第六章

分支限界法
Branch and bound algorithm

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

（1）队列式(FIFO)分支限界法

（2）优先队列式分支限界法

通过算法实例学习分支限界法的设计策略

6.1 基本思想 分支限界法与回溯法

类似于回溯法，也是在问题的解空间树上搜索问题解的算法。

深度优先生成状态空间树中的结点,并使用剪枝函数的求解方法称为回溯法; 广度优先生成状态空间树中的结点，并使用剪枝函数的求解方法称为分枝限界法.

- 求解目标：回溯法的求解目标是找出解空间树中满足约束条件的所有解或最优解;分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出在某种意义下的最优解。
- 搜索方式：回溯法以深度优先的方式搜索解空间树;分支限界法则以广度优先或以最小耗费优先的方式搜索解空间树。

6.1 基本思想　　搜索策略

分支限界法的搜索策略：在扩展结点处，先生成其所有的儿子结点（分支），从当前的活结点表中选择下一个扩展结点。

在分支限界法中，每一个活结点只有一次机会成为扩展结点。活结点一旦成为扩展结点，就一次性产生其所有儿子结点。在这些儿子结点中，导致不可行解或导致非最优解的儿子结点被舍弃，其余儿子结点被加入活结点表中。

为有效选择下一扩展结点，加速搜索的进程，在每一活结点处，计算限界函数U的值，并根据这些已计算出的函数值，从当前活结点表中选择一个最有利的结点（计算耗费最小）作为扩展结点，使搜索朝着解空间树上有最优解的分支推进，尽快找出一个最优解。

6.1 基本思想 求解过程

分支限界法求解过程

1.确定解空间结构

2.确定约束条件和目标函数

3.取限界函数U=U(T).

4.扩展根结点的所有儿子.对每一子结点x判定其是否满足约束条件，对满足约束条件的 x 计算最小耗费下界$\overline{c(x)}$, 将 $\overline{c(x)}$ ≤U 的x 加入活动结点表.

5. x为叶结点时,检查是否c(x)<U, 是则用c(x)更新U.

6. 取活动结点表中的第一个结点为根, 重复4.

6.1 常见的两种分支限界法

（1）队列式(FIFO)分支限界法

将活动结点表组织成一个队列，按照先进先出（FIFO）原则选取下一个结点为扩展结点。

（2）优先队列式分支限界法

将活动结点表组织成一个优先队列，按照规定的优先级选取优先级最高的结点成为当前扩展结点。

区别：从活结店表中取下一节点为扩展节点的方式不同，队列式按照队列先进先出的原则选取下一个子节点为扩展节点；优先队列式按照优先队列中的优先级进行选取子节点成为扩展节点。

