分支限界法

分支限界法

- 类似于回溯法，在解空间上搜索解
- 求解目标不同：
  - 回溯法：找到满足约束条件的所有解；
  - 分支限界法：找到满足约束条件的一个解，或在满足约束条件的解中找到使某个目标函数值达到极大值或者极小值（某种意义下的最优解）。
- 搜索策略：
  - 广度优先或最小耗费优先。
  - 每一个活节点计算限界值，选择最有利的节点作为扩展节点，尽快找到一个最优解。

学习要点

- 分支限界法的算法框架
  - 队列式（FIFO）分支限界法
  - 优先队列式分支限界法——堆式分支限界法
- 应用分支限界法解决
  - 单源最短路径问题
  - 装载问题；
  - 0-1背包问题；
  - 旅行售货员问题
  - 批处理作业调度问题

活节点组织成一个队列，按照先进先出原则选择下一个节点为当前扩展节点。

活节点组织成一个优先队列，按照优先队列中规定的节点优先级选择优先级最高的下一个节点为当前扩展节点

哈尔滨工程大学

分支限界法基础

分支限界法基础

- 8-Puzzle问题
  - 输入：具有8个编号小方块的魔方

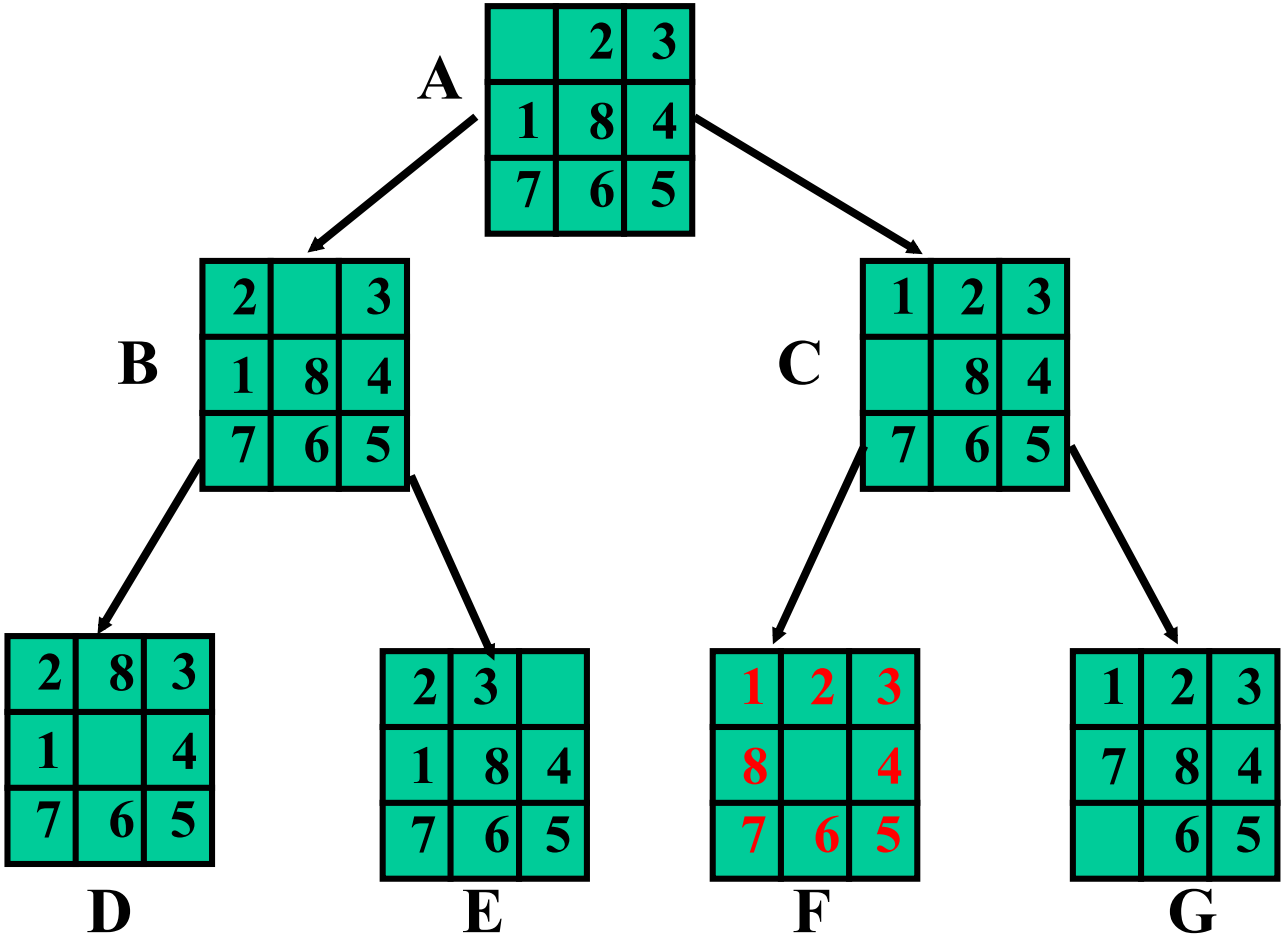
	2	3
1	8	4
7	6	5

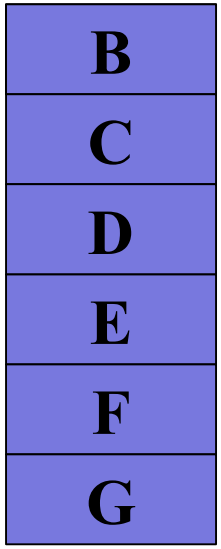
  - 输出：移动序列，经过这些移动，魔方达目标状态

1	2	3
8		4
7	6	5

分支限界法基础

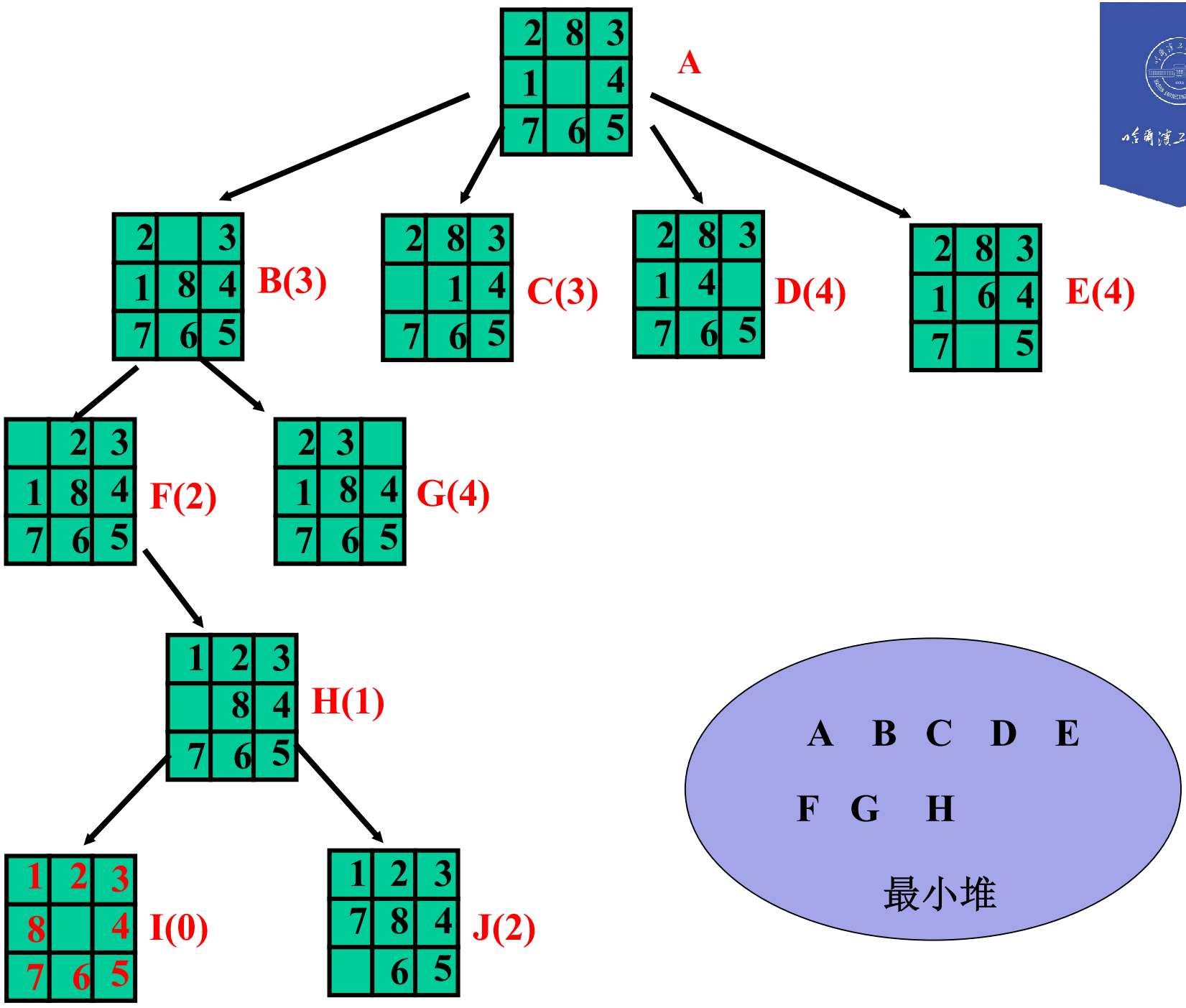
- 队列式（FIFO）分支限界法
  - 广度优先搜索解空间树





分支限界法基础

- 优先队列式分支限界法——堆式分支限界
  - 为解空间树中的每个节点指定一个优先级——测度函数
  - 每次扩展树中优先级最高的节点
  - 用最大（最小）堆来保存待搜索节点
- 8-Puzzle问题
  - 测度函数 $f(v)$=节点v中处于错误位置的方块数
  - 每次扩展 $f(v)$ 值最小的节点


