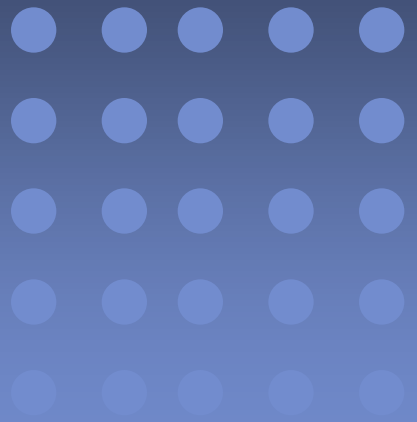
第6章 分支限界法



提纲

- 分支限界法的基本思想
- 最大团问题
- 旅行售货员问题
- 总结

相关概念

组合优化问题的相关概念

目标函数（极大化或极小化）

约束条件

搜索空间中满足约束条件的解称为可行解

使得目标函数达到极大(或极小)的解称为最优解

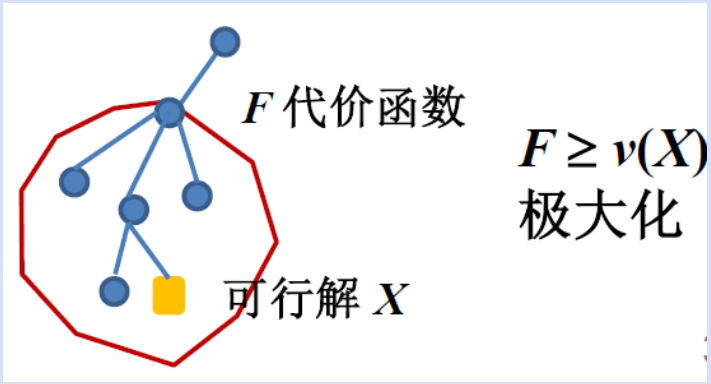
相关概念

代价函数（以极大值问题为例）

计算位置：搜索树的结点

值：极大化问题是以该点为根的子树所有可行解的值的上界

性质：对极大化问题父结点代价不小于子结点的代价

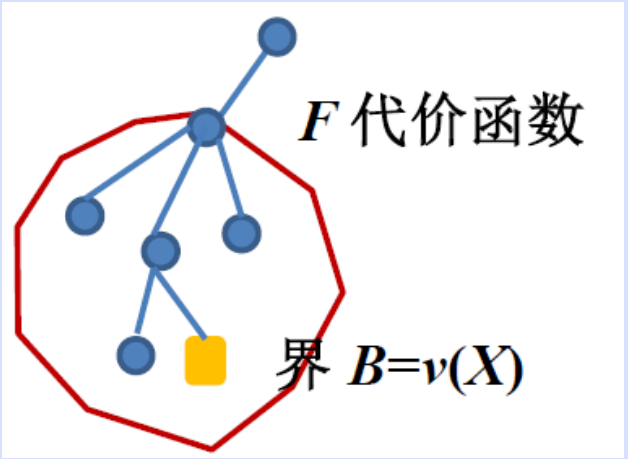


界

含义：当前得到可行解的目标函数的最大值

初值：极大化问题初值为0

更新：得到更好的可行解时



相关概念

分支限界

停止分支回溯父结点的依据：

1. 不满足约束条件
2. 对于极大化问题，代价函数值小于当前界

界的更新

对极大化问题，如果一个新的可行解的优化函数值大于当前的界，则把界更新为该可行解的值

引例

- **0-1背包问题**

背包问题：4种物品，重量*wi*与价值*vi*分别为

*v*1=1,*v*2=3,*v*3=5,*v*4=9　　*w*1=2, *w*2=3, *w*3=4, *w*4=7

背包重量限制为10

搜索空间：子集树，一个物品要么装入，要么不装入

预处理：将输入按照单位重量价值的顺序排序

代价函数与分支策略确定

结点$<x_1, x_2, \ldots, x_k>$的代价函数

$$\sum_{i=1}^{k} v_i x_i + (b - \sum_{i=1}^{k} w_i x_i) \frac{v_{k+1}}{w_{k+1}}$$