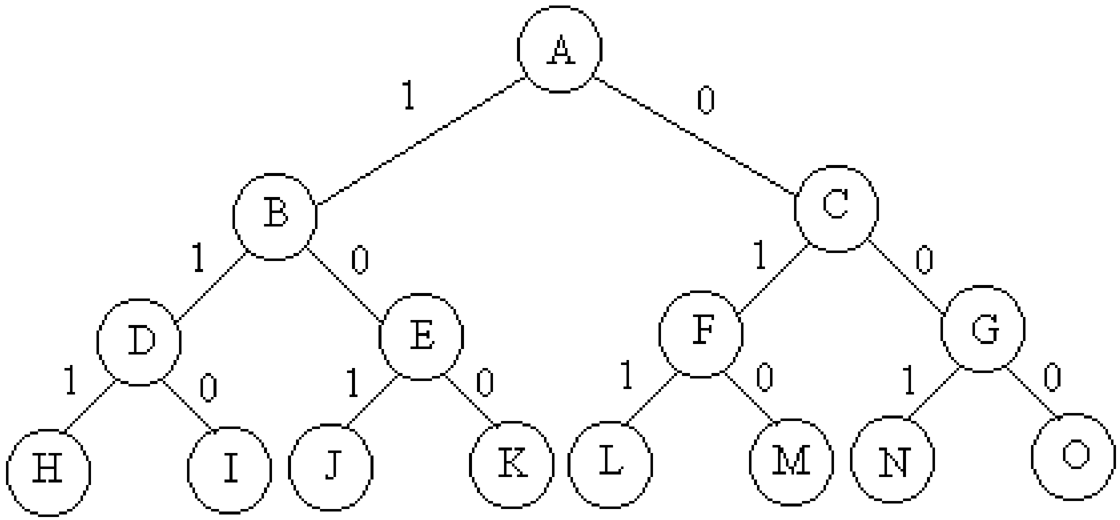
6.2 分支限界法求0-1背包问题

问题陈述 设有n个物体和一个背包,物体i的重量为w_i价值为p_i,背包的载荷为M, 若将物体i(1≤ i ≤n,)装入背包,则有价值为p_i . 目标是找到一个方案, 使得能放入背包的物体总价值最高.