2 8 3
1 4
7 6 5
A
2 3
1 8 4
7 6 5
B(3)
2 8 3
1 4
7 6 5
C(3)
2 8 3
1 4
7 6 5
D(4)
2 8 3
1 6 4
7 5
E(4)
2 3
1 8 4
7 6 5
F(2)
2 3
1 8 4
7 6 5
G(4)
1 2 3
8 4
7 6 5
H(1)
1 2 3
8 4
7 6 5
I(0)
1 2 3
7 8 4
6 5
J(2)
A B C D E
F G H
最小堆

分支限界法基础

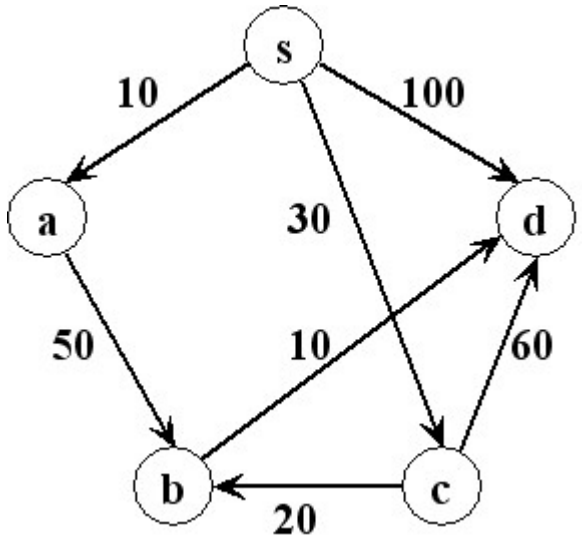
- 思想
  - 广度优先或者优先级优先搜索解空间树
- 实现
  - 每个节点只有一次机会成为扩展节点
  - 一旦节点v成为扩展节点
    - 将 v 的所有孩子加入到队列或者堆中
    - 接着选取队列顶或堆顶节点作为下一个扩展节点
- 效率
  - 耗时比回溯法少
  - 需要空间比回溯法多

哈爾濱工程大學

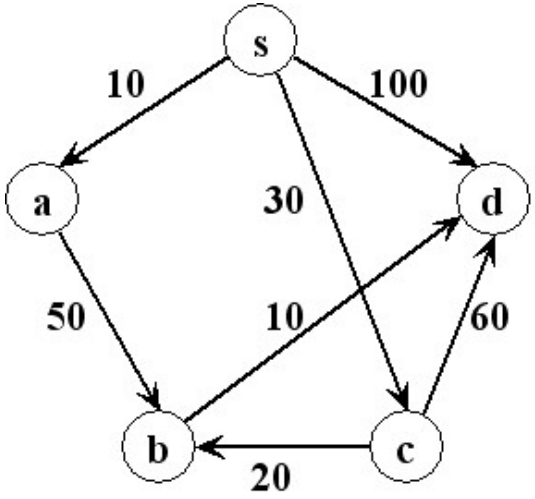
单源最短路径问题

单源最短路径问题

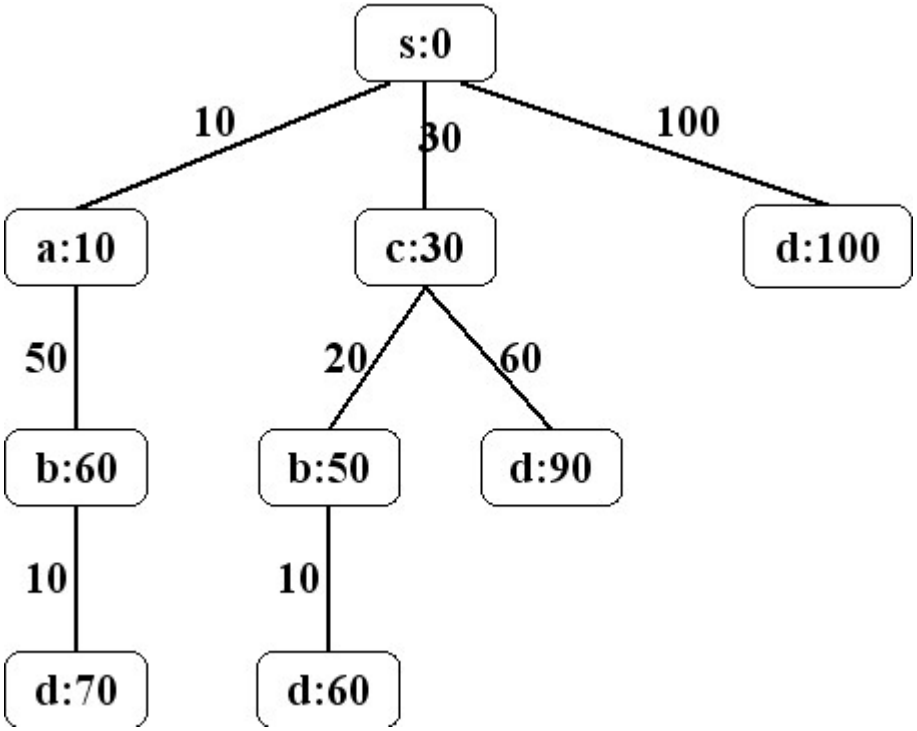
- 输入
  - 有向带权图G=(V, E)
  - 图中顶点s
- 输出
  - s到图中其它顶点的最短路径