若对某个$j > k$有$b - \sum_{i=1}^{k} w_i x_i \geq w_j$

$\sum_{i=1}^{k} v_i x_i$　　否则

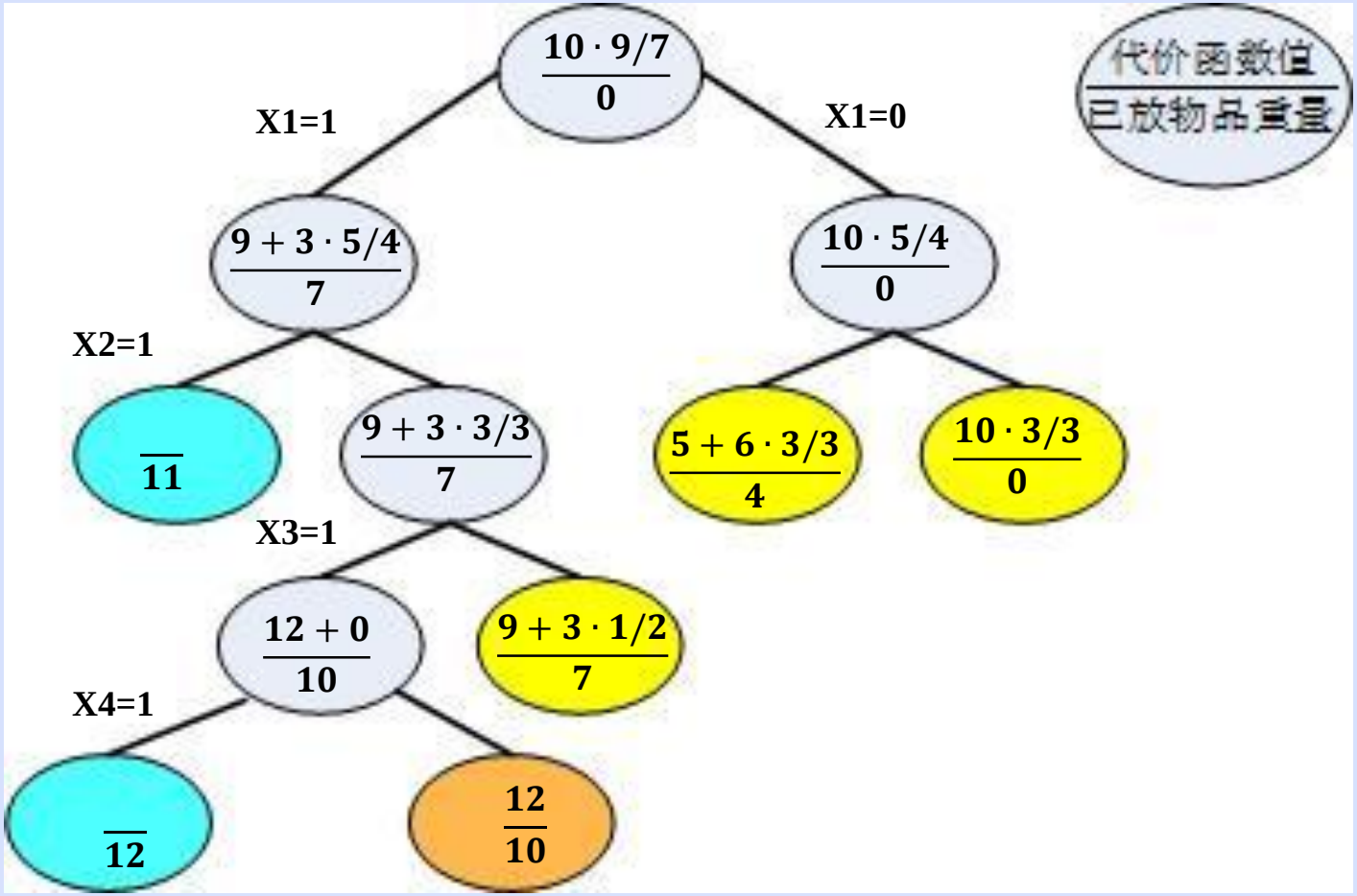
分支策略----深度优先

0-1背包问题： 4种物品，重量*wi*与价值*vi*分别为（重排）

*v*1=9，*v*2=5，*v*3=3，*v*4=1

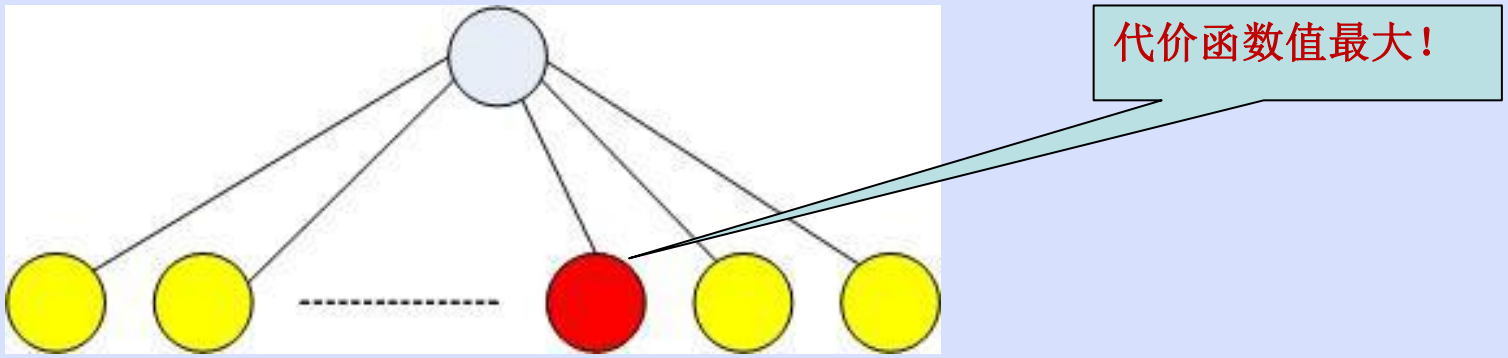
*w*1=7，*w*2=4，*w*3=3，*w*4=2

背包重量限制为10



代价函数告诉了我们什么？

从这点向下搜索所能获得的最大值！



启发：从最可能获得最大值的分支进行搜索！

广度优先，代价函数值优先

- 分支限界法中搜索树空间扩展

（1）队列式(FIFO)分支限界法

按照队列先进先出（FIFO）原则选取下一个结点为扩展结点

（2）优先队列式(minHeap / maxHeap)分支限界法

按照优先队列中规定的优先级选取优先级最高的结点成为当前扩展结点

0-1背包问题

- 基本思想

- 搜索空间：子集树，一个物品要么装入，要么不装入
- 分支限界的搜索空间扩展：优先队列——最大堆
- 预处理：将输入按照单位重量价值的顺序排序

代价函数　对于结点*i*（第i层的结点）

up=已装入物品的价值+剩余容量装满获得的最大价值

- 剪枝策略

<1> 左子树：装入w[i]，若ew+w[i]<c，则可行

若cp+p[i]>bestp, 则bestp=cp+p[i]

下一层活结点优先级：heap.addNode(up,cp+p[i],cw+w[i],i+1)

<2> 右子树：不装入w[i], up=bound(i+1)

若up>bestp，则可行

下一层活结点优先级：heap.addNode(up,cp,cw,i+1) -

- 算法主要步骤：

```
node=heap.removeMax()  \\扩展节点
 cw=node.weight
 cp=node.profit