- 设N=3, W=(16,15,15), P=(45,25,25), C=30

1.队列式分支限界法

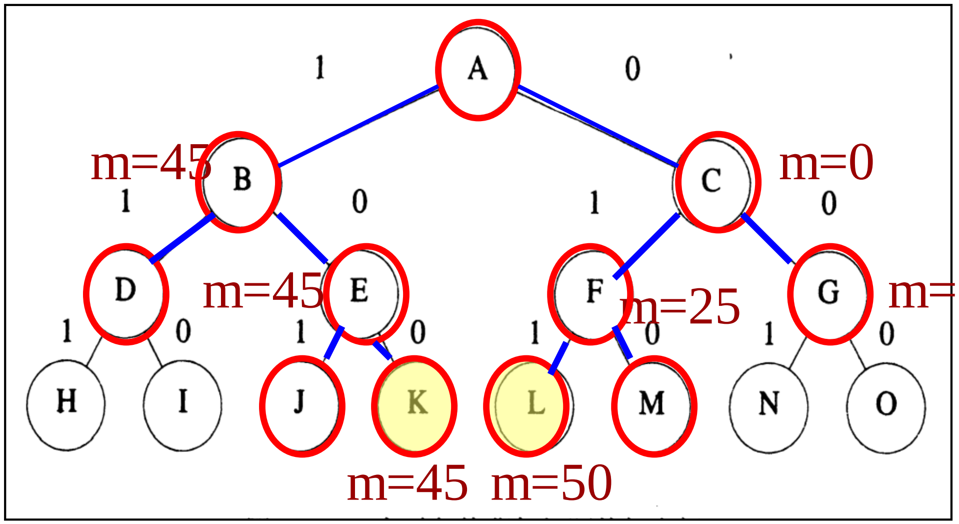
2.优先队列式分支限界法



6.2 分支限界法求0-1背包问题

队列式分支限界法

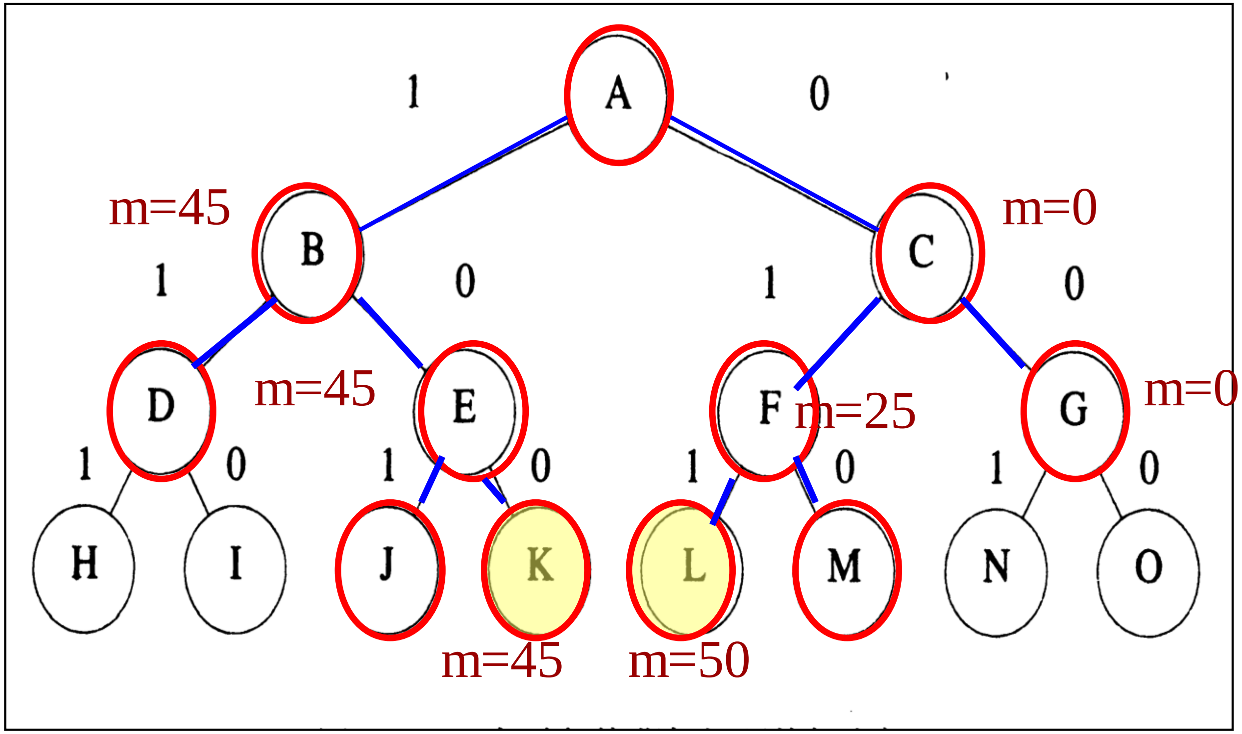
- 用一个队列存储活结点表，初始为空
- A为当前扩展结点，其儿子结点B和C均为可行结点，将其**按从左到右顺序加入活结点队列**，并**舍弃**A。
- 按FIFO原则，下一扩展结点为B，其儿子结点D不可行，舍弃；E可行，加入。舍弃B
- C为当前扩展结点，儿子结点F、G均为可行结点，加入活结点表，舍弃C
- 扩展结点E的儿子结点J不可行而舍弃；K为可行的叶结点，是问题的一个可行解，价值为45