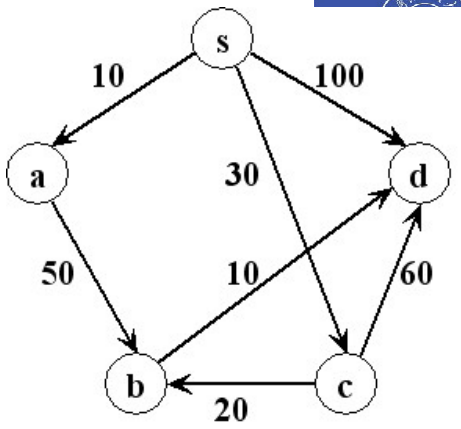
单源最短路径问题



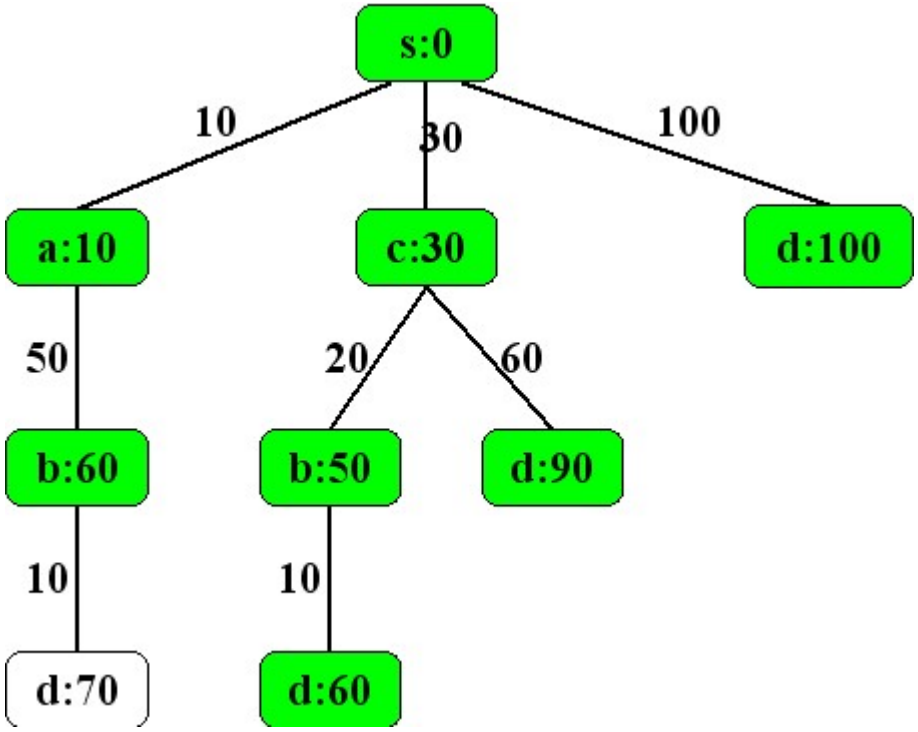
- 解空间树



单源最短路径问题



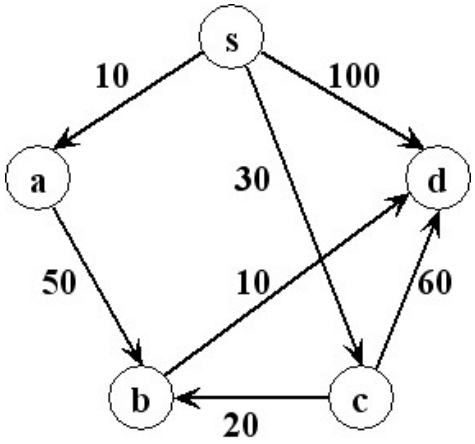
- 优先队列式分支限界法
  - 优先级测度：当前路径长度 （最小堆）

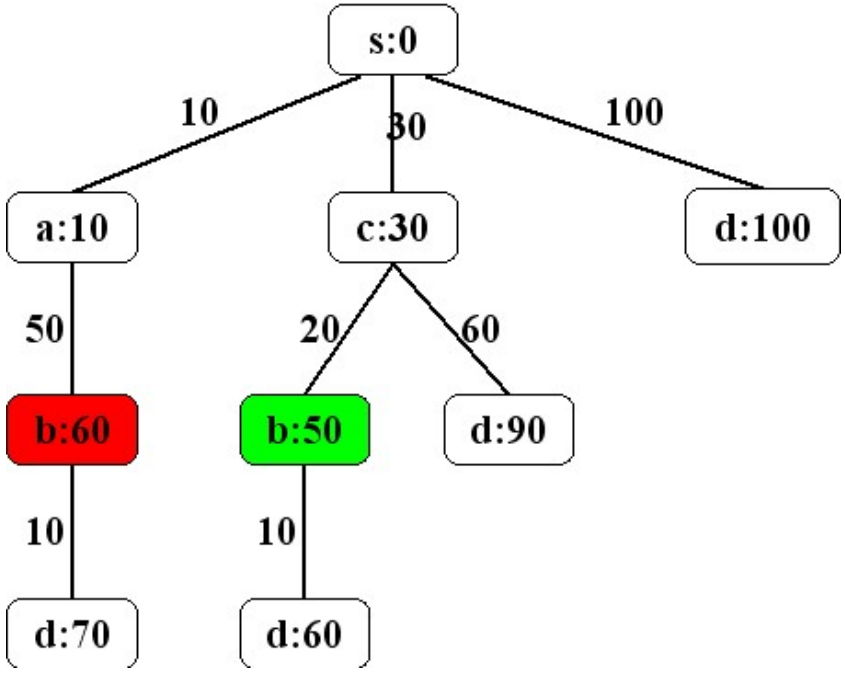


v	pre[v]	dist[v]
a	s	10
b	c	50
c	s	30
d	b	60

单源最短路径问题

- 剪枝策略
  - 节点**[b:50]**控制**[b:60]**
  - 将**[b:60]** 及其子树剪去
  - 当前扩展节点有儿子**[v: x]**
    - 如果x>=dist[v]，不扩展v
    - 如果x<dist[v]，扩展v，将v的孩子加入堆中





哈尔滨工程大学

0-1背包问题

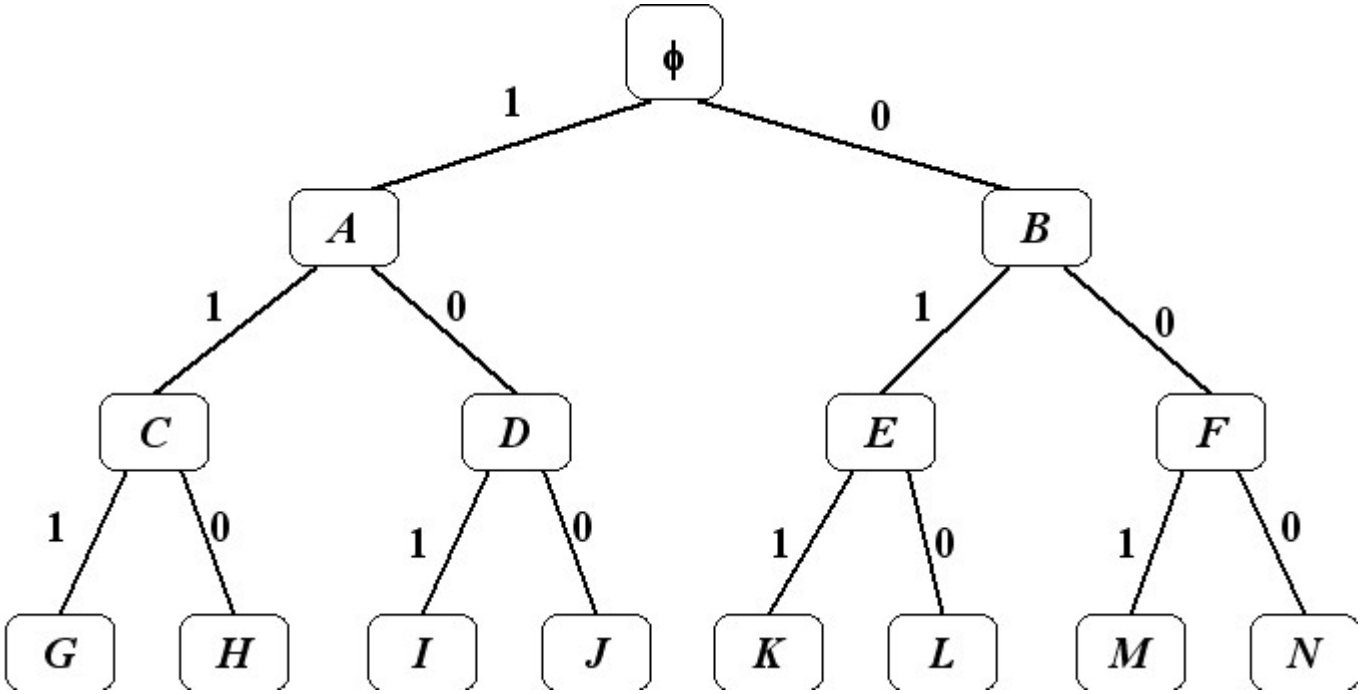
0-1背包问题

- 输入：
  - $\{<w_1, v_1>, <w_2, v_2>, \ldots, <w_n, v_n>\}$和$C$
- 输出：
  - $(x_1, x_2, \ldots, x_n)$，$x_i \in \{0, 1\}$满足 $\sum_{i=1}^{n} w_i x_i \le C$
- 优化目标：$\max \sum_{i=1}^{n} v_i x_i$