 p=node.up
 i=node.level

if(cw+w[i]<=c){        \\左子树
 if(cp+p[i]>bestp) bestp=cp+p[i]
  heap.addNode(up,cp+p[i],cw+w[i],i+1)
}

up=bound(i+1)          \\右子树
if(up>bestp)
 heap.addNode(up,cp,cw,i+1)
```

如何实现bound的计算？

- 上界bound的计算

- 预处理：将输入按照单位重量价值的顺序排序

- 计算bound：

```
cleft=c-cw ; b=cp
while (i<=n && w[i]<=cleft){
  cleft=cleft-w[i];  b=b+p[i]; i++;
}
if(i<=n) b=b+p[i]/w[i]*cleft;
return b;
```

❖分支限界法 vs. 回溯法

- 分支限界法与回溯法的不同

(1) 求解目标：

- 分支限界法: 适于求解满足约束条件的最优解
- 回溯法：找出解空间树中满足约束条件的解（一个或多个）

(2) 搜索方式：

- 分支限界法:一般为广度优先、或代价函数优先
- 回溯法：深度优先

- 分支限界法的结点生成

- 选择一个活结点为扩展结点
- 生成扩展结点的所有儿子结点
- 可行（可能）的儿子结点加入活结点队列

- 分支限界法解决优化问题的基本思路
  - 确定解空间树的结构
  - 确定代价函数，作为结点扩展的依据
  - 确定优先队列和优先级：最大堆/最小堆 （代价函数最优）
  - 最优代价函数优先+剪枝函数
- 常用剪枝函数
  - 用约束函数在扩展结点处剪去不满足约束的子树（问题本身的约束）
  - 用限界函数剪去得不到最优解的子树

    <1> 上界/下界 限界函数

    <2>将可能导致最优解的活结点加入优先队列中

提纲

- 分支限界法的基本思想
- 最大团问题
- 旅行售货员问题
- 总结

最大团问题

例 最大团问题：给定无向图$G=<V, E>$, 求G中的最大团.