6.2 分支限界法求0-1背包问题

队列式分支限界法

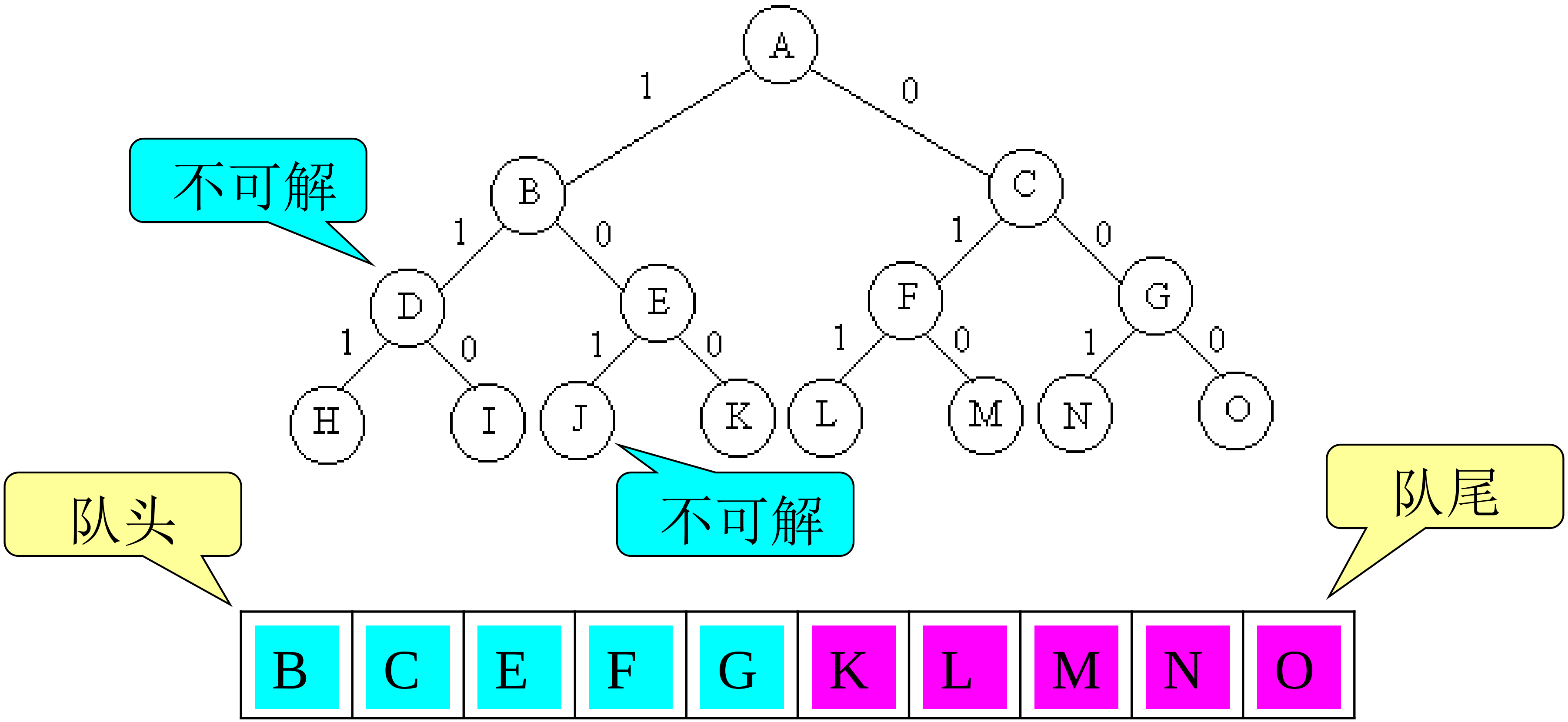
- 当前活结点队列的队首为F, 儿子结点L、M为可行叶结点，价值为50、25
- G为最后一个扩展结点，儿子结点N、O均为可行叶结点，其价值为25和0
- 活结点队列为空，算法结束，其最优值为50



