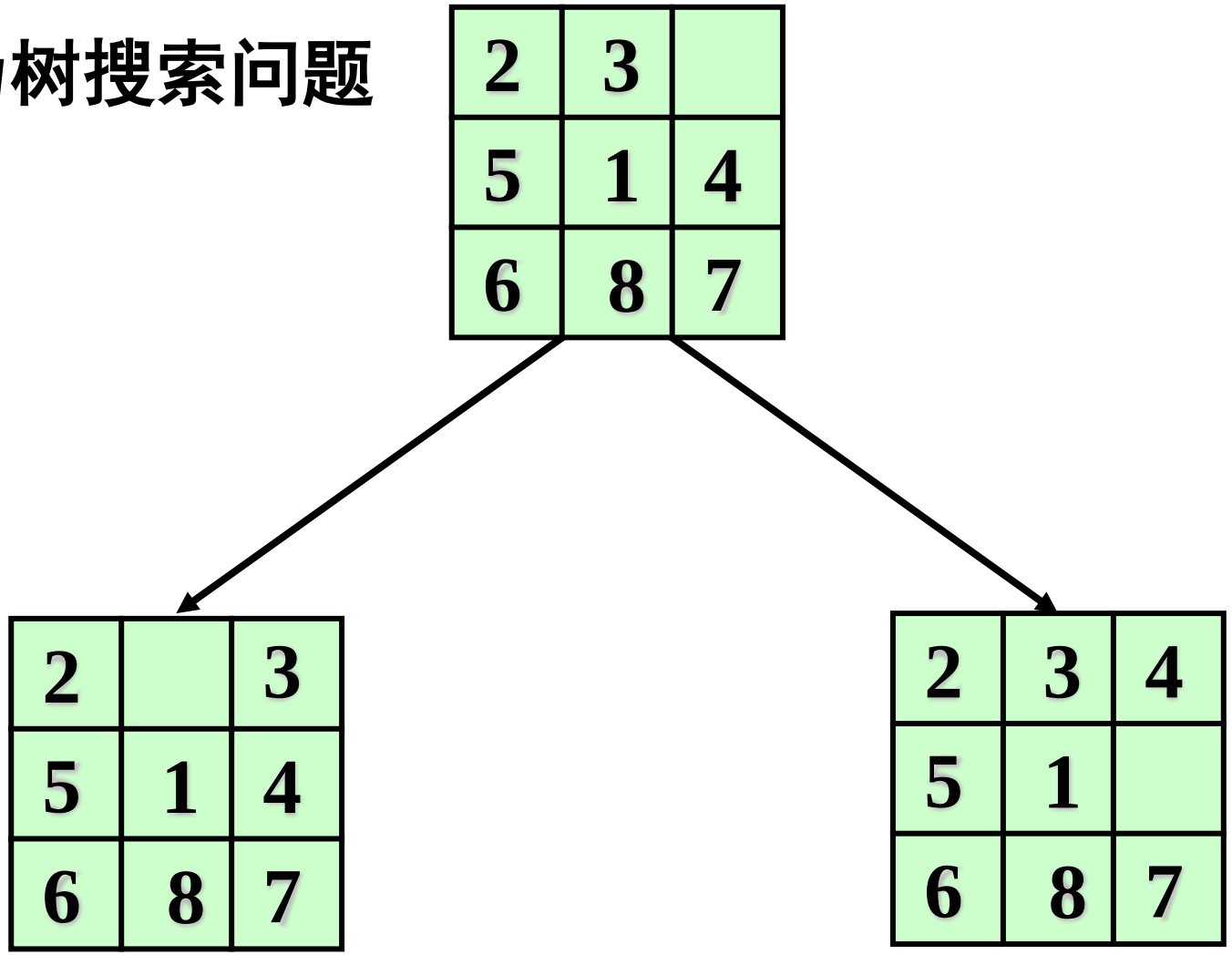
2	3	
5	1	4
6	8	7

  - 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

树的搜索

- 转换为树搜索问题

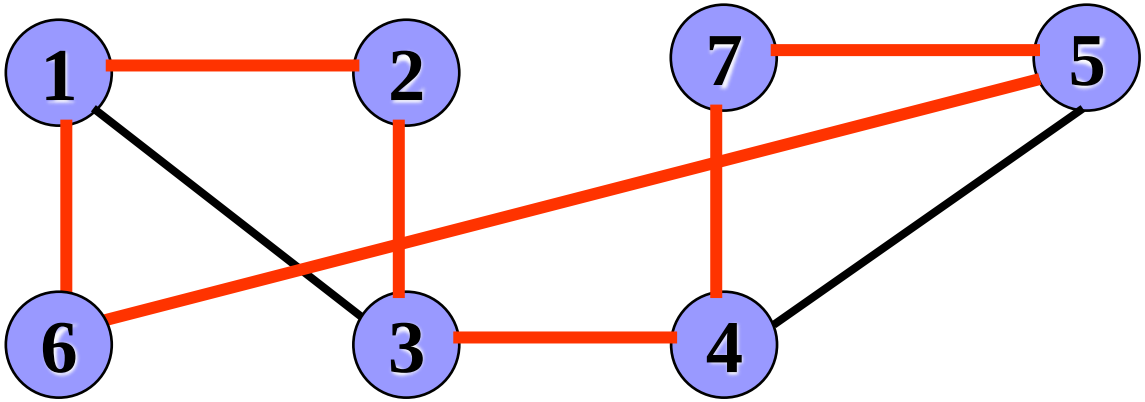


树的搜索

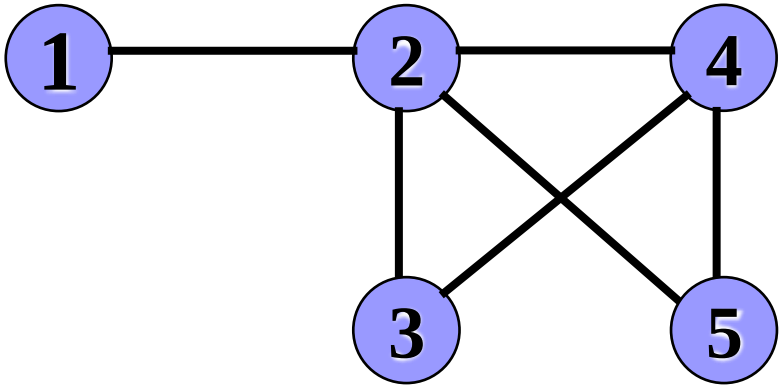
- 汉密尔顿环问题
  - 输入: 具有n个节点的连通图G=(V, E)
  - 输出: G中是否具有汉密尔顿环
    - 沿着G的n条边经过每个节点一次, 并回到起始节点的环称为G的一个汉密尔顿环

树的搜索

有汉密尔顿环图:

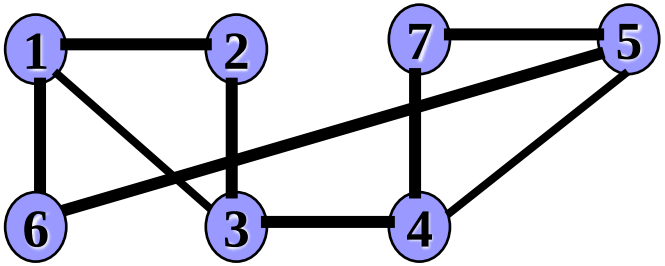


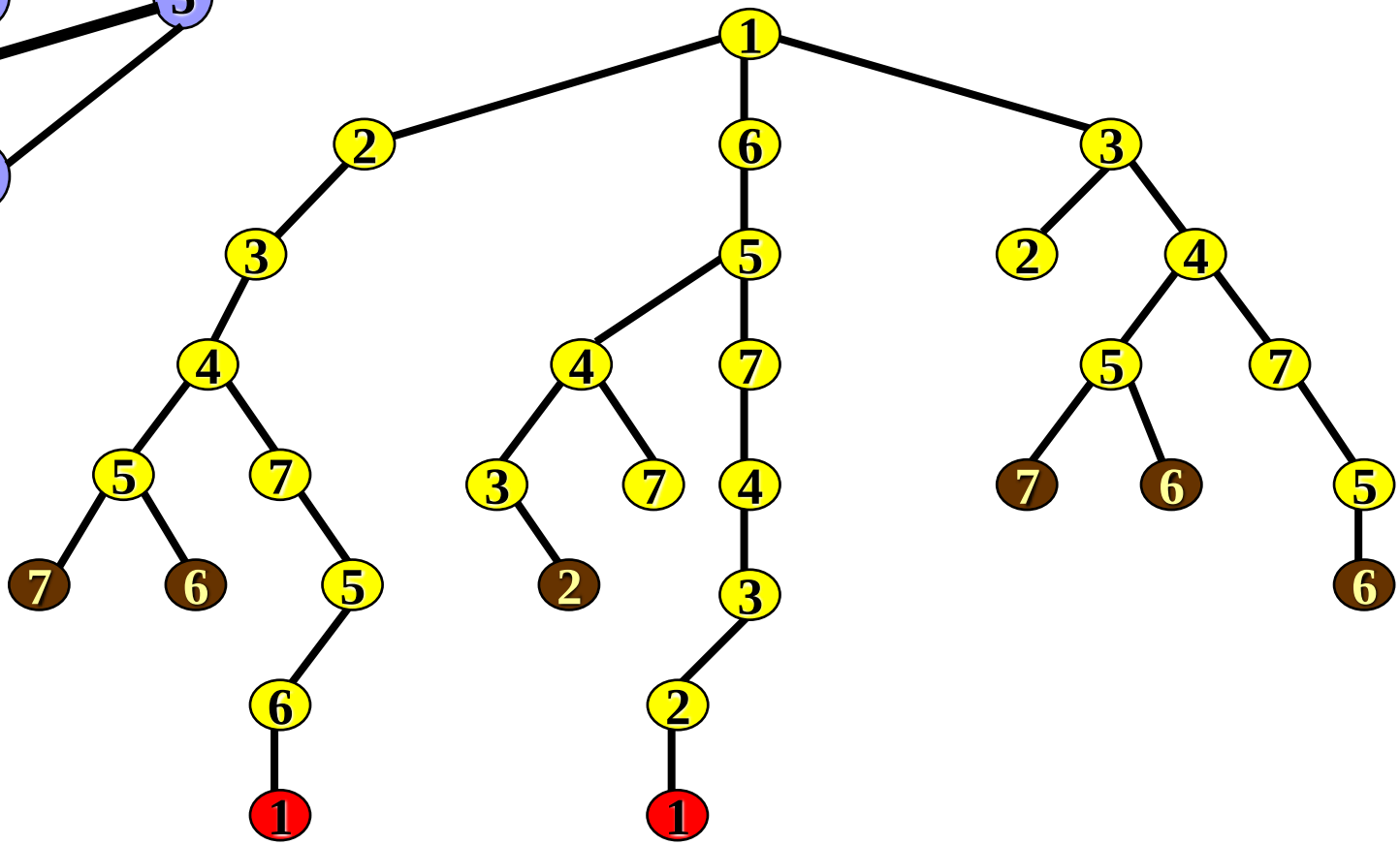
无汉密尔顿环图:



树的搜索

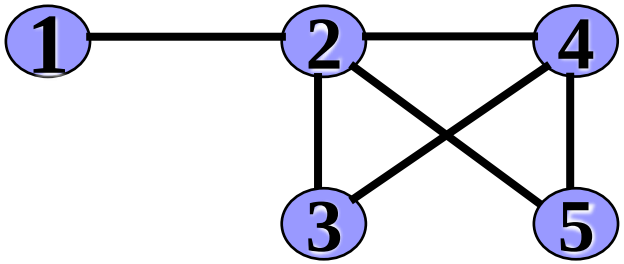
- 转换为树搜索问题

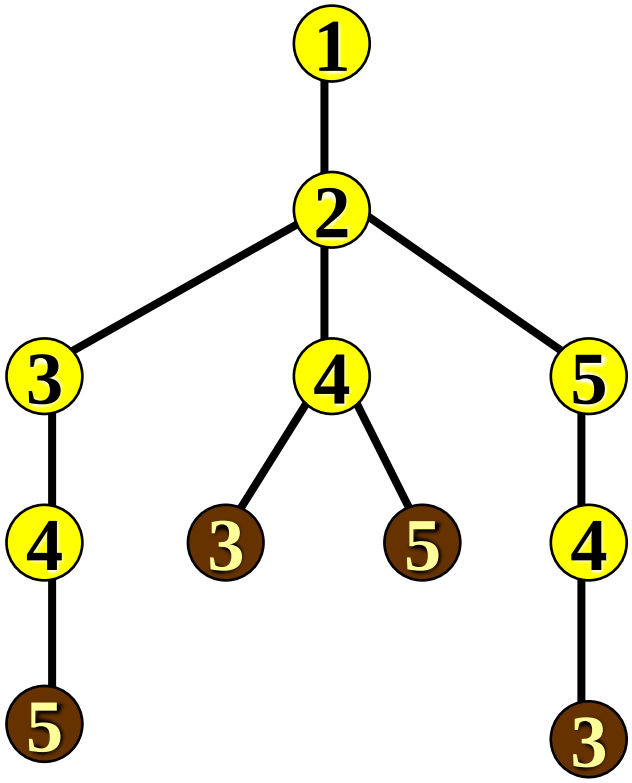




树的搜索

- 转换为树搜索问题





树的搜索

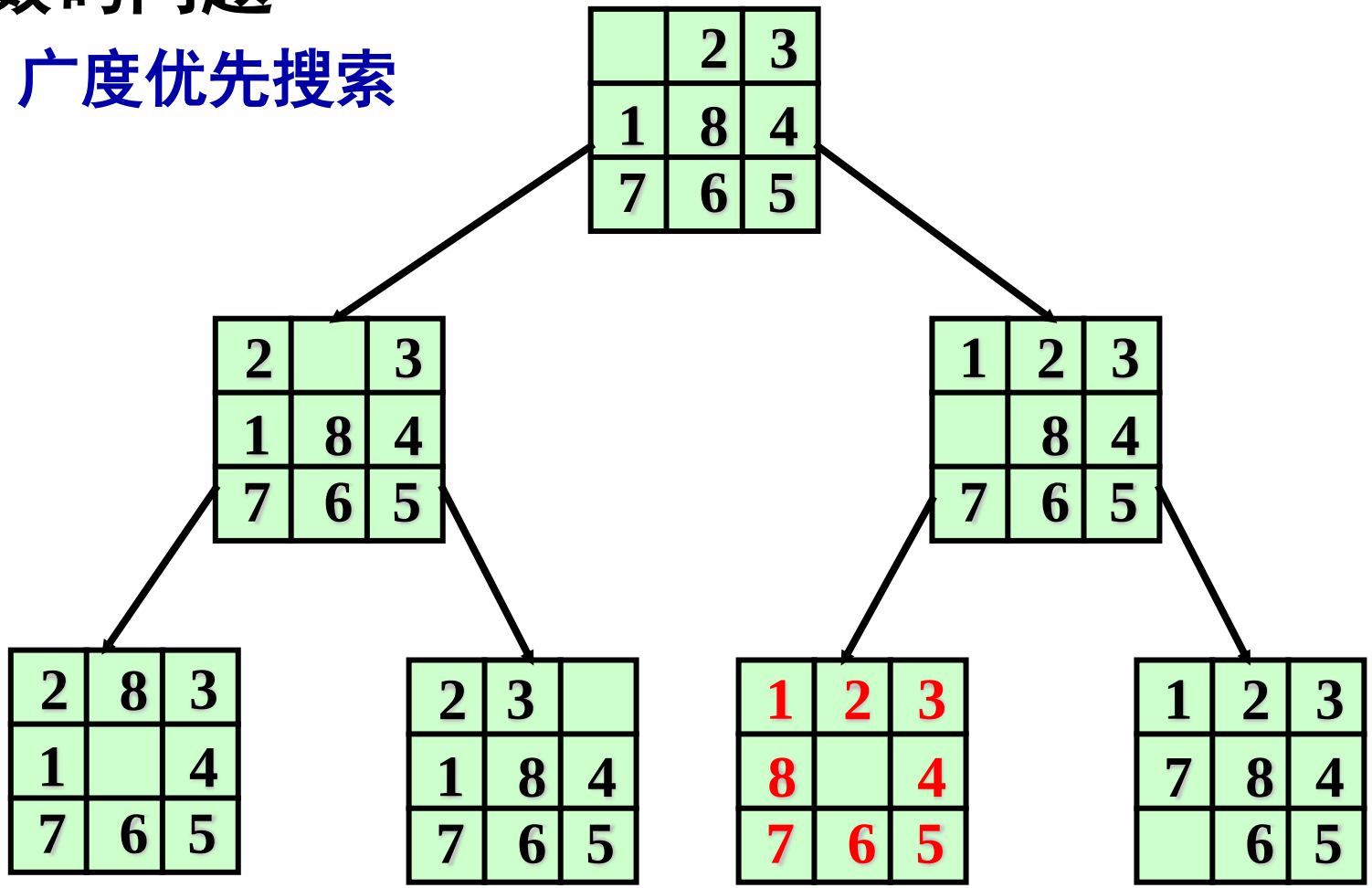
- 树的搜索策略
  - 广度优先搜索
  - 深度优先搜索

树的搜索

- 树的搜索策略
  - 广度优先搜索
    - 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.

树的搜索

- 8数码问题
  - 广度优先搜索

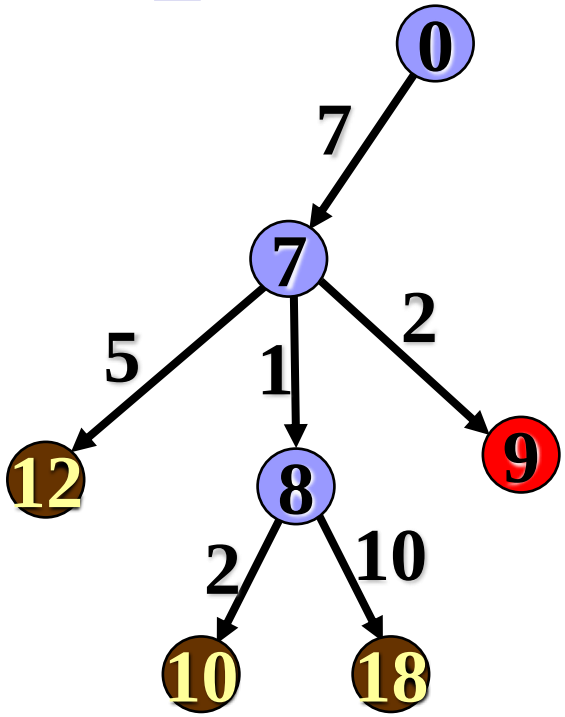


树的搜索

- 树的搜索策略
  - 深度优先搜索
    - 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.