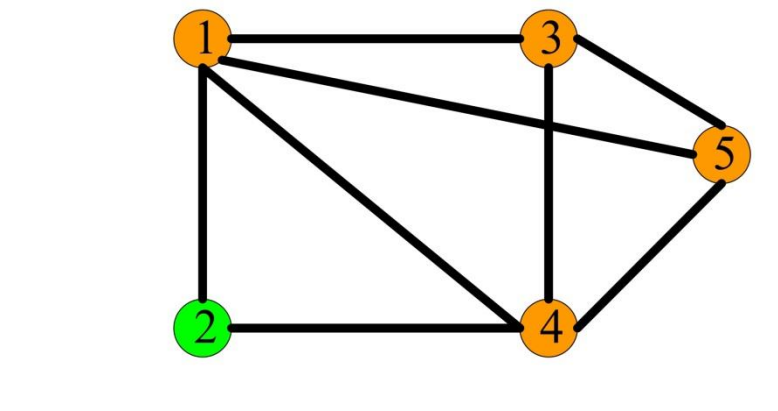
G的子图：$G'= <V',E'>$, 其中$V' \subseteq V$, $E' \subseteq E$,

G的补图：$\overline{G} =<V, E'>$, E'是E关于完全图边集的补集

G中的团：G 的完全子图

G中的最大团：顶点数最多的团

实例
最大团：{ 1, 3, 4, 5 }



独立集与团

G 的点独立集：G 的顶点子集A，且$\forall u,v\in A, \{u,v\}\notin E$.