0-1背包问题

w={16，15，15}
v={45，25，25}
背包容量30

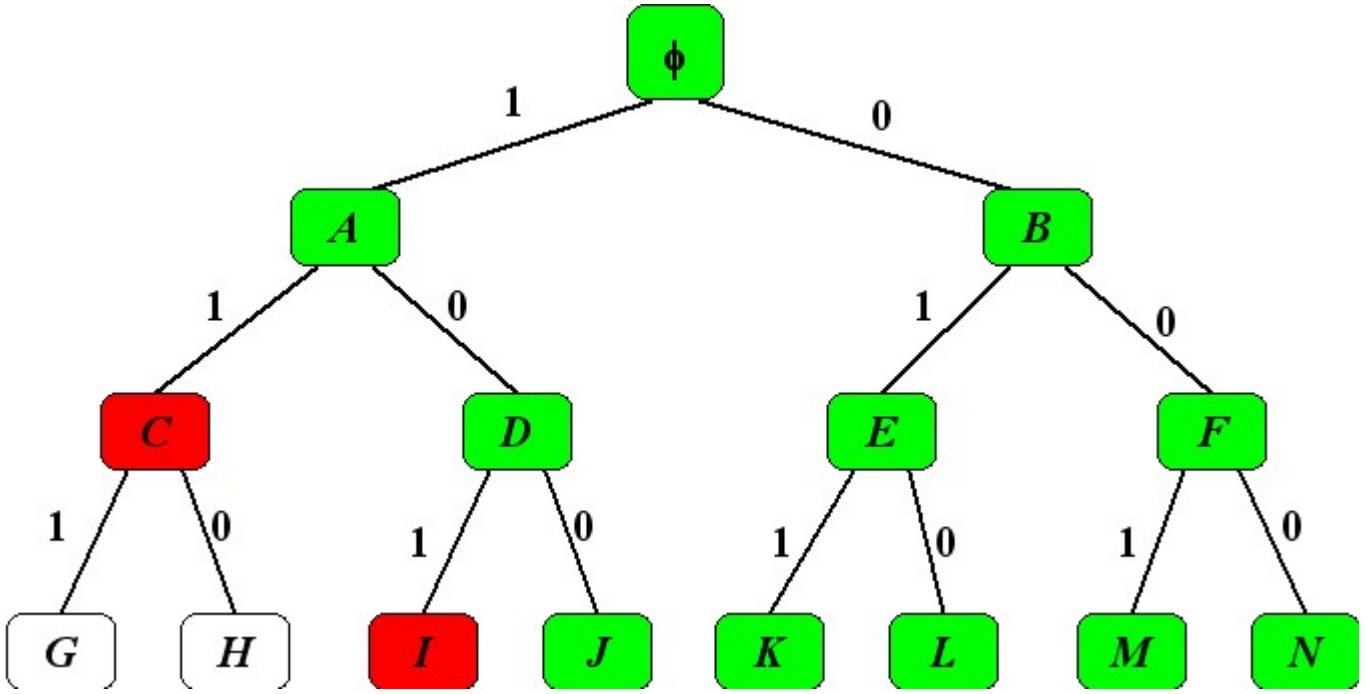
- 解空间树



0-1背包问题

w={16，15，15}
v={45，25，25}
背包容量30

- FIFO队列式分支限界法



FIFO队列

{ϕ}

{A, B}

{D, E, F}

{J, K, L, M, N}

节点C对应：$x_1=1, x_2=1$

节点K对应：$x_1=0, x_2=1, x_3=1$

0-1背包问题

物品按价重比排序为1, …, n

- 对于树中的第i层节点V
  - 已经完成了对物品{1,…, i}的取舍，剩余物品为{i+1, ..., n}
  - 设已选择的物品价值为p，背包剩余容量为C’
- 限界函数bound(V) = p + q
  - 其中q是针对输入{i+1, ..., n}和C’的一般背包问题的最优解
  - 可以按价重比顺序依次向背包中装入物品得到q
- **以V为根的子树中解的价值不会超过bound(V)**

0-1背包问题

- 优先队列式分支限界法
  - 优先级测度：bound(V)　——最大堆
  - 直到某个叶节点x成为扩展节点结束
    - 叶节点x的bound(x)等于拿走物品的总价值（解）
    - 堆中剩余节点及其子树的解不会超过bound(x)

0-1背包问题

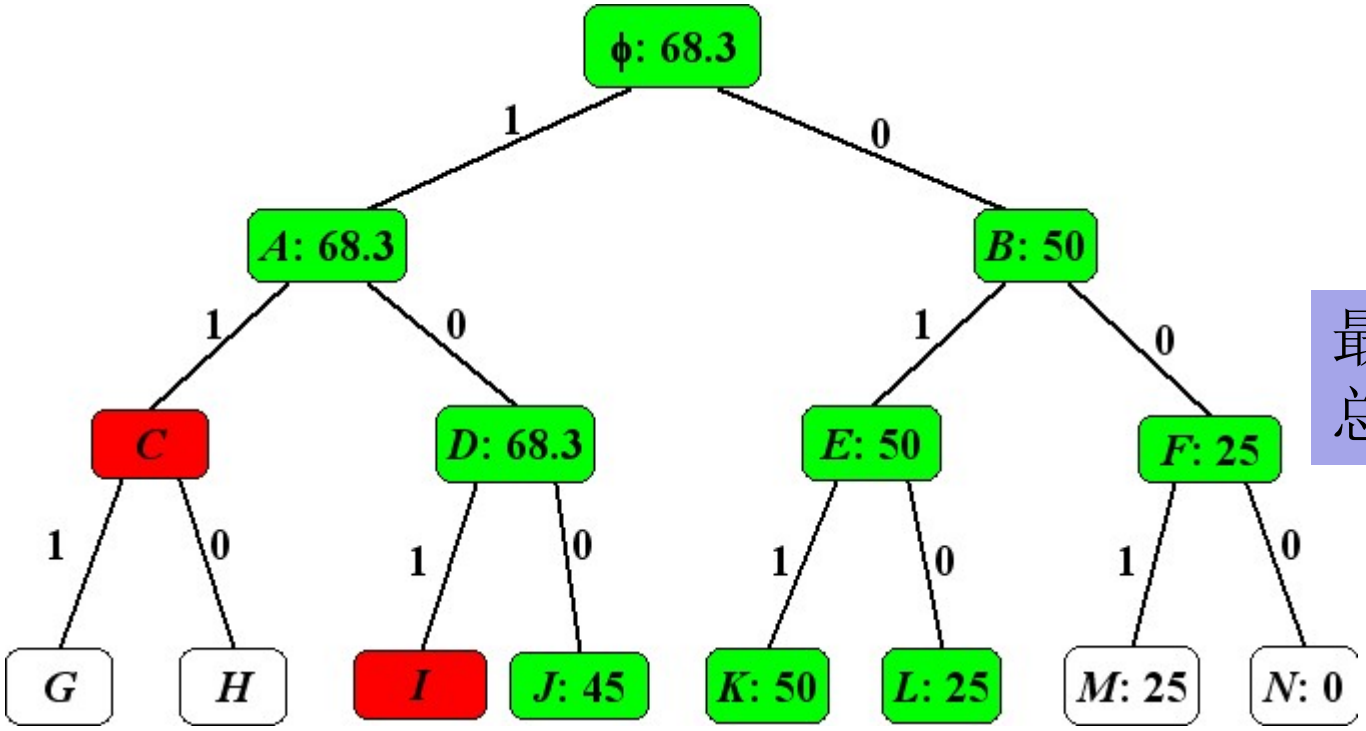
w={16，15，15}
v={45，25，25}
背包容量30

- 优先队列式分支限界法

  bound(V) = p + q

  - 优先级测度：bound(V)　——最大堆

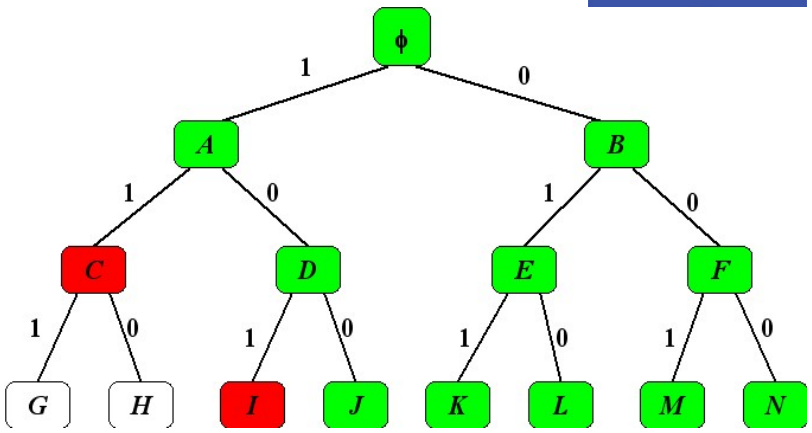
最大堆



{K, J, L, F}

最优解：K (0, 1, 1)
总价值：50

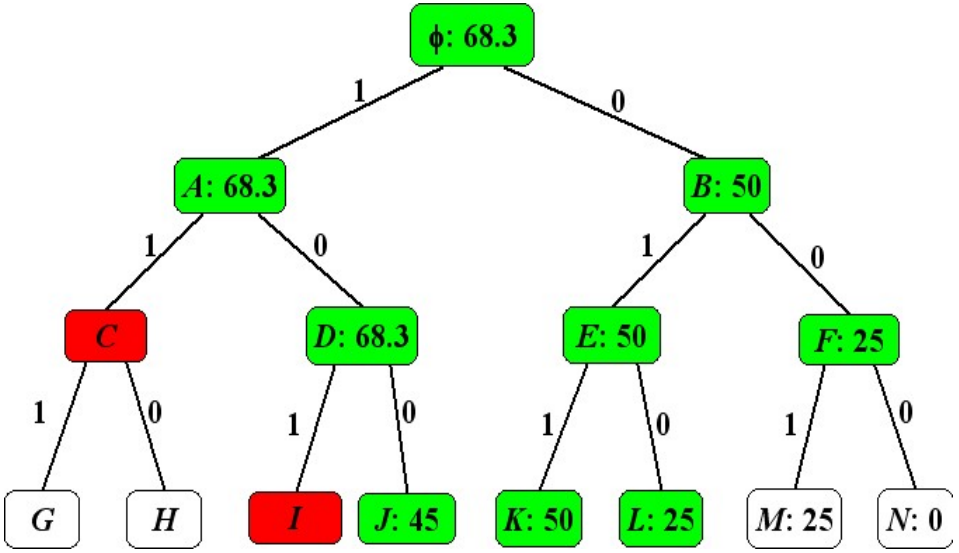
0-1背包问题



- 队列（堆）中的每个元素
  - 存储已经获得的部分解
    - D入队列：存储x[1]=1, x[2]=0
    - K入队列：存储x[1]=0, x[2]=1, x[3]=1
  - 存储已构造的解空间树
    - B入队列：存储B的父亲为ϕ，B是其右孩子
    - E入队列：存储E的父亲为B，E是其左孩子
    - K入队列：存储K的父亲为E，K是其左孩子
    - K对应的解：x[3]=1, x[2]=1, x[1]=0