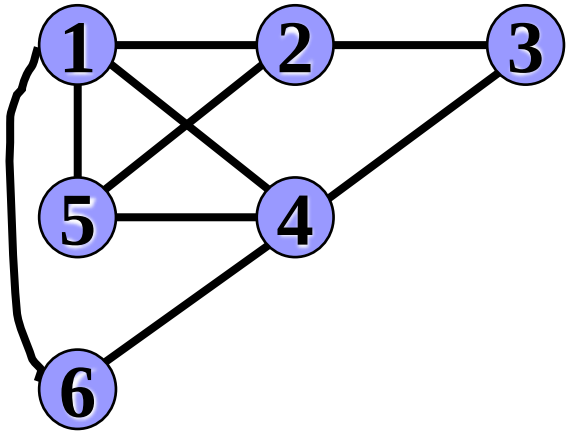
树的搜索

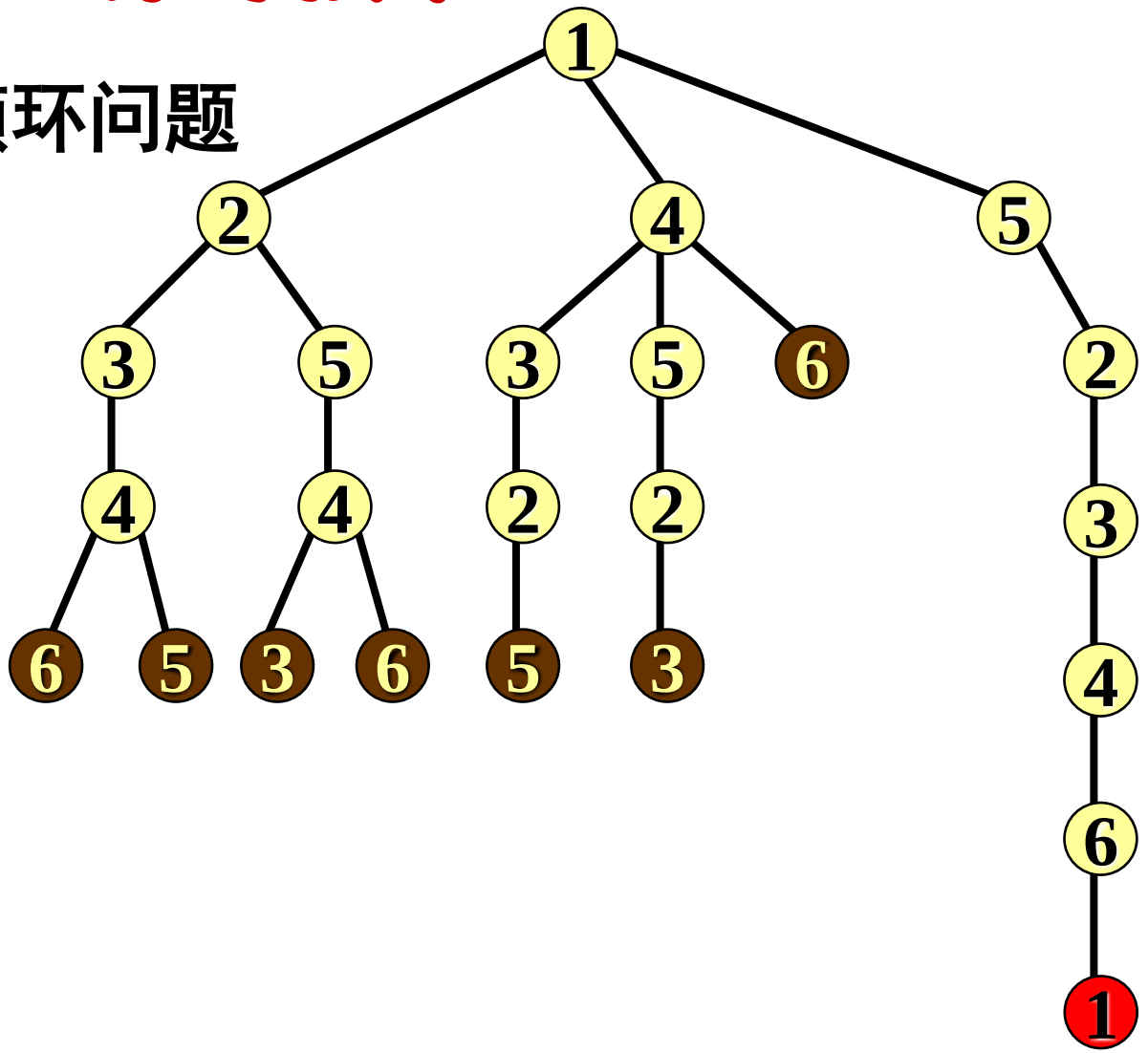
- 求解子集合和问题
  - 输入: S={7, 5, 1, 2, 10}
  - 输出: 是否存在S’ ⊆ S，使得Sum(S’)=9



树的搜索

- 求解汉密尔顿环问题





树的搜索

- 搜索策略的优化
  - 爬山法
  - Best first strategy
  - 分支界限法

树的搜索

- 搜索策略的优化
  - 爬山法
    - 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

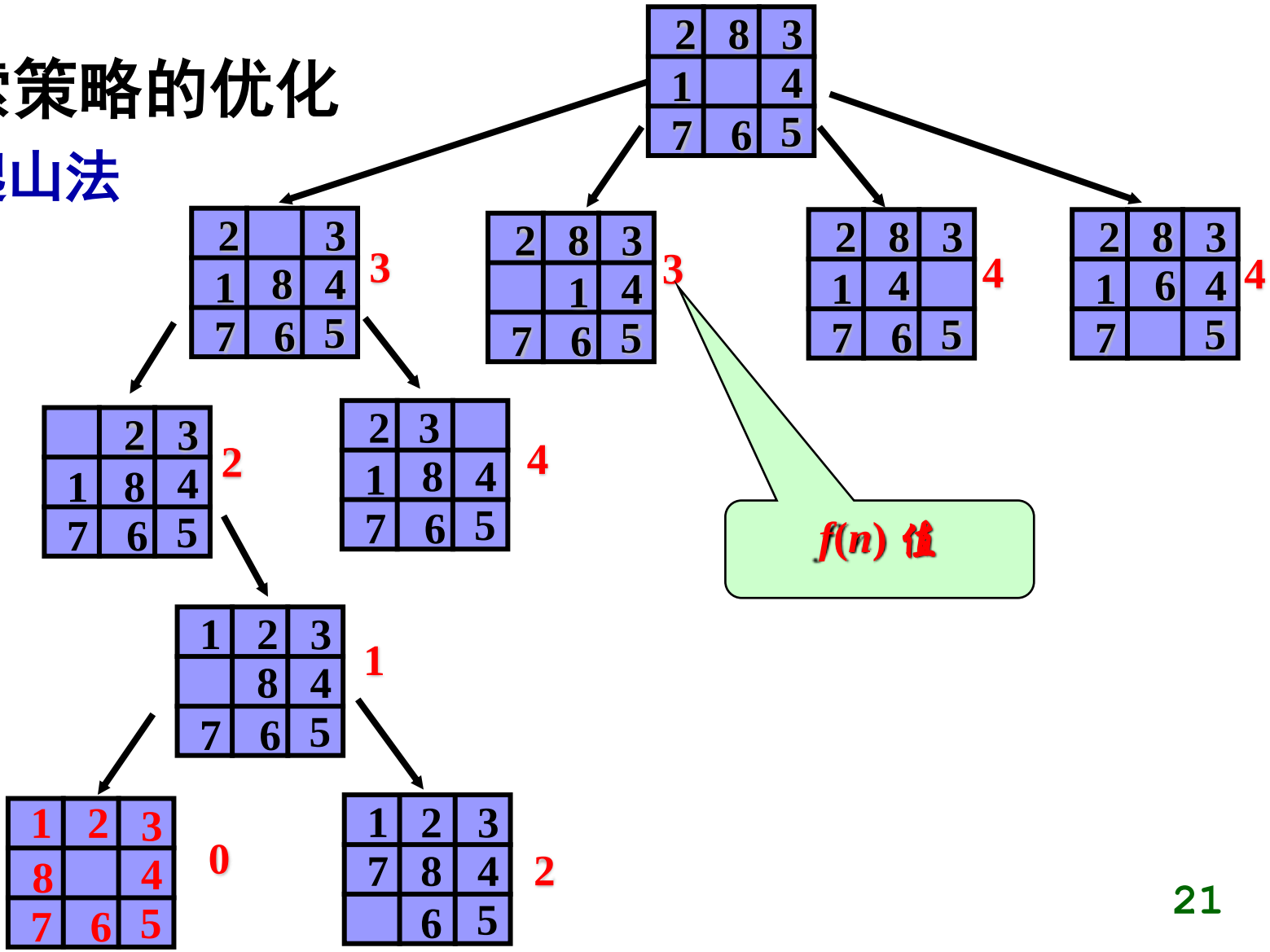
树的搜索

- 搜索策略的优化
  - 爬山法
    - 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

树的搜索

- 搜索策略的优化
  - 爬山法



树的搜索

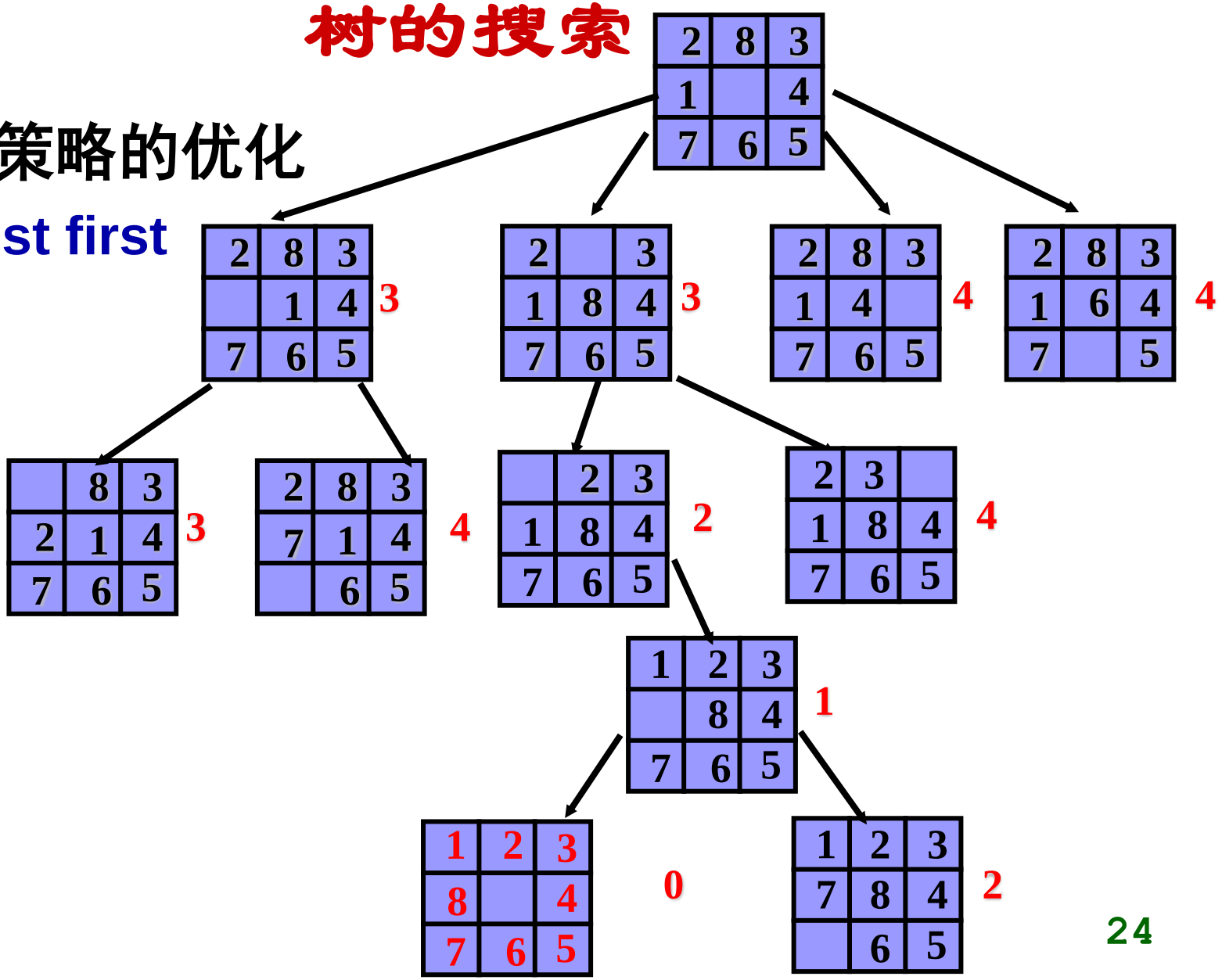
- 搜索策略的优化
  - 爬山法
    - 1. 构造由根组成的单元素栈S;
    - 2. If Top(S)是目标节点 Then 停止;
    - 3. Pop(S);
    - 4. S的子节点按照其启发测度由大到小的顺序压入S;
    - 5. If S空 Then 失败 Else goto 2.