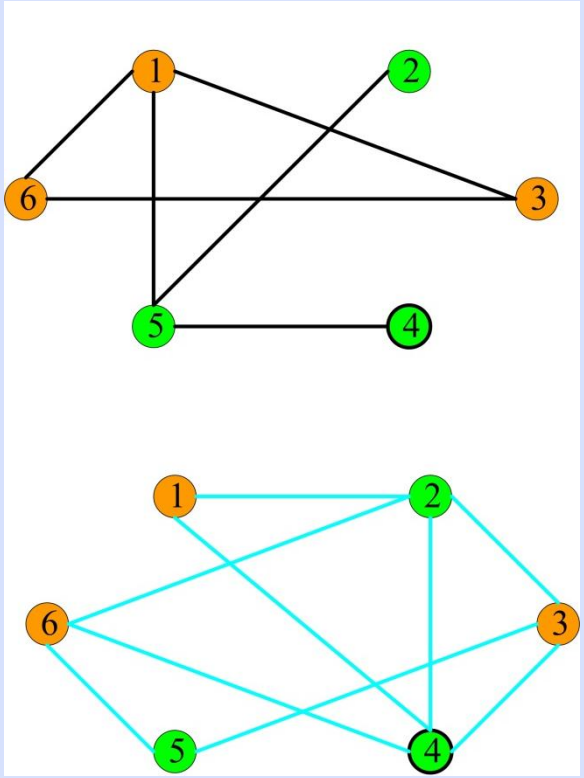
最大点独立集：顶点数最多的点独立集

命题：U是G 的最大团，当且仅当U是$\overline{G}$的最大点独立集

G 的最大团：

U={ 1, 3, 6 }

补图$\overline{G}$的最大点独立集



分支限界算法设计

搜索树为子集树，搜索策略为深度优先

结点$<x_1, x_2, \cdots, x_k>$的含义：

已检索 k 个顶点，其中 $x_i=1$ 对应的顶点在当前的团内

约束条件：该顶点与当前团内每个顶点都有边相连

界：当前图中已检索到的极大团的顶点数

代价函数：目前的团扩张为极大团的顶点数上界

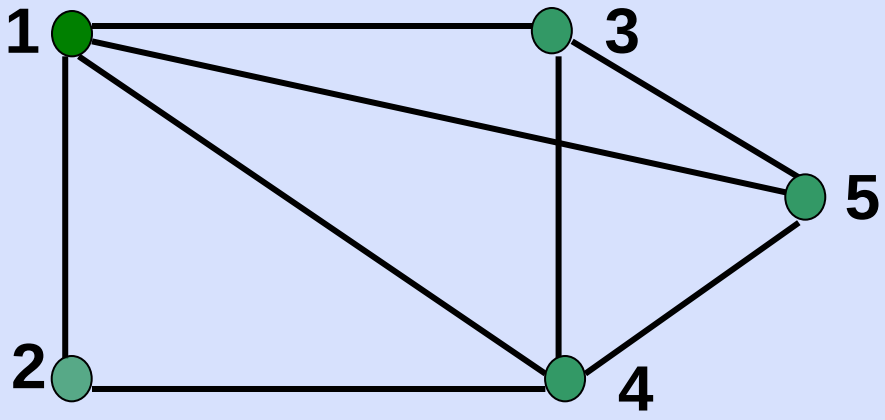
$$F = C_n + n - k$$

其中 C_n 为目前团的顶点数（初始为0），

k 为结点层数

最坏情况下时间：$O(n2^n)$

最大团的实例



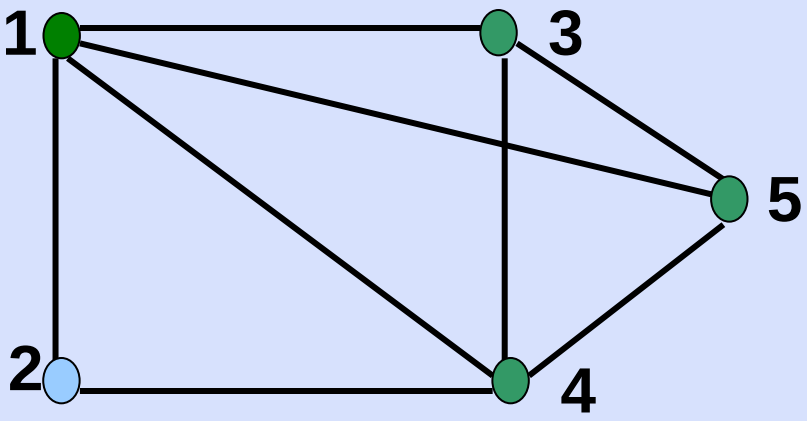
顶点编号顺序为 1, 2, 3, 4, 5,

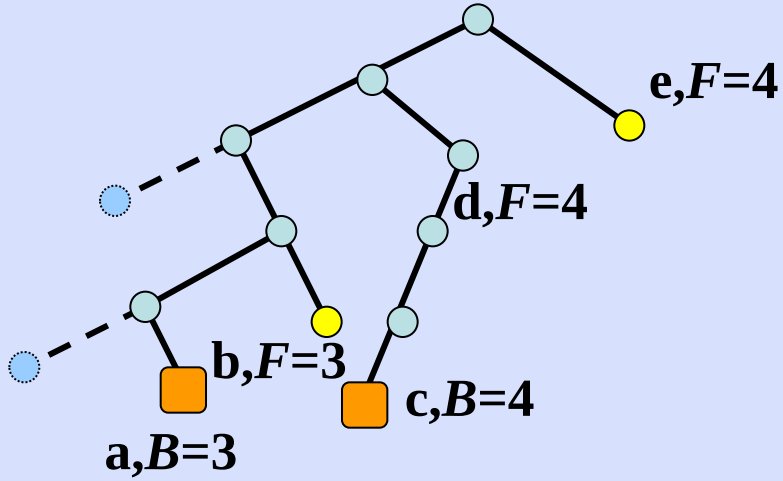
对应 x_1, x_2, x_3, x_4, x_5, x_i=1 当且仅当 i 在团内

分支规定左子树为1，右子树为0.

B 为界，F 为代价函数值.

实例求解





a：得第一个极大团 { 1, 2, 4 }, 顶点数为3, 界为3;
b：代价函数值 F = 3, 回溯;
c：得第二个极大团{1, 3, 4, 5}, 顶点数为4, 修改界为4;
d：不必搜索其它分支, 因为F = 4, 不超过界;
e：F = 4, 不必搜索.
最大团为 {1, 3, 4, 5}, 顶点数为 4.

提纲

- 分支限界法的基本思想
- 最大团问题
- 旅行售货员问题
- 总结

旅行售货员问题

例 给定n个城市集合$C=\{c_1,c_2,\ldots,c_n\}$, 从一个城市到另一个城市的距离 d_{ij} 为正整数，求一条最短且每个城市恰好经过一次的巡回路线.

问题的类型：有向图、无向图.

设巡回路线从1开始，

解向量为$<i_1, i_2, \ldots, i_{n-1}>$,

其中 $i_1, i_2, \ldots, i_{n-1}$为$\{ 2, 3, \ldots, n \}$的排列.