6.2 分支限界法求0-1背包问题

队列式分支限界法

详细求解过程图示：用队列式分支限界法解此问题时，用一个队列来存储活结点表。

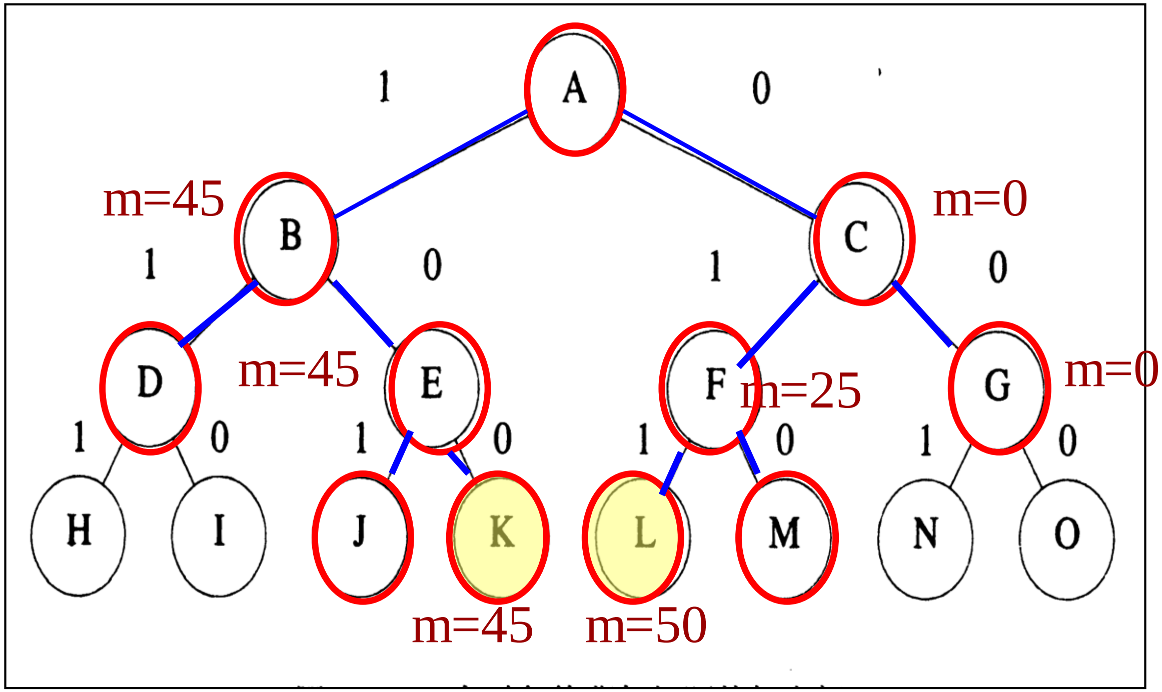


起始结点为A

活结点队列

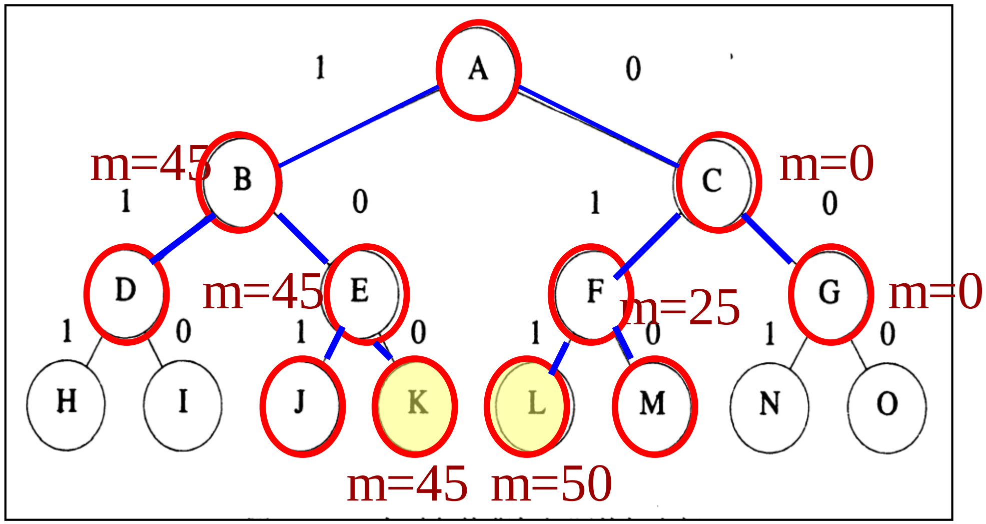
6.2 分支限界法求0-1背包问题 优先队列式分支限界法

- 用一个极大堆表示活结点表的优先队列，其优先级定义为活结点所获得的价值。初始为空。
- 由A开始搜索解空间树，其儿子结点B、C为可行结点，加入堆中，舍弃A。
- B获得价值45，C为0. B为堆中价值最大元素，并成为下一扩展结点。