0-1背包问题

- 分支限界法优化
  - 限界函数：bound(v)
  - 当前的最优值：bestp
  - 如果bound(v)≤bestp
    - 剪去v及其子树



哈尔滨工程大学

装载问题

装载问题

- 输入
  - n个集装箱，其中集装箱i的重量为w_i
  - 载重量分别为C_1和C_2的轮船

$$\sum_{i=1}^{n} w_i \le C_1 + C_2$$

- 输出
  - （是否有）合理的装载方案将所有集装箱装上船
- 等价于

$$\max \sum_{i=1}^{n} w_i x_i$$

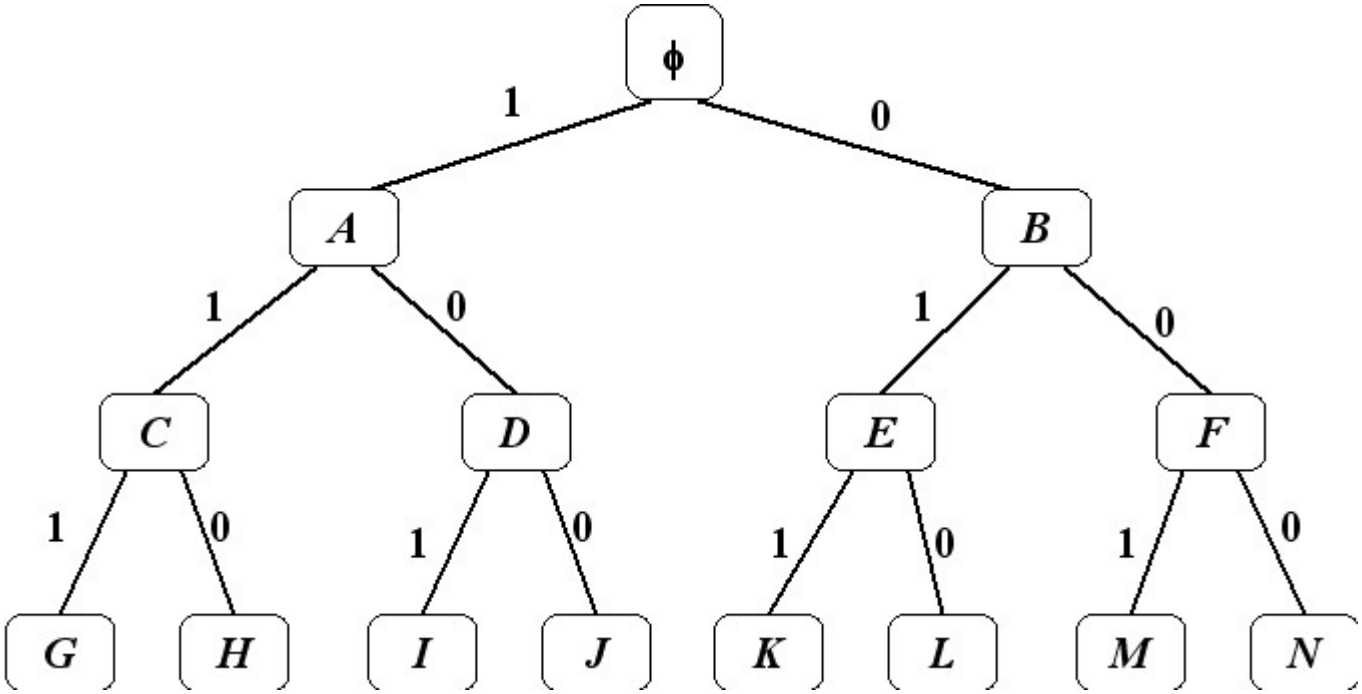
$$\text{s.t.} \quad \begin{cases} \sum_{i=1}^{n} w_i x_i \le C_1 \\ x_i \in \{0,1\},\ 1 \le i \le n \end{cases}$$

特殊的0-1背包问题：每种物品的价值等于重量

装载问题

w={16，15，15}
C_1载重量30

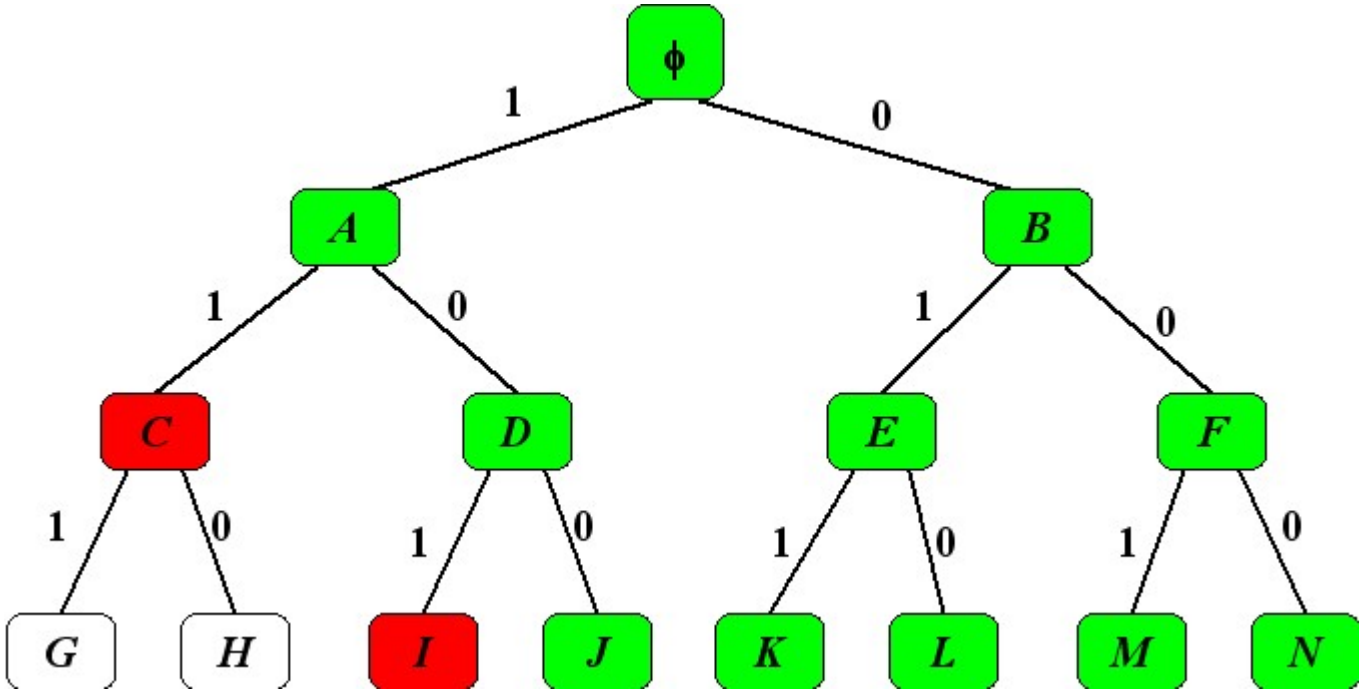
- 解空间树



装载问题

w={16，15，15}
C_1载重量30

- FIFO队列式分支限界法



装载问题

- 解空间树的第i层节点v
  - 已经完成了对集装箱$\{1,\ldots, i\}$的取舍，剩余集装箱为$\{i+1, \ldots, n\}$
  - 已经装上第一艘船的集装箱重量和为p
  - 剩余集装箱的重量和为r
  - 限界函数**bound(v)**=p+r
- 以v为根的子树中的解的总重量不会超过bound(v)

装载问题

- 优先队列式分支限界法
  - 优先级测度：bound(v)　——最大堆
  - 直到某个叶节点x成为扩展节点结束
    - 叶节点x的bound(x)等于拿走物品的总重量
    - 堆中剩余节点及其子树的总重量不会超过bound(x)