搜索空间为排列树，结点$<i_1, i_2, \ldots, i_k>$ 表示得到 k 步路线

旅行售货员问题

约束条件：令$B = \{ i_1, i_2, \dots, i_k \}$, 则

$$i_{k+1} \in \{ 2, \dots, n \} - B$$

界：当前得到的最短巡回路线长度

代价函数：设顶点 c_i 出发的最短出边长度为 ***minout*[i]**，则

b=cc+rcost

$$rcost = \sum_{i \notin B} minout[i]$$

cc为已走过路径长度

rcost为剩余长度下界

旅行售货员问题

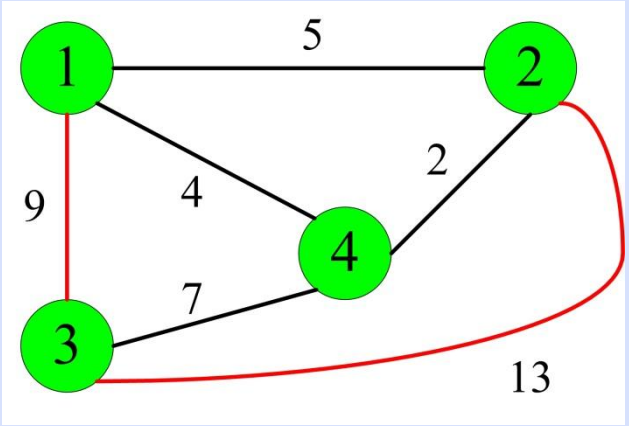
初始化，计算每个顶点的最短出边

$minout[1] = 5$　　$cc = 0$

$minout[2] = 2$　　$rcost = 16$

$minout[3] = 7$

$minout[4] = 2$　　$b = 16$



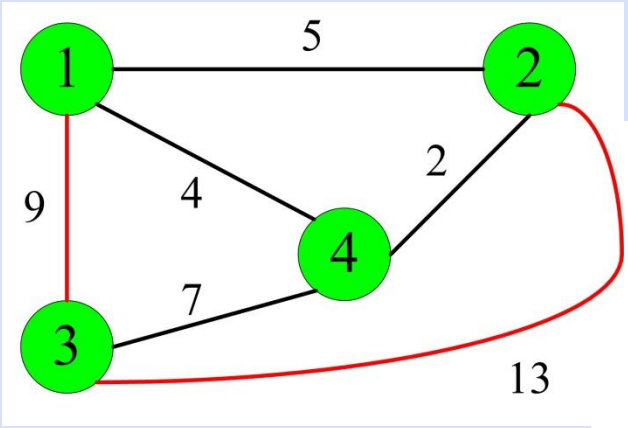
假设部分路线为<1, 3, 2>

$b = 9 + 13 + 2 + 2 = 26$

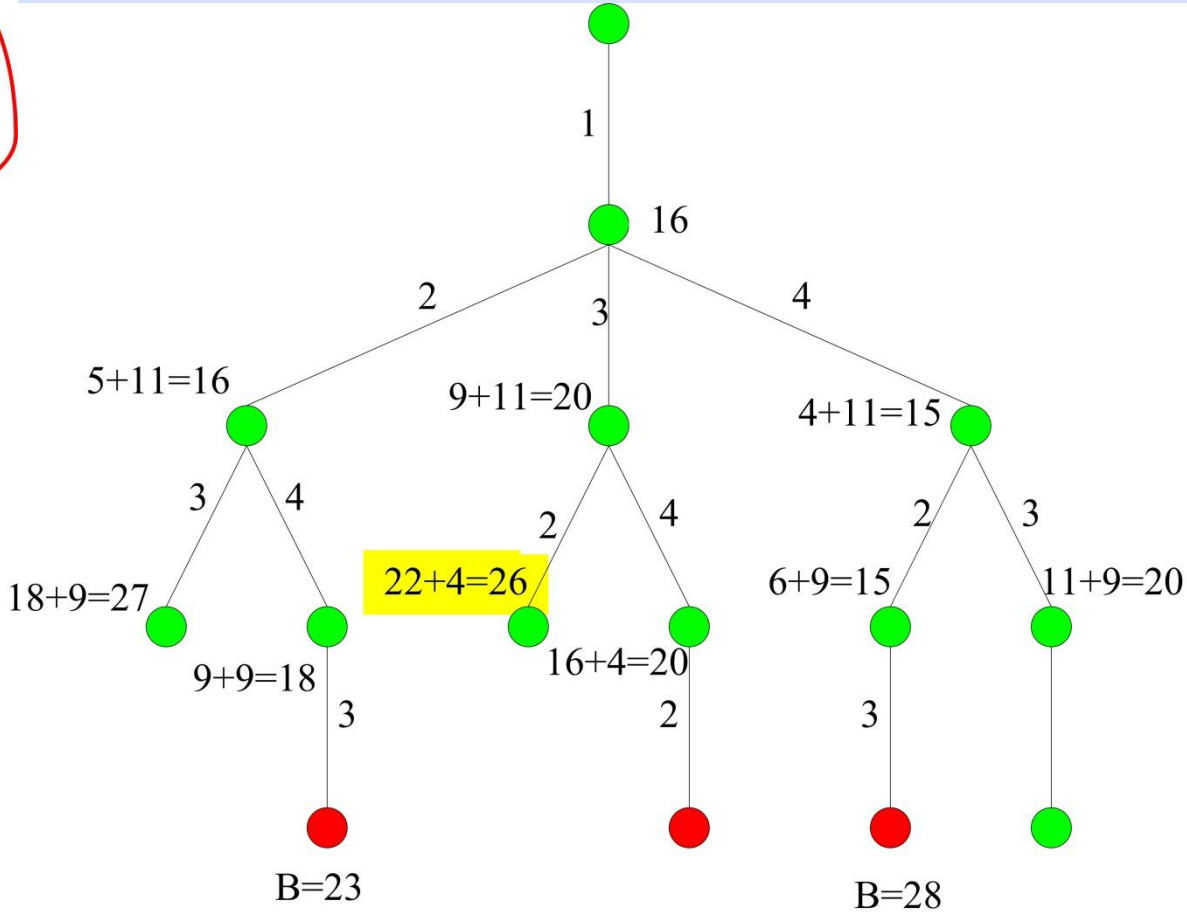
9+13为走过的路径长度

后两项分别为从结点2及4出发的最短边长

旅行售货员问题



$$minout[1] = 5$$
$$minout[2] = 2$$
$$minout[3] = 7$$
$$minout[4] = 2$$



旅行售货员问题

❖ 基本思想

- 搜索空间：排列树，

- 搜索空间扩展：最小堆优先队列

- 优先级：

子树下界b ($b=cc+rcost$)

- 剪枝策略：

若当前扩展结点的儿子结点i有b>bestc，则结点i不可能导致最优解

仅当当前扩展结点的儿子结点i有b<bestc，则结点i被加入活结点优先队列

- **分支限界算法主要步骤**

- 取当前扩展结点：heap.removeMin()

- 对于当前扩展结点*enode*（*s*层）：