树的搜索

- 搜索策略的优化
  - Best first strategy
    - 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.

树的搜索

- 搜索策略的优化
  - Best first



树的搜索

- 搜索策略的优化
  - Best first strategy
    - 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.

树的搜索

- 搜索策略的优化
  - 分支界限法
    - 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.
  - 0/1背包问题
    - 给定 n 个物品和一个背包，物品 i 的重量是 wi，价值 vi，背包容量为C，问如何选择装入背包的物品，使装入背包中的物品的总价值最大？

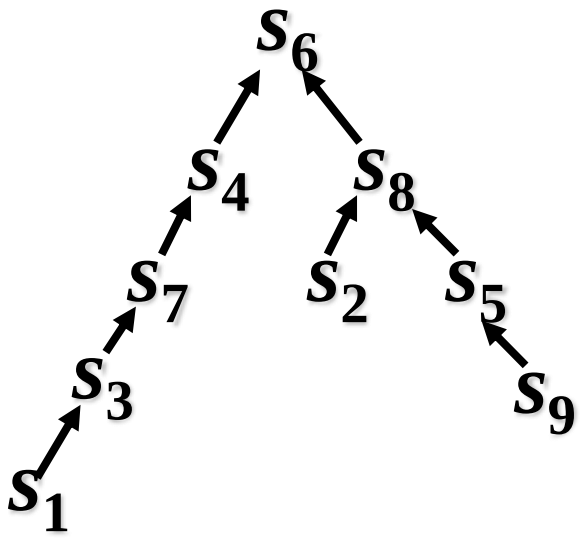
人员分配问题

- 问题定义
  - 人的集合$P=\{P_1, P_2, \ldots, P_n\}$，$P_1<P_2<\ldots<P_n$
  - 工作的集合$J=\{J_1, J_2, \ldots, J_n\}$，$J$是偏序集合
  - 矩阵$[C_{ij}]$, C_{ij}是工作J_j分配到P_i的代价
- 输出
  - 矩阵$[X_{ij}]$, $X_{ij}=1$表示P_i被分配J_j，$\sum_{i,j} C_{ij}X_{ij}$最小
  - 每个人被分配一种工作, 不同人分配不同工作
  - 如果$f(P_i)\le f(P_j)$，则$P_i \le P_j$

给定P=$\{P_1, P_2, P_3\}$, $J=\{J_1, J_2, J_3\}$，$J_1\le J_3$, $J_2\le J_3$
$P_1\to J_1$、$P_2\to J_2$、$P_3\to J_3$是可能的解.
$P_1\to J_1$、$P_2\to J_3$、$P_3\to J_2$不可能是解.

人员分配问题

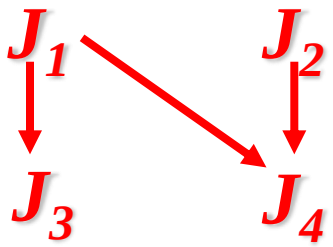
- 转换为树搜索问题
  - 拓朴排序
  - 输入: 偏序集合(S, ≤)
  - 输出: S的拓朴序列是<s1, s2, …, sn>,
  - 满足: 如果si ≤ sj, 则si排在sj的前面.



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

人员分配问题

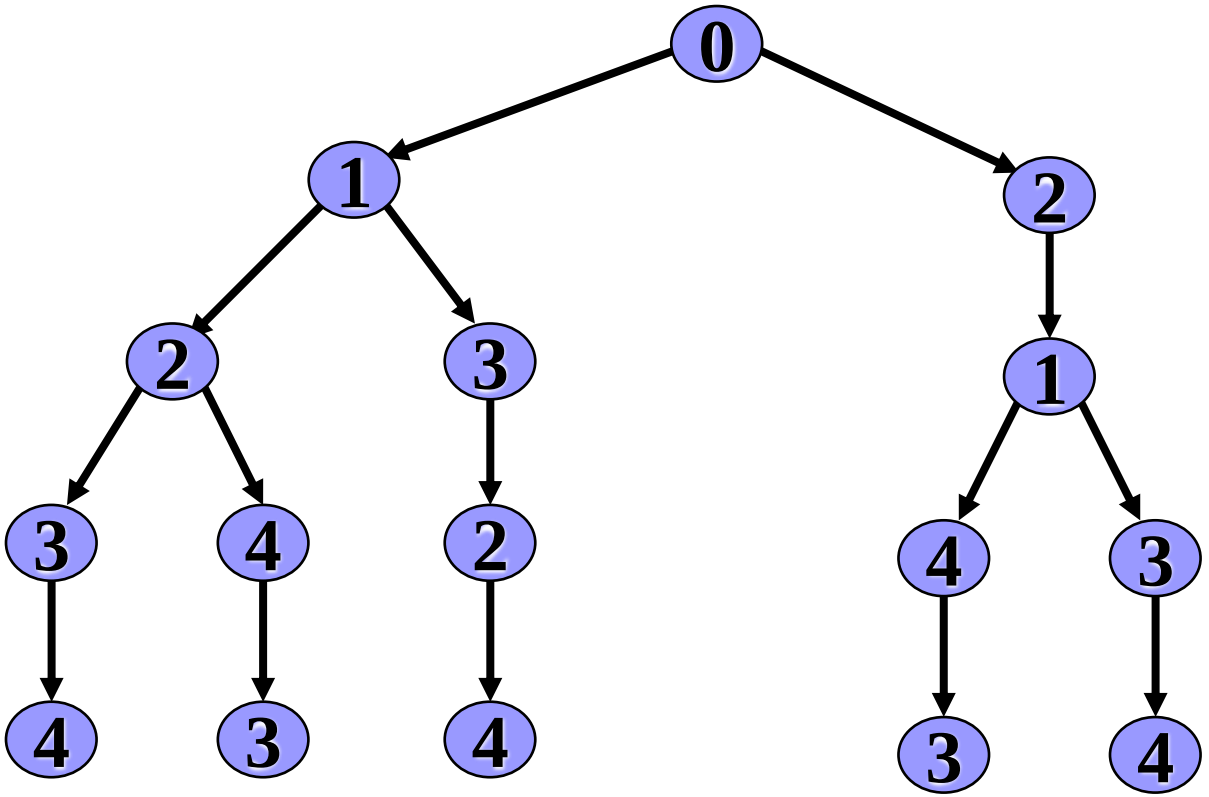
- 问题的解空间
  - $P_1 \rightarrow J_{k1}$、$P_2 \rightarrow J_{k2}$、…、$P_n \rightarrow J_{kn}$是一个可能解，则J_{k1}、J_{k2}、…、J_{kn}必是一个拓朴排序的序列.
  - $P=\{P_1, P_2, P_3, P_4\}$, $J=\{J_1, J_2, J_3, J_4\}$, J的偏序如下



  - 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)是拓朴排序序列
  - (J_1, J_2, J_4, J_3)对应于$P_1 \rightarrow J_1$、$P_2 \rightarrow J_2$、$P_3 \rightarrow J_4$、$P_4 \rightarrow J_3$

人员分配问题

- 问题的树表示(即用树表示所有拓朴排序序列)