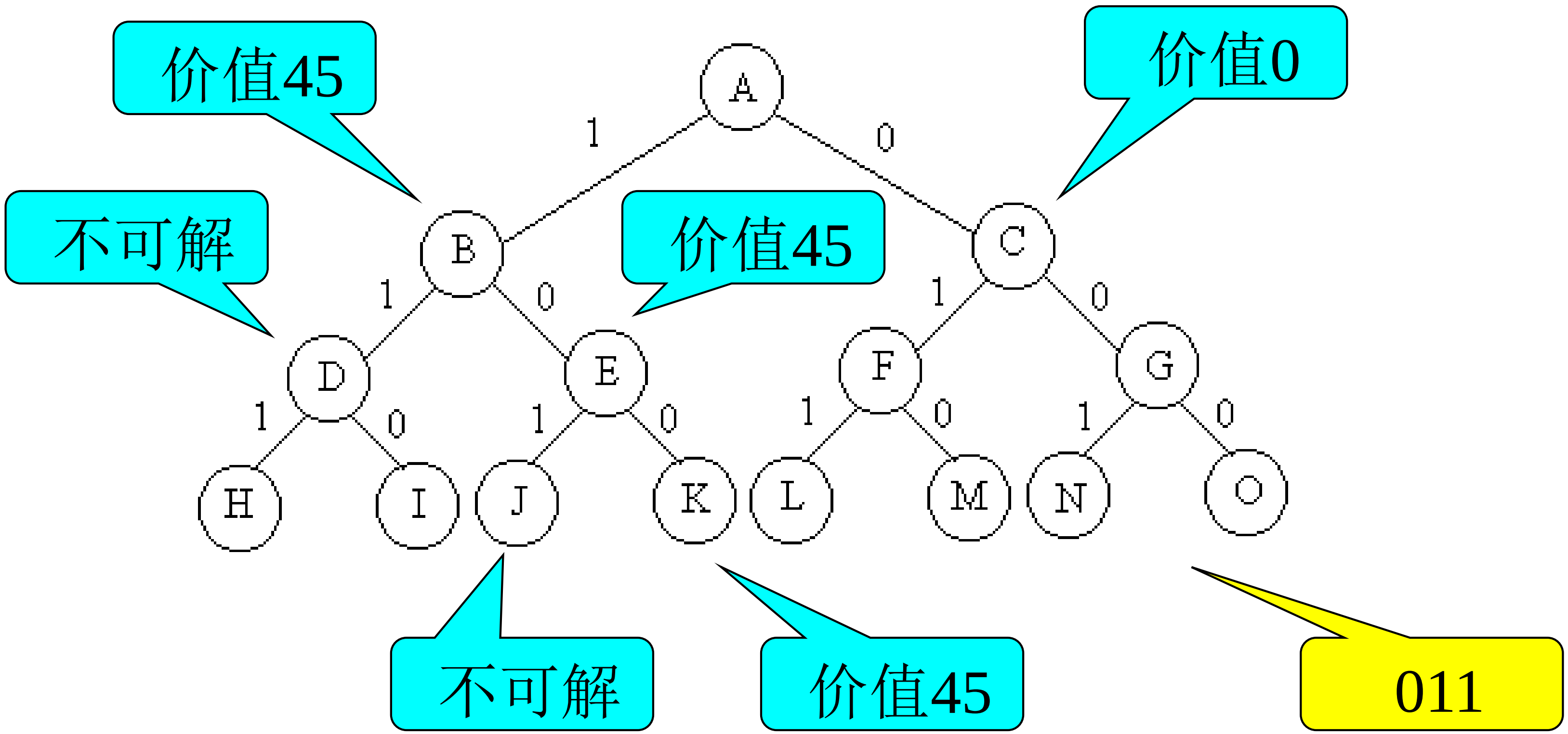
6.2 分支限界法求0-1背包问题　优先队列式分支限界法

- B的儿子结点D是不可行结点，舍弃。E是可行结点，加入到堆中。舍弃B。
- E的价值为40，是堆中最大元素，为当前扩展结点。
- E的儿子J是不可行叶结点，舍弃。K是可行叶结点，为问题的一个可行解价值为45.
- 继续扩展堆中唯一活结点C，直至存储活结点的堆为空，算法结束。
- 算法搜索得到最优值为50，最优解为从根结点A到叶结点L的路径（0，1，1）。



6.2 分支限界法求0-1背包问题　优先队列式分支限界法

➢求解过程图示：



初始堆为空，扩展结点A得到它的2个儿子结点B，C。

6.2 分支限界法求0-1背包问题　　算法描述-上界函数值

```
// 初始化为目前背包的重量
b=cp;
// n表示物品总数，cleft为剩余空间
while (i <= n && w[i] <= cleft) {
        //w[i]表示i所占空间
        cleft -= w[i];
        //p[i]表示i的价值
        b += p[i];
        i++;
      }
// 装填剩余容量装满背包
if (i <= n) b += p[i] / w[i] * cleft;
// b为上界函数值
return b;
```