产生*s* 的儿子结点*i*：

<1> i: s+1 ，且a[s,i]∈E

<2> cc=enode.cc+a[s,i],

rcost=enode.rcost-minOut[s]

b=cc+rcost

<3> if(b<bestc) heap.addNode(node i)

MinHeapNode

.lcost

.cc

.rcost

.s //根结点到当前结点的路径为x[0:s]

.x //数组，需要进一步搜索的顶点是x[s+1:n-1]

旅行售货员问题的分支限界算法

约束条件：只能选没有走过的结点

代价函数：走过长度+后续长度的下界

时间复杂度: $O(n!)$

连续邮资问题

例 给定n种不同面值的邮票，每个信封至多m张， 试给出邮票的最佳设计，使得从1开始，增量为1的连续邮资区间达到最大？

实例：n=5，m=4，

设计1

面值 X_1=<1,3,11,15,32>， 邮资连续区间为{ 1, 2, …,70（32+32+3+3） }

设计2

面值 X_2=<1,6,10,20,30>， 邮资连续区间为{1, 2, 3, 4}

连续邮资问题

可行解：$<x_1, x_2, \ldots, x_n>$，$x_1=1$，$x_1< x_2< \ldots < x_n$

搜索策略：深度优先

约束条件：在结点$<x_1, x_2, \ldots, x_r>$处，邮资最大连续区间为$\{1, \ldots, r_i\}$，x_{i+1}的取值范围是$\{x_i+1, \ldots, r_i+1\}$

若$x_{i+1}>r_i+1$, r_i+1的邮资将没法支付.

例：$n=5$，$m=4$,$X=<1,3,11>$，邮资连续区间为$\{1, 2, \ldots, 18\}$

x_4的取值范围是$\{12, \ldots, 19\}$

若x_4取20，那么19的邮资无法支付

确定x_i后r_i的计算

$y_i(j)$：用至多m张面值x_i和$x_1,x_2,\ldots,x_{i-1}$面值的邮票，贴出j邮资时的最少邮票数，设x_i的张数为$0\le t\le m$，则