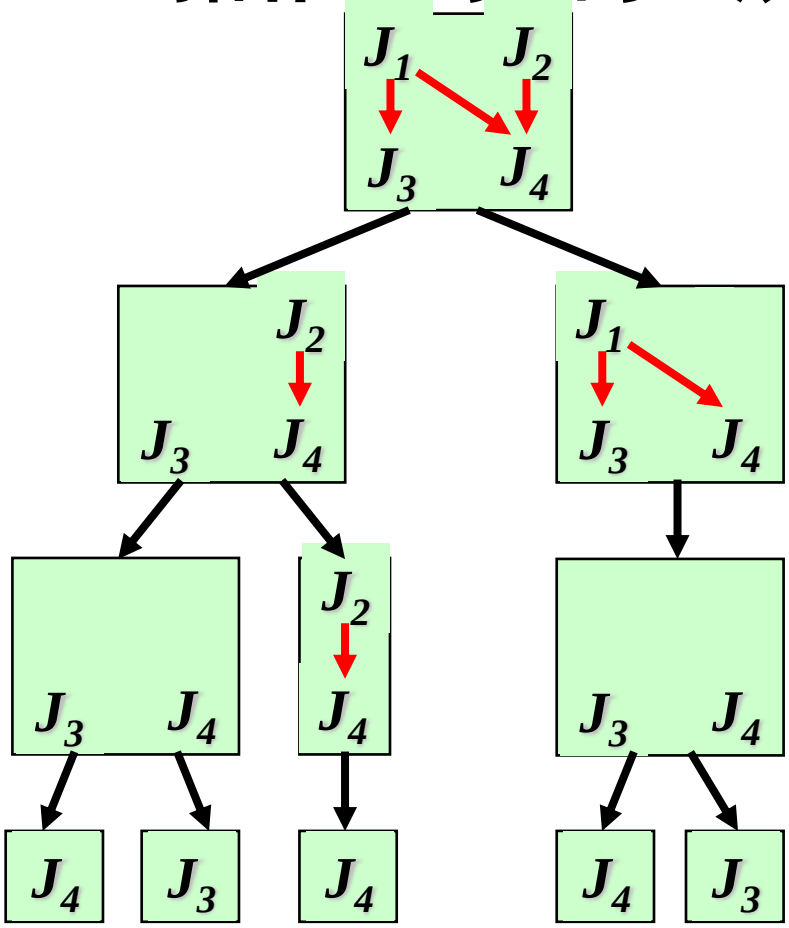


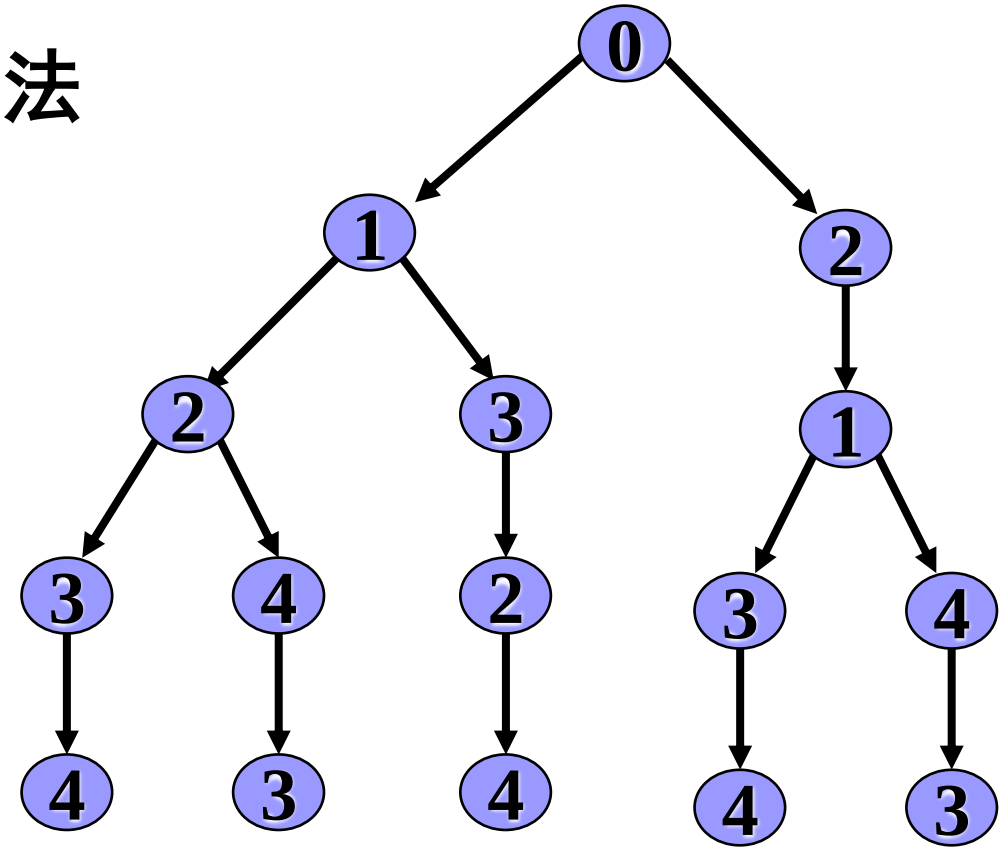
人员分配问题

- 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.

人员分配问题

- 拓朴序列树的生成算法





人员分配问题

- 计算解的代价的下界
  - 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

人员分配问题

- 每行(列)减去的数的和即为解的下界

	J_1	J_2	J_3	J_4	
P_1	29	19	17	12	-12
P_2	32	30	26	28	-26
P_3	3	21	7	9	-3
P_4	18	13	10	15	-10
		-3			

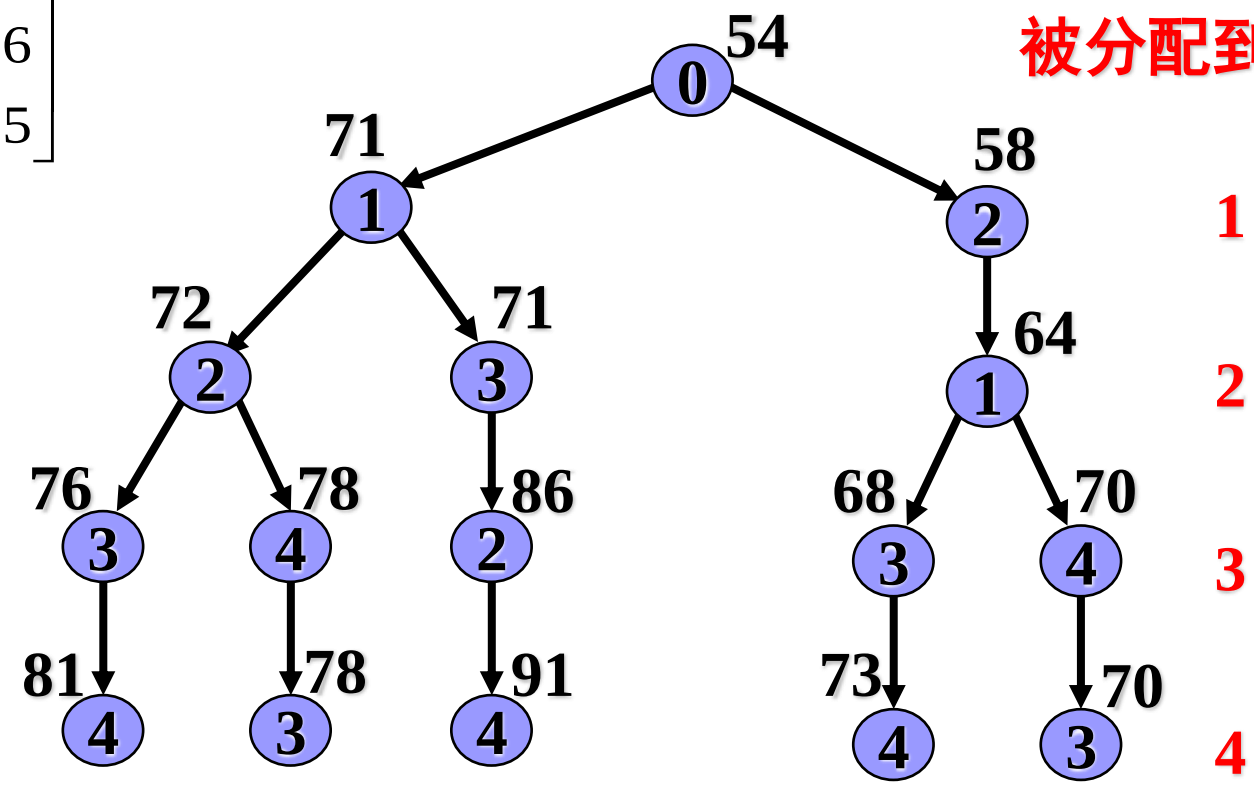
$$\rightarrow \begin{bmatrix} 17 & 4 & 5 & 0 \\ 6 & 1 & 0 & 2 \\ 0 & 15 & 4 & 6 \\ 8 & 0 & 0 & 5 \end{bmatrix}$$

解代价下界=12+26+3+10+3=54

人员分配问题

- 解空间的加权树表示

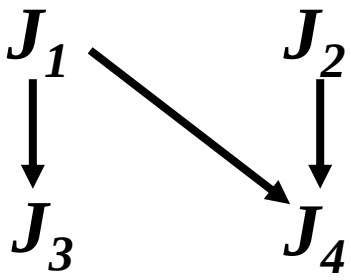
$$\begin{bmatrix} 17 & 4 & 5 & 0 \\ 6 & 1 & 0 & 2 \\ 0 & 15 & 4 & 6 \\ 8 & 0 & 0 & 5 \end{bmatrix}$$



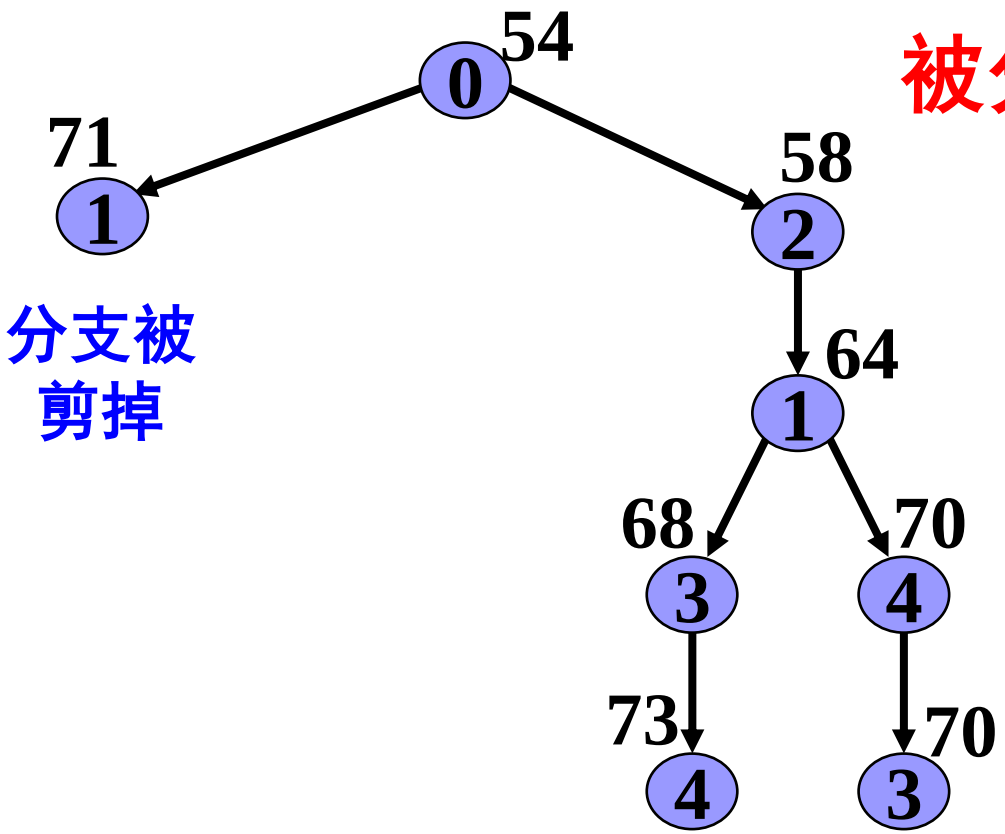
人员分配问题

- 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.