装载问题

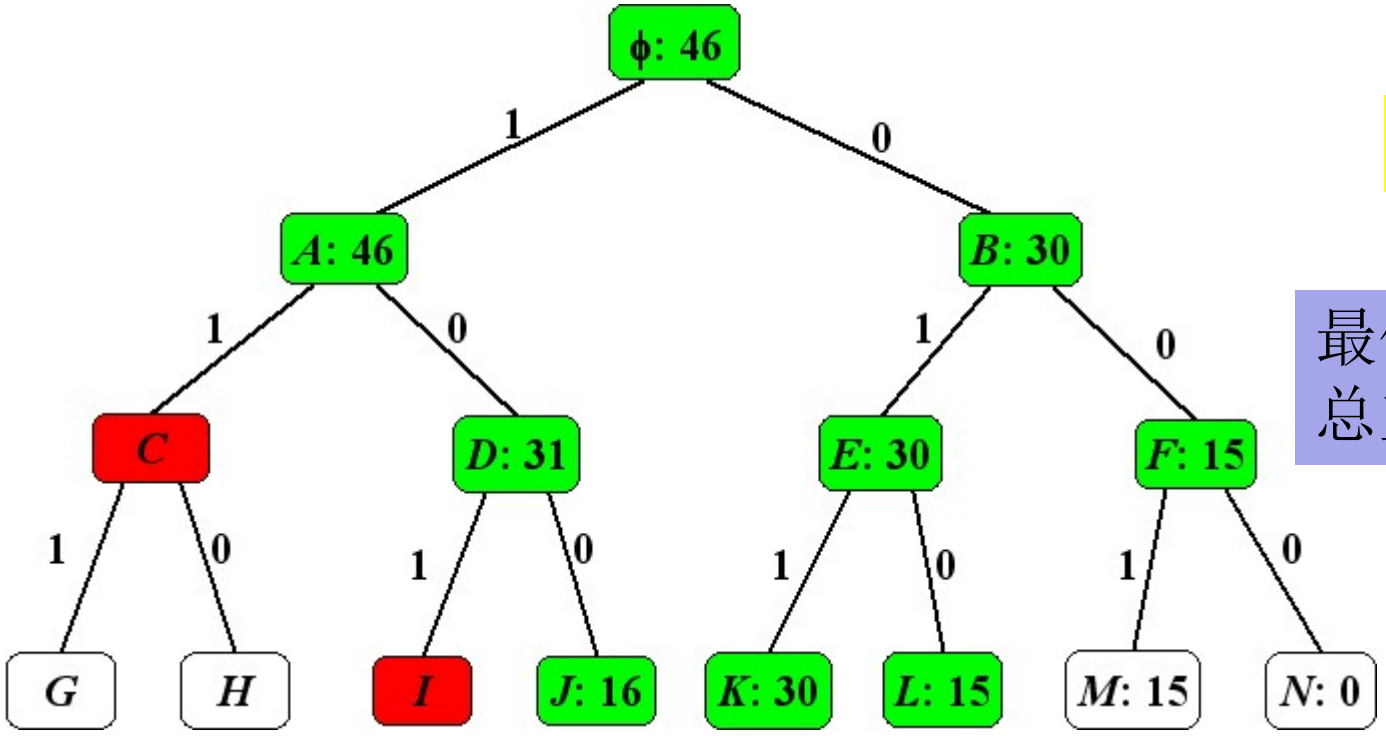
- 优先队列式分支限界法

w={16，15，15}
C_1载重量30

bound (v) =p+r

最大堆

{K, J, F, L}



最优解：K (0, 1, 1)
总重量：30

哈尔滨工程大学

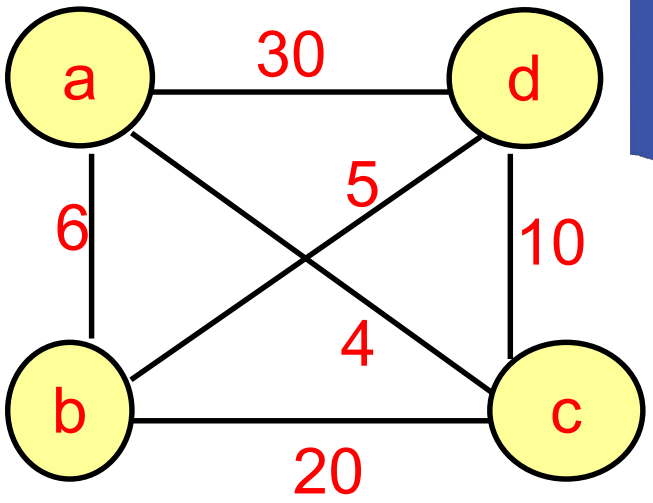
旅行商问题

旅行商问题

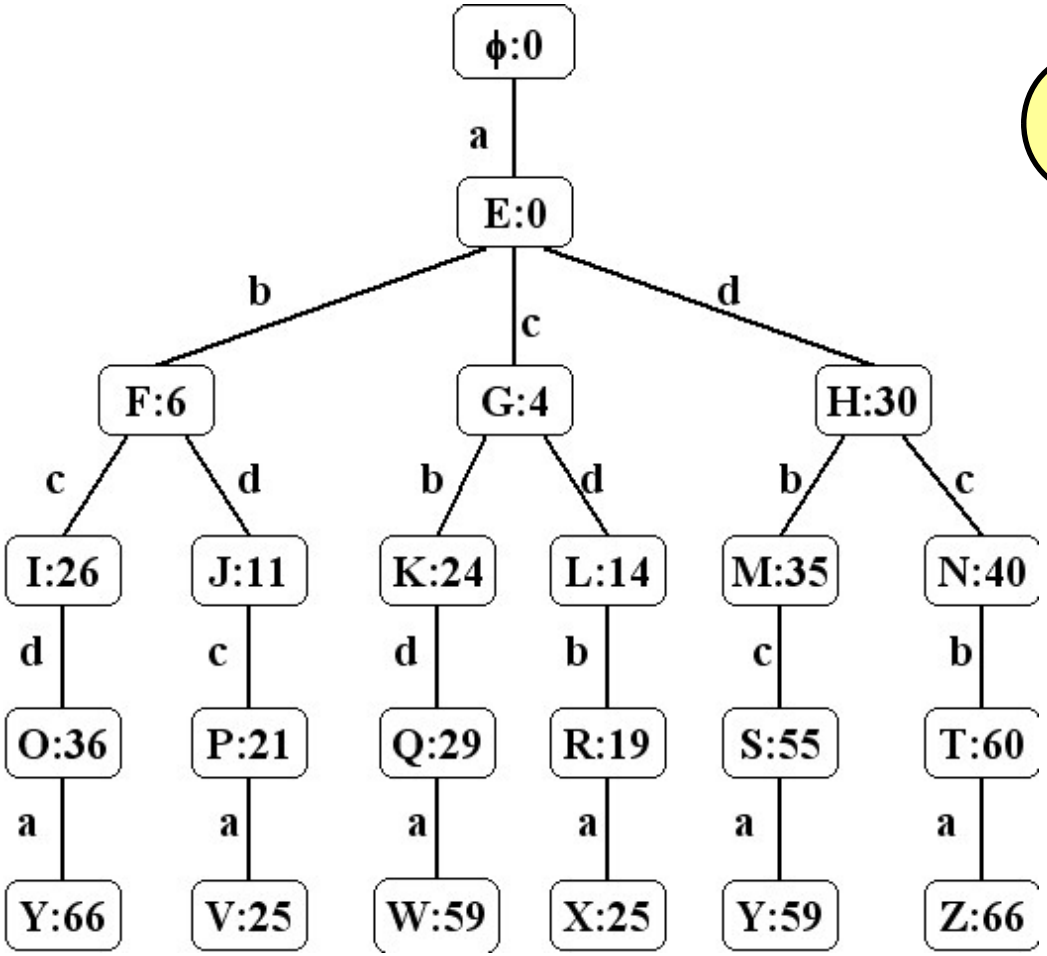
- 输入
  - 完全无向带权图G=(V, E)
    - |V|=n, |E|=m
    - 对于E中的某条边e，其长度为c(e)
- 输出
  - 最短的哈密尔顿回路
    - 经过每个节点一次且仅一次的回路