$$y_i(j) = \min_{0\le t\le m}\{t + y_{i-1}(j - tx_i)\}$$

$$y_1(j) = j$$

$$r_i = \min\{j \mid y_i(j) \le m, y_i(j+1) > m\}$$

例： $n=5$，$m=4$,$X=<1,3,11>$，邮资连续区间为{ 1, 2, …,18 }

x_4的取值范围是{ 12, … , 19}

$x_4=12$，

$y_i(19) = min\{1+3\}$　　$y_i(20) = min\{0+4, 1+4\}$

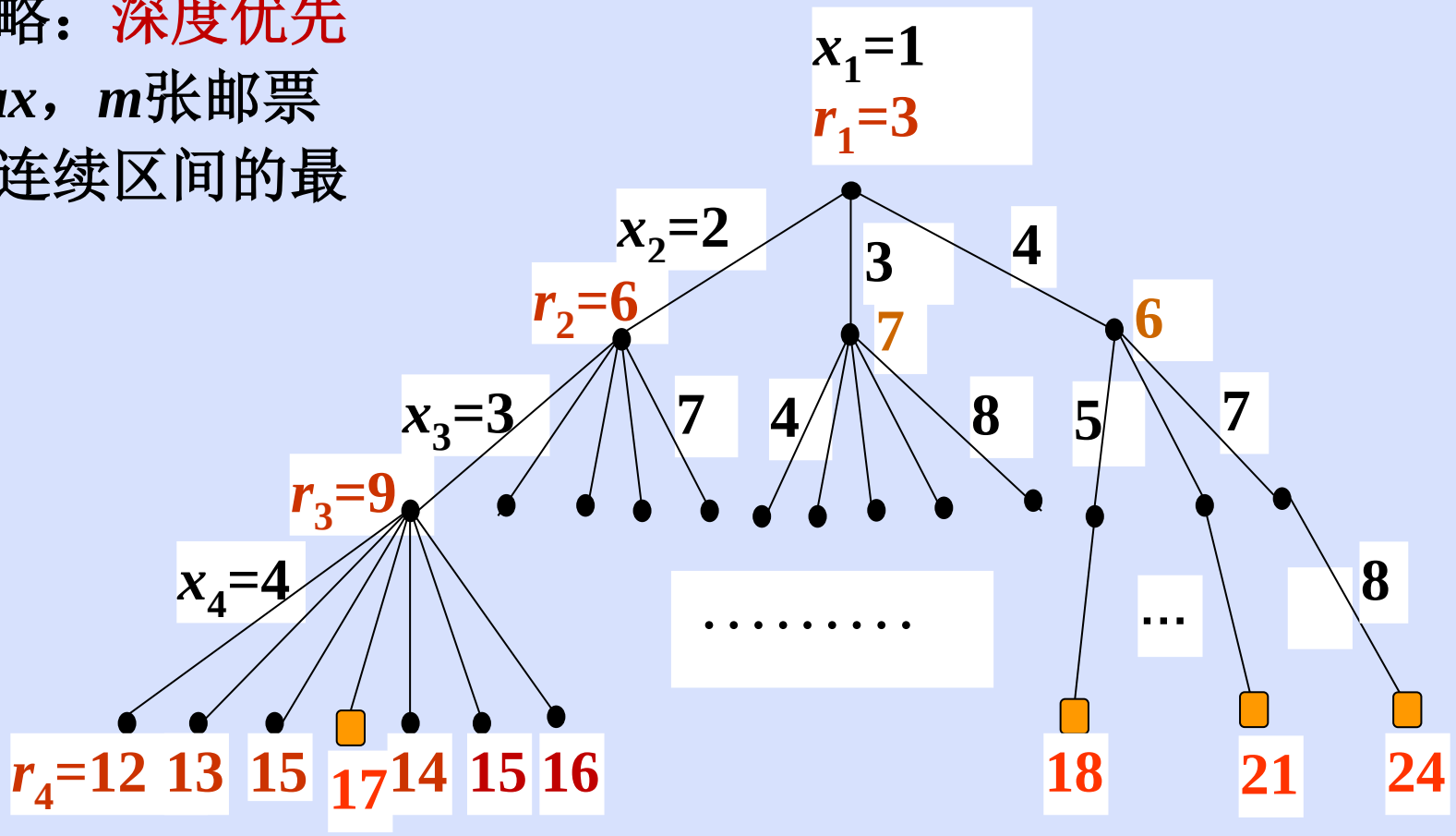
……

$y_i(30) = min\{0+6, 1+4, 2+2\}$　　$y_i(31) = min\{0+5, 1+5, 2+3\}$

实例：$n=4,m=3$

搜索策略：深度优先

界：max，m张邮票可付的连续区间的最大邮资



解：X=<1,4,7,8>, 最大连续区间为{1,2,…,24}

总　结

分治策略

适用条件：能分解为独立求解的子问题

设计步骤：递归分解, 初始子问题的计算, 子问题解的综合方法. 注意子问题划分均衡，类型相同

递归算法分析：求解递推方程

改进途径：减少子问题数

典型问题: 二分检索,归并排序,快速排序，大整数乘法，Strassen矩阵乘法

动态规划

适用条件：优化问题, 多步判断求解, 满足优化原则, 子问题重叠

设计步骤：确定子问题, 列原问题和子问题的递归方程（最优子结构）；自底向上,备忘录存储；解的追踪方法

复杂度分析：备忘录, 递推方程

典型问题: 矩阵链相乘,背包,最长公共子序列,图像压缩,最大子段和,最优二分检索树

贪心法

适用条件：组合优化问题, 多步判断求解, 有贪心选择性质

设计步骤：局部优化策略的确定及算法正确性证明(直接证明, 数学归纳法)

复杂度分析：比较简单

典型问题: 活动选择,装载问题, 最优前缀码,最小生成树,单源最短路

回溯和分支限界

适用条件：搜索或优化问题, 多步判断求解, 满足多米诺性质

设计步骤：确定解向量, 搜索树结构,搜索顺序,结点分支搜索的约束条件与代价函数,路径存储

复杂度分析：最坏情况和蛮力算法没有区别，平均情况要好，空间需求较少

典型问题: *n*后问题, 背包问题,旅行售货员问题,装载问题, 最大团问题, 连续邮资问题


谢 谢 大 家！