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



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



分支被剪掉

被分配到的人

1

2

3

4

旅行商问题

- 问题定义
  - 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(即权值只和)最小.

旅行商问题

- 设代价矩阵如下，计算根节点的代价下界

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

旅行商问题

- 得到如下根节点及其代价下界

所有解的集合

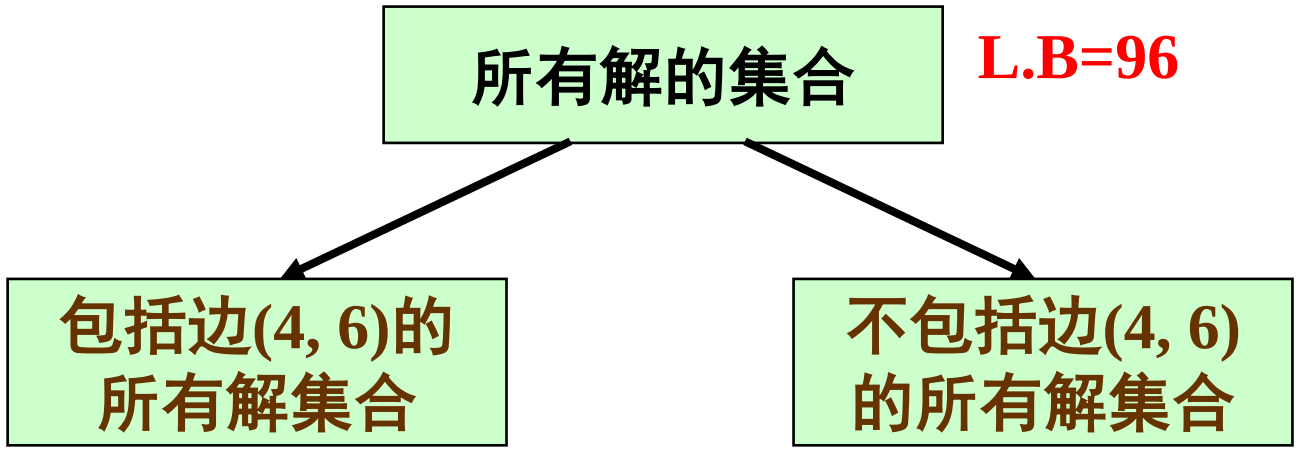
L.B=96

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

旅行商问题

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

- 构造根节点的两个子节点
  - 选择使子节点代价下界增加
  - 最小的划分边(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
    - 左子节点为不包括边(4, 6)的所有解集合

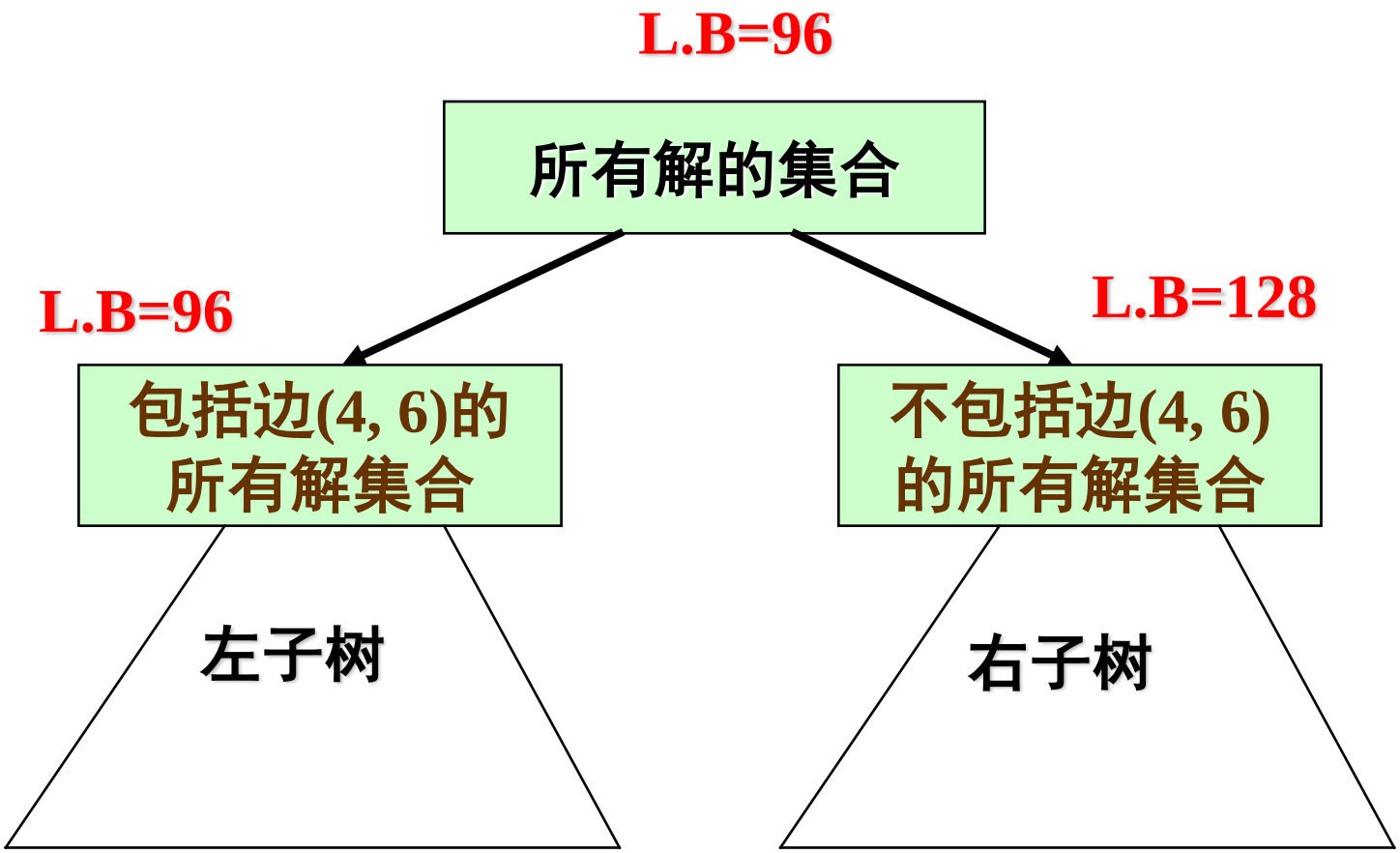


旅行商问题

- 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.
  - 现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.

旅行商问题

- 计算左右子节点的代价下界



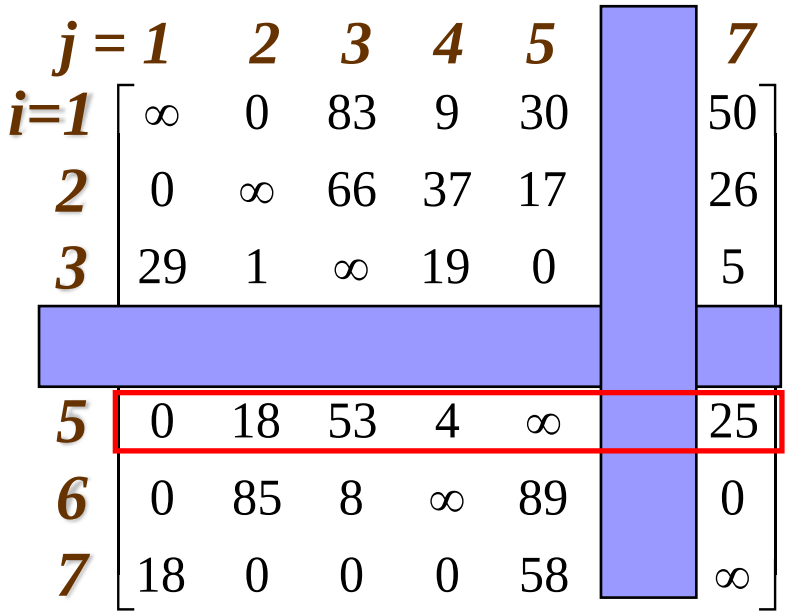
旅行商问题

- 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∞
    - 结果矩阵如下

	j = 1	2	3	4	5		7	
i=1	∞	0	83	9	30		50	
2	0	∞	66	37	17		26	
3	29	1	∞	19	0		5	
5	3	21	56	7	∞		28	-3
6	0	85	8	42	89		0	
7	18	0	0	0	58		∞	

旅行商问题

- 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	∞	0	83	9	30		50
2	0	∞	66	37	17		26
3	29	1	∞	19	0		5
5	0	18	53	4	∞		25
6	0	85	8	∞	89		0
7	18	0	0	0	58		∞