NP难问题

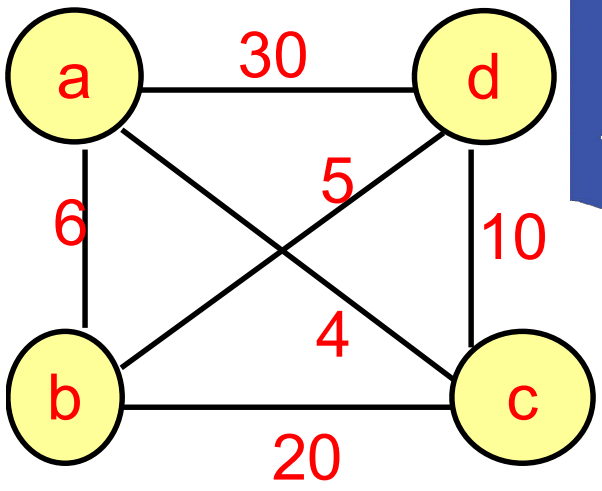
旅行商问题


a
d
b
c
30
6
5
10
4
20

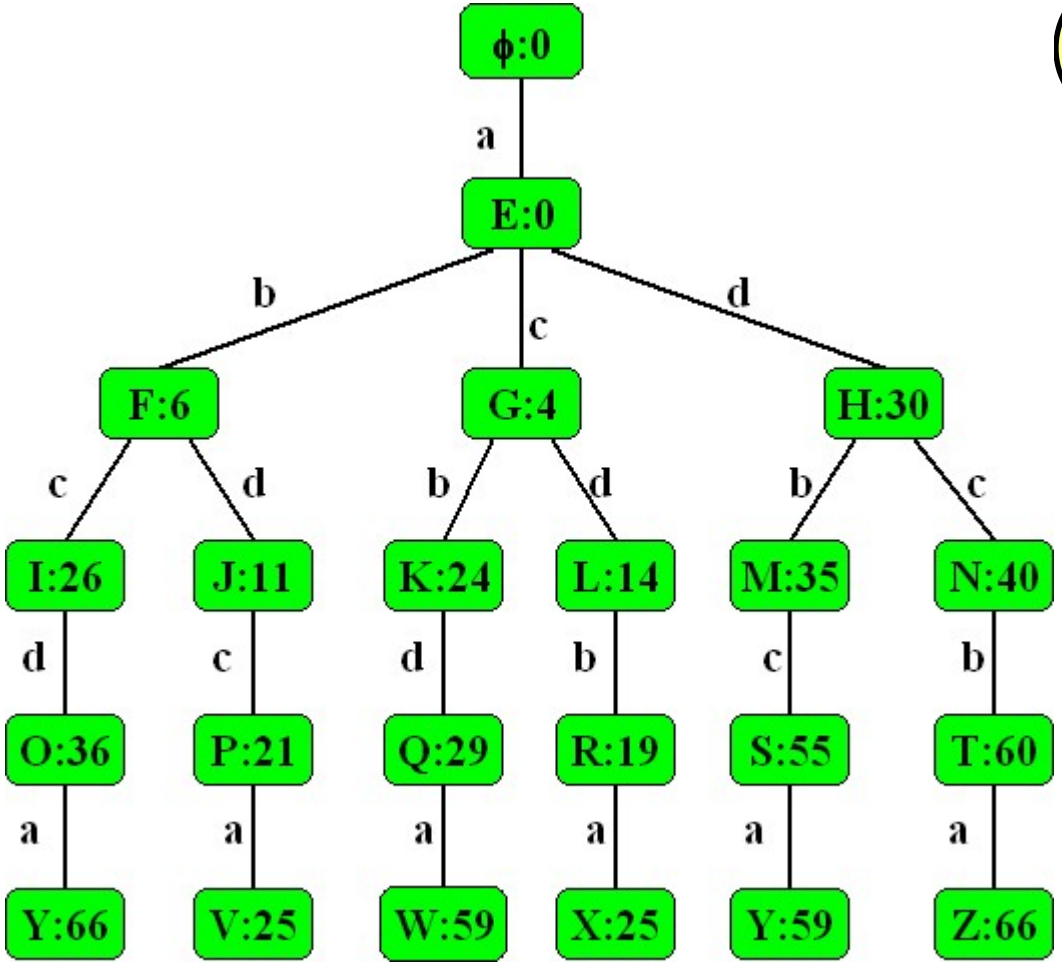
- 解空间树


ϕ:0
a
E:0
b
c
d
F:6
G:4
H:30
c
d
b
d
b
c
I:26
J:11
K:24
L:14
M:35
N:40
d
c
d
b
c
b
O:36
P:21
Q:29
R:19
S:55
T:60
a
a
a
a
a
a
Y:66
V:25
W:59
X:25
Y:59
Z:66

旅行商问题



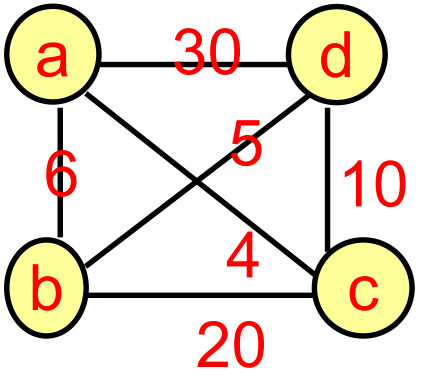
- FIFO队列式分支限界法



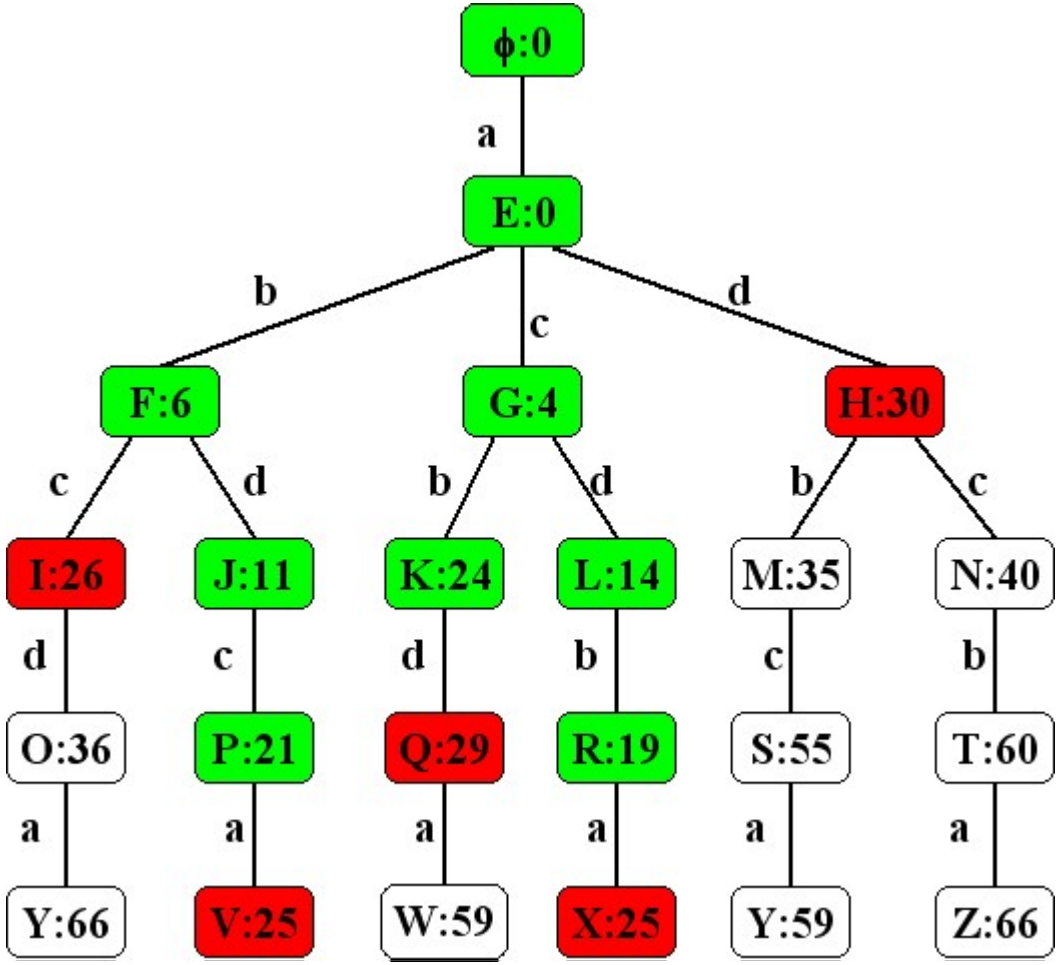
旅行商问题

- 堆式分支限界法1
  - 优先级测度：当前路径长度（最小堆）
  - bestp：当前最优解
  - 剪去当前代价大于等于bestp的节点及其子树

旅行商问题



- 堆式分支限界法1



最小堆

{ { { {H} } } }

最优解：X (acdba)
bestp=25

旅行商问题

- 对于树中的第 i 层节点W
  - 路径上已选顶点为$\{v_1, v_2, \ldots, v_i\}$
    - 当前路径的长度为 ***P***
  - 剩余顶点为$\{v_{i+1}, \ldots, v_n\}$
    - 连接 v_j 的最短出边的长度为**Minout**(v_j)

- $\text{bound(W)=P+} \sum_{j=i}^{n} \text{Minout}(v_j)$

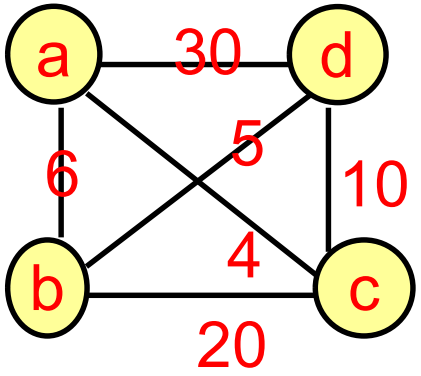
  - 以W为根的子树中的解的代价不少于bound(W)

旅行商问题

- 堆式分支限界法2
  - 优先级测度：bound(W) —— 最小堆
  - 直到某个叶节点Y成为扩展节点
    - bound(Y)等于Y的路径长度
    - 堆中其它节点的代价都大于bound(Y)
- 优化
  - bestp：当前最优解
  - 剪去当前代价大于等于bestp的节点及其子树