旅行商问题

- 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∞
  - 结果矩阵如下

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

-32

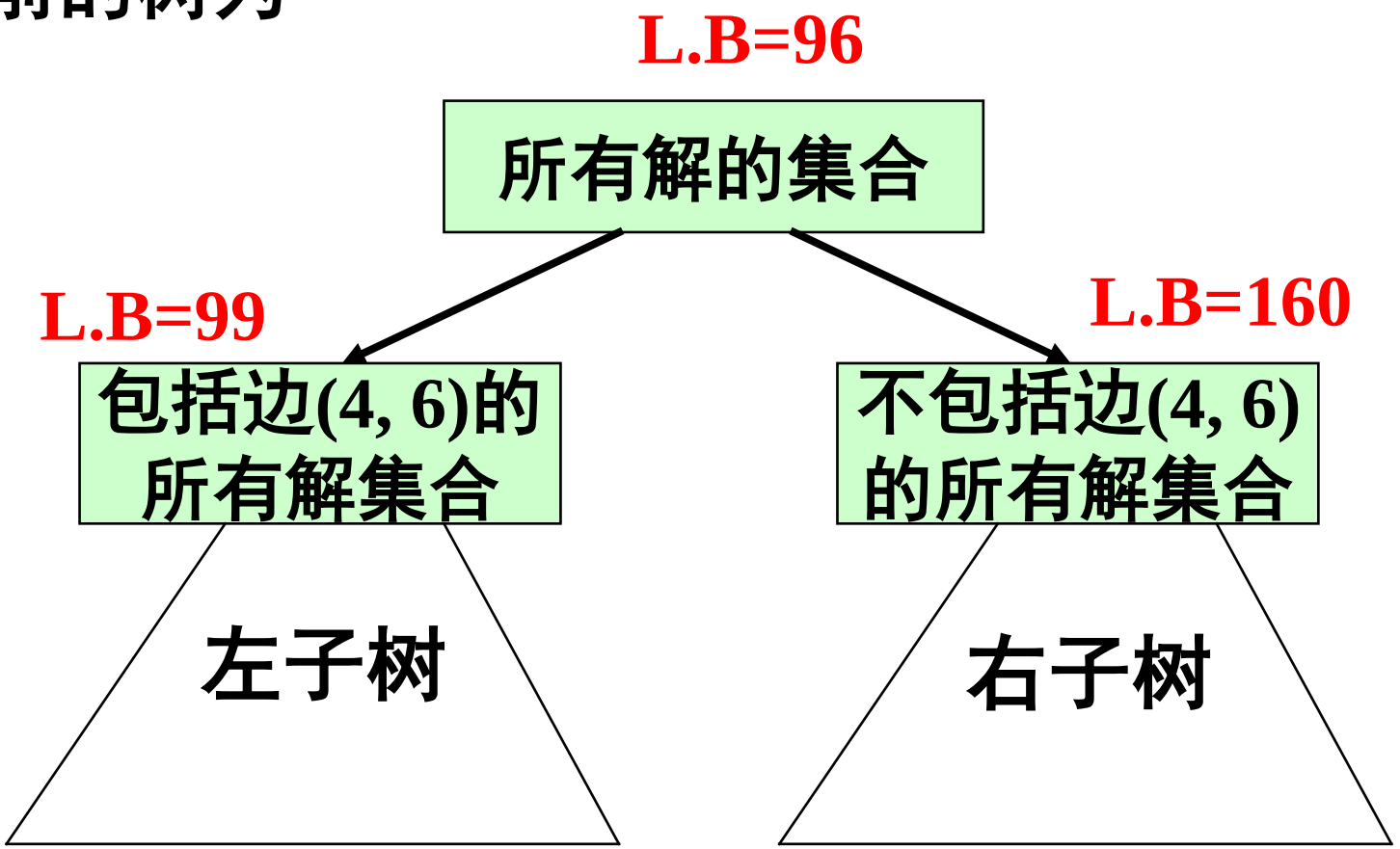
旅行商问题

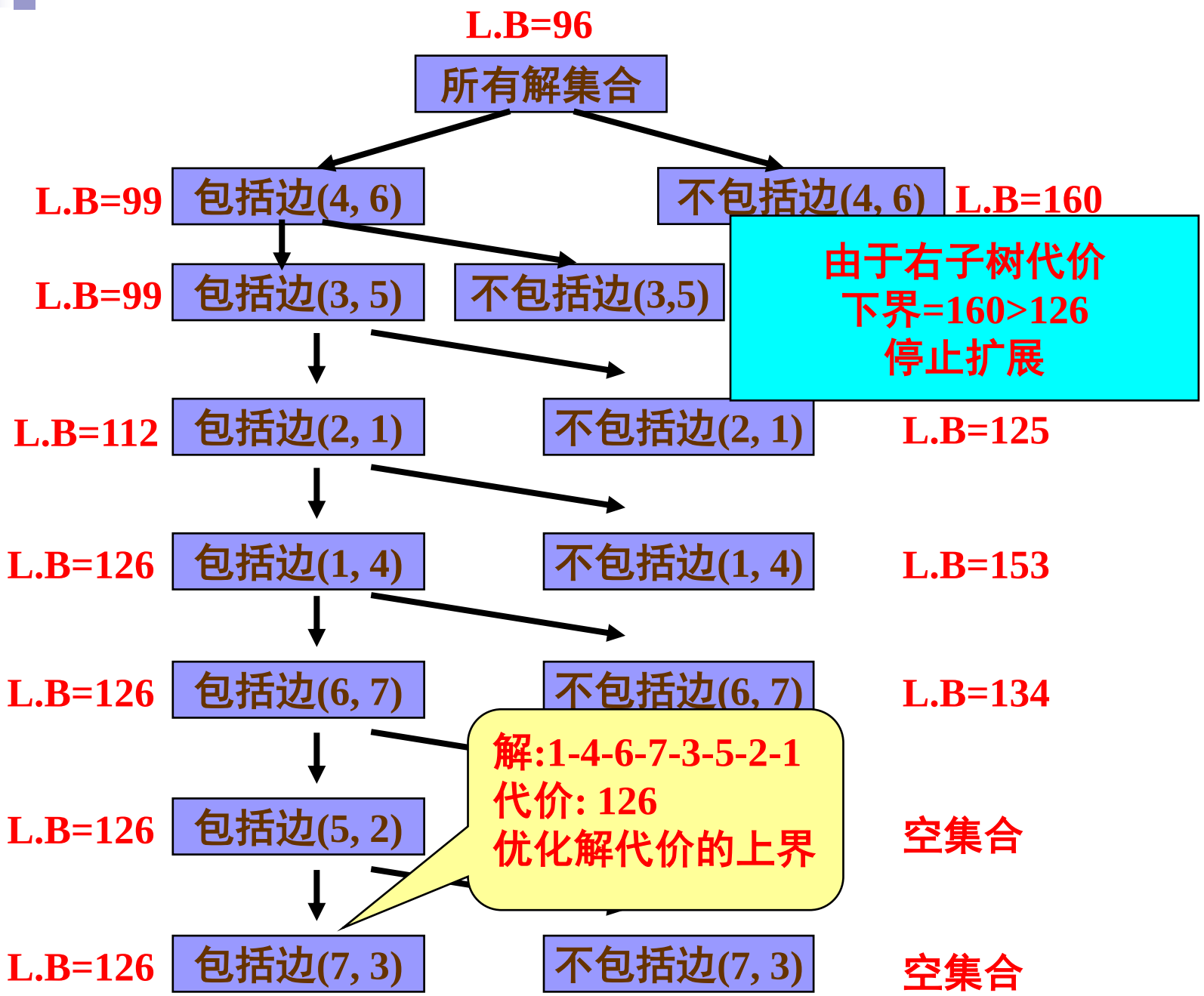
- 计算右子树根的代价下界
  - 矩阵的第4行不包含0
  - 第4行元素减32，右子树根代价下界为: 128+32=160
  - 结果矩阵如下

$$\begin{bmatrix} \infty & 0 & 83 & 9 & 30 & 6 & 50 \\ 0 & \infty & 66 & 37 & 17 & 12 & 26 \\ 29 & 1 & \infty & 19 & 0 & 12 & 5 \\ 0 & 51 & 34 & \infty & 17 & \infty & 48 \\ 3 & 21 & 56 & 7 & \infty & 0 & 28 \\ 0 & 85 & 8 & 42 & 89 & \infty & 0 \\ 18 & 0 & 0 & 0 & 58 & 13 & \infty \end{bmatrix}$$

旅行商问题

- 目前的树为




L.B=96
所有解集合
L.B=99
包括边(4, 6)
不包括边(4, 6)
L.B=160
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(3,5)
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)
不包括边(6, 7)
L.B=134
解:1-4-6-7-3-5-2-1
代价: 126
优化解代价的上界
L.B=126
包括边(5, 2)
空集合
L.B=126
包括边(7, 3)
不包括边(7, 3)
空集合

A*算法

- 基本思想
  - (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.

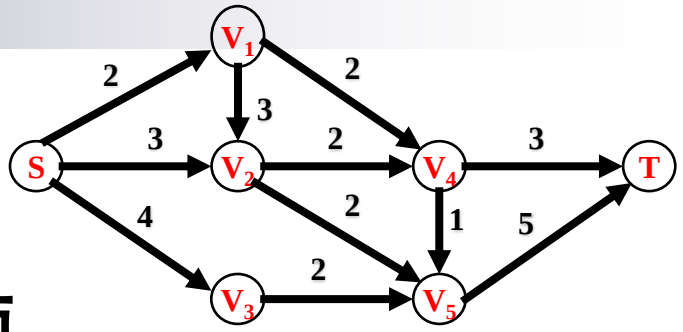
A*算法

- 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
  - h*(n)为多少？
    - 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) ≤ h*(n)总为真
    - f(n)=g(n)+h(n) ≤ g(n)+h*(n)=f*(n)定义为n的代价

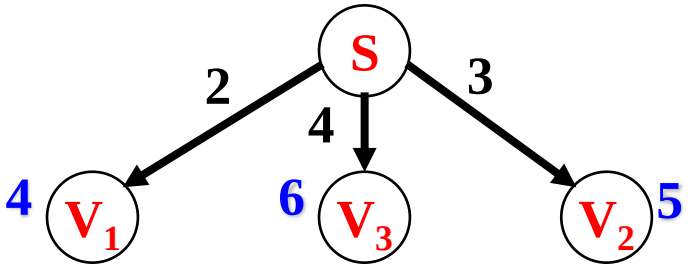
A*算法

- A*算法本质：已经发现的解是优化解
  - 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.
    - 令n是任意扩展到的节点，t是选中目标节点.
    - 往证f(t)=g(t)是优化解代价.
    - 1). A*算法使用Best-first策略，$f(t) \le f(n)$.
    - (2). A*算法使用$h(n) \le h^*(n)$估计规则, $f(t) \le f(n) \le f^*(n)$.
    - (3). {f*(n)}中必有一个为优化解的代价, 令其为f*(s).
    - 我们有$f(t) \le f^*(s)$.
    - (4). t是目标节点h(t)=0, 所f(t)=g(t)+h(t)=g(t) $\le$ f*(s).
    - (5). f(t)=g(t)是一个可能解, $g(t) \ge f^*(s)$, f(t)=g(t)=f*(s).

A*算法



- 应用A*算法求解最短路径问题

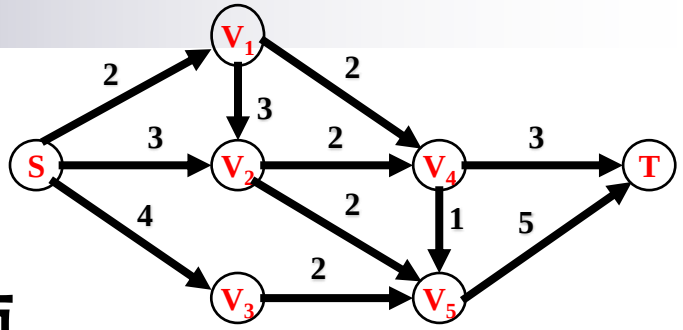


$g(V_1)=2$ $h(V_1)=min\{2,3\}=2$ $f(V_1)=2+2=4$

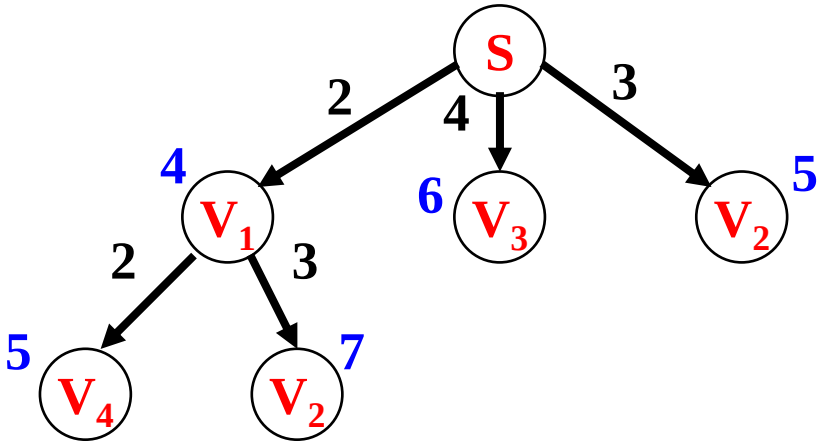
$g(V_3)=4$ $h(V_3)=min\{2\}=2$ $f(V_3)=4+2=6$

$g(V_2)=3$ $h(V_2)=min\{2,2\}=2$ $f(V_2)=2+2=5$

A*算法



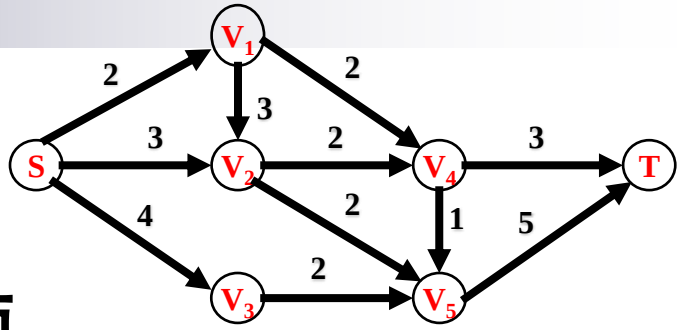
- 应用A*算法求解最短路径问题



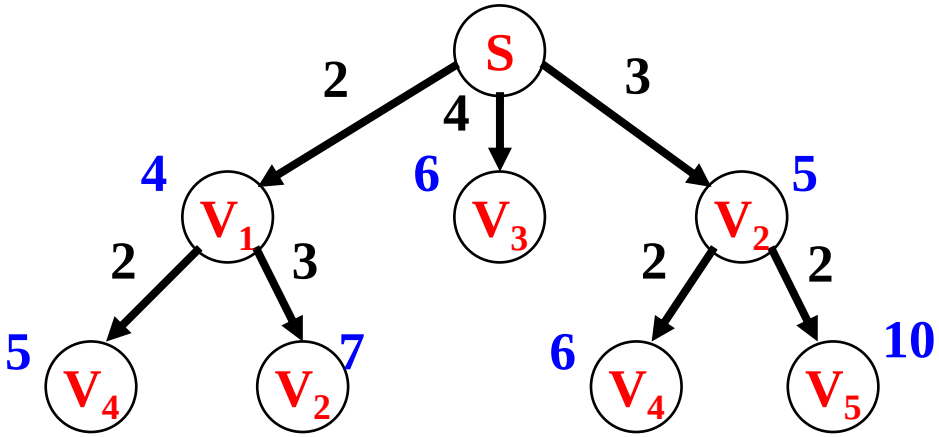
$g(V_4)=2+2=4$　　$h(V_4)=min\{3,1\}=1$　　$f(V_4)=4+1=5$

$g(V_2)=2+3=5$　　$h(V_2)=min\{2,2\}=2$　　$f(V_2)=5+2=7$

A*算法



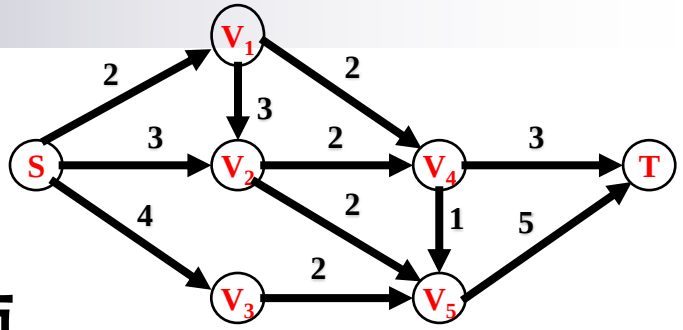
- 应用A*算法求解最短路径问题



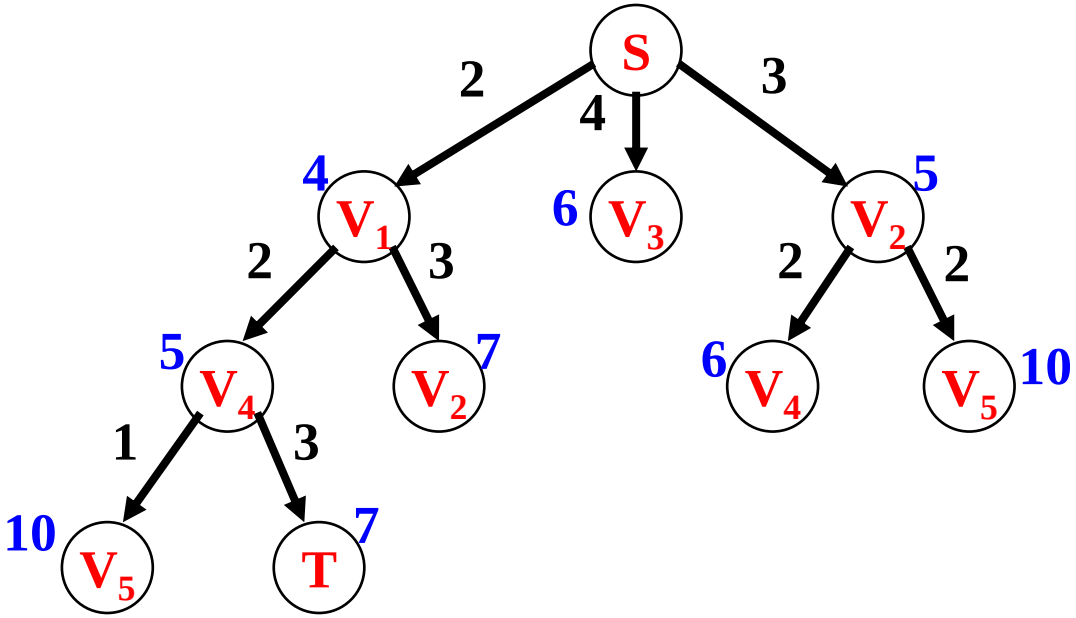
$g(V_4)=3+2=5$　　$h(V_4)=min\{3,1\}=1$　　$f(V_4)=5+1=6$

$g(V_5)=3+2=5$　　$h(V_5)=min\{5\}=5$　　$f(V_2)=5+5=10$

A*算法



- 应用A*算法求解最短路径问题



$g(V_5)=2+2+1=5$　　$h(V_5)=min\{5\}=5$　　$f(V_5)=5+5=10$

$g(T)=2+2+3=7$　　$h(T)=0$　　$f(V_2)=7+0=7$

一副骨牌共 28 张，由以下点数组合构成：

骨牌号	点数	骨牌号	点数	骨牌号	点数	骨牌号	点数
1	0 | 0	8	1 | 1	15	2 | 3	22	3 | 6
2	0 | 1	9	1 | 2	16	2 | 4	23	4 | 4
3	0 | 2	10	1 | 3	17	2 | 5	24	4 | 5
4	0 | 3	11	1 | 4	18	2 | 6	25	4 | 6
5	0 | 4	12	1 | 5	19	3 | 3	26	5 | 5
6	0 | 5	13	1 | 6	20	3 | 4	27	5 | 6
7	0 | 6	14	2 | 2	21	3 | 5	28	6 | 6

这 28 张牌可以摆成 7×8 的点数网格，其对应的骨牌号如下所示：

点数网格

```
6 6 2 6 5 2 4 1
1 3 2 0 1 0 3 4
1 3 2 4 6 6 5 4
1 0 4 3 2 1 1 2
5 1 3 6 0 4 5 5
5 5 4 0 2 6 0 3
6 0 5 3 4 2 0 3
```

骨牌号图

```
28 28 14 7  17 17 11 11
10 10 14 7  2  2  21 23
8  4  16 25 25 13 21 23
8  4  16 15 15 13 9  9
12 12 22 22 5  5  26 26
27 24 24 3  3  18 1  19
27 6  6  20 20 18 1  19
```

请从给定的点数网格$a[1,2,\ldots,7][1,2,\ldots,8]$使用搜索算法求出对应的骨牌号图$b[1,2,\ldots,7][1,2,\ldots,8]$，有可能的话，给出相关剪枝策略。