旅行商问题

- 堆式分支限界法2



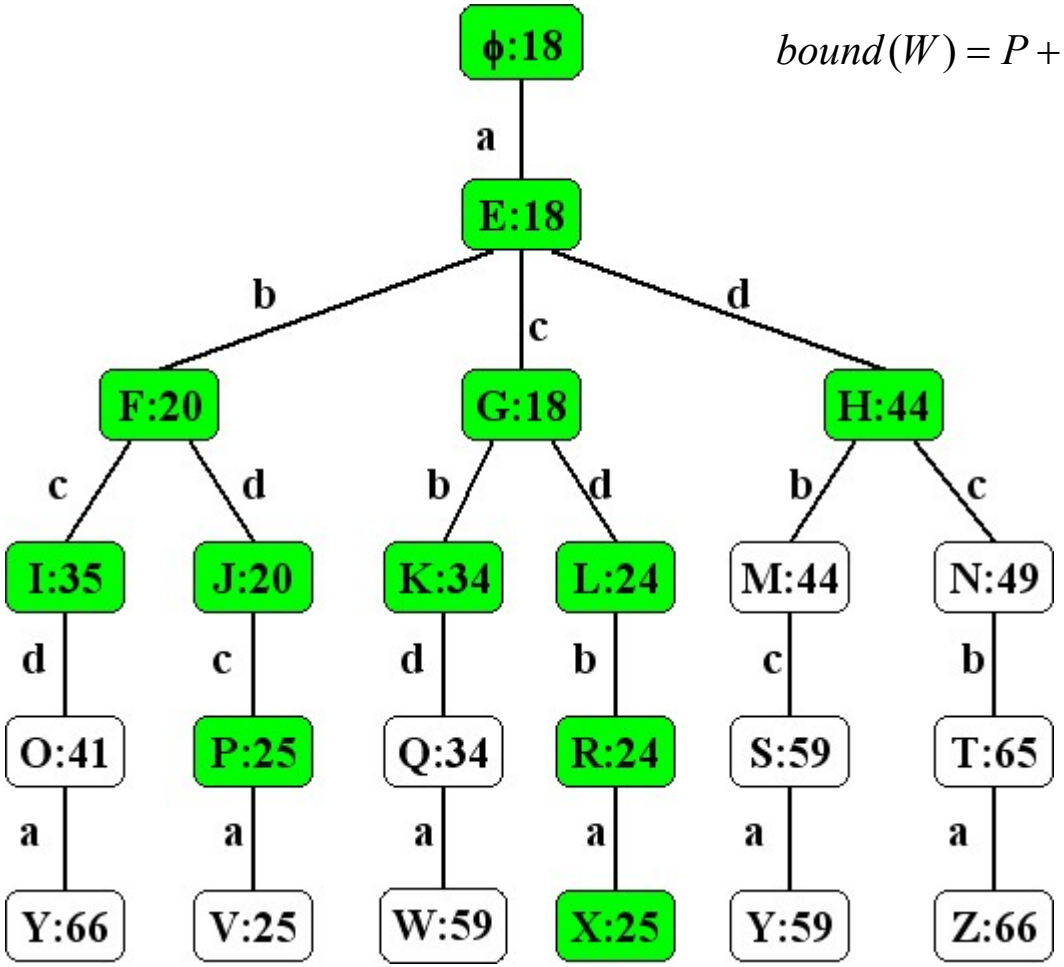
Minout(a)=4
Minout(b)=5
Minout(c)=4
Minout(d)=5

$$bound(W) = P + \sum_{j=i}^{n} \text{Minout}(v_j)$$

最小堆

{I, P, K, X, H}

最优解：X (acdba)
bestp=25



批处理作业调度问题

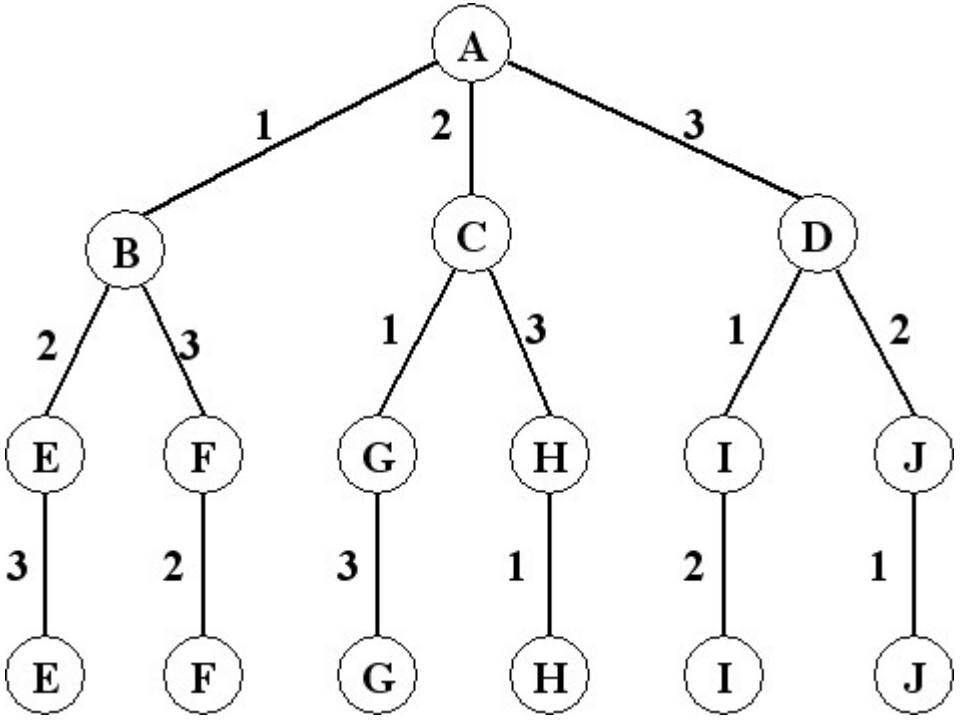
批处理作业调度问题

- 输入
  - n个作业$\{1, \ldots, n\}$
  - 两台机器（M1和M2）
    - 作业 i 在M1和M2上的处理时间分别为 $a[i]$ 和 $b[i]$
    - 每个作业必须现由M1处理，再由M2处理
- 输出
  - 作业调度方案使得总等待时间最小
    - 作业 i在M1和M2上的完成时间分别为 $A[i]$ 和 $B[i]$
    - 总等待时间为 $\sum_{i=1}^{n} B[i]$

批处理作业调度问题

- 解空间树

作业	*a*[*i*]	*b*[*i*]
Job 1	2	1
Job 2	3	1
Job 3	2	3



批处理作业调度问题

- 对于树中第 i 层节点V
  - V已经安排了作业$\{J_1, J_2, \ldots, J_i\}$
    - 已安排的作业的等待时间为 $\sum_{x=1}^{i} B[J_x]$

如果$a[J_x]$在$x \geq i+1$时按非递减序排列

S_1取得极小值S'_1

$\{J_{i+1}, \ldots, J_n\}$的总等待时间≥ S'_1 ，

- 如果从J_{i+1}开始机器1没有空闲，则

$$B[J_{i+1}] \geq A[J_i] + a[J_{i+1}] + b[J_{i+1}]$$

$$B[J_{i+2}] \geq A[J_i] + a[J_{i+1}] + a[J_{i+2}] + b[J_{i+2}]$$

$$\vdots$$

$$B[J_n] \geq A[J_i] + a[J_{i+1}] + a[J_{i+2}] + \ldots + a[J_n] + b[J_n]$$

- $\{J_{i+1}, \ldots, J_n\}$的总等待时间不少于

$$S_1 = (n-i)A[J_i] + \sum_{x=i+1}^{n}(n-x+1)a[J_x] + \sum_{x=i+1}^{n} b[J_x]$$

如果$b[J_x]$在$x \geq i+1$时按非递减序排列

S_2取得极小值S'_2

$\{J_{i+1}, \ldots, J_n\}$的总等待时间≥ S'_2 ，

- 如果从J_{i+1}开始机器2没有空闲，则

$$B[J_{i+1}] \geq B[J_i] + b[J_{i+1}]$$

$$B[J_{i+2}] \geq B[J_i] + b[J_{i+1}] + b[J_{i+2}]$$

$$\vdots$$

$$B[J_n] \geq B[J_i] + b[J_{i+1}] + b[J_{i+2}] + \ldots + b[J_n]$$

- $\{J_{i+1}, \ldots, J_n\}$的总等待时间≥

$$S_2 = (n-i)B[J_i] + \sum_{x=i+1}^{n} (n-x+1)b[J_x]$$

批处理作业调度问题

- 对于树中第 i 层节点V
  - V已经安排了作业$\{J_1, J_2, \ldots, J_i\}$
  - 以其为根的子树的叶节点的总等待时间下界为

$$\text{bound(V)} = \sum_{x=1}^{i} B[J_x] + \max\{S'_1, S'_2\}$$

  - $a[J_x]$在$x \geq i+1$时按非递减序排列

$$S'_1 = (n-i)A[J_i] + \sum_{x=i+1}^{n}(n-k+1)a[J_x] + \sum_{x=i+1}^{n} b[J_x]$$

  - $b[J_x]$在$x \geq i+1$时按非递减序排列

$$S'_2 = (n-i)B[J_i] + \sum_{x=i+1}^{n}(n-k+1)b[J_x]$$

批处理作业调度问题

- 堆式分支限界法
  - 优先级测度：bound(V) —— 最小堆
  - 直到某个叶节点Y成为扩展节点
    - bound(Y)等于Y的总等待时间
    - 堆中其它节点的等待时间都大于等于bound(Y)

总结

- 理解分支限界法的剪枝搜索策略
- 掌握分支限界法的算法框架
  - 队列式(FIFO)分支限界法
  - 优先队列式分支限界法