6.2 分支限界法求0-1背包问题

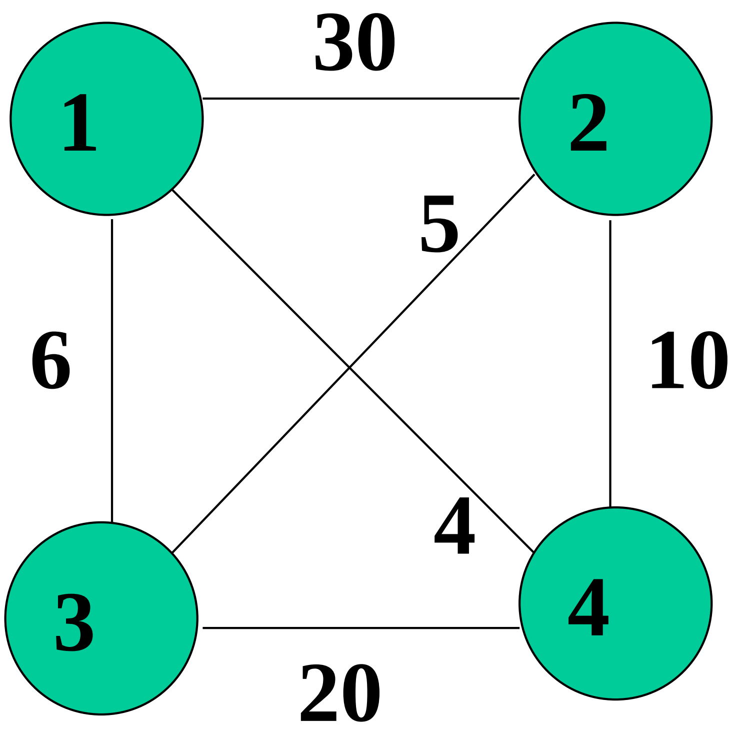
算法核心代码段描述

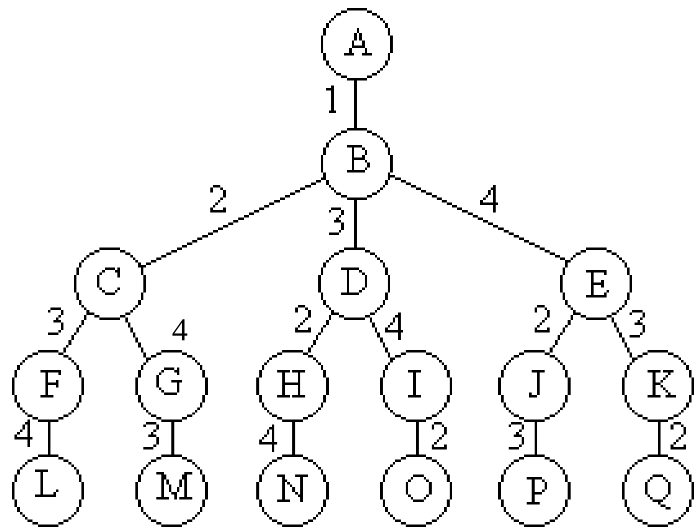
```
while (i != n+1) {// 非叶结点
    // 检查当前扩展结点的左儿子结点
    Typew wt = cw + w[i];
    if (wt <= c) {// 左儿子结点为可行结点
        if (cp+p[i] > bestp) bestp = cp+p[i];
        AddLiveNode(up, cp+p[i], cw+w[i], true, i+1);
    }
    //上界函数值
    up = Bound(i+1);
    // 检查当前扩展结点的右儿子结点
    if (up >= bestp) // 右子树可能含最优解
        AddLiveNode(up, cp, cw, false, i+1);

    // 取下一个扩展节点（略）

}
```

分支限界搜索过程

6.3分支限界法求旅行商问题

旅行售货员问题就是在图中找到一个权最小的周游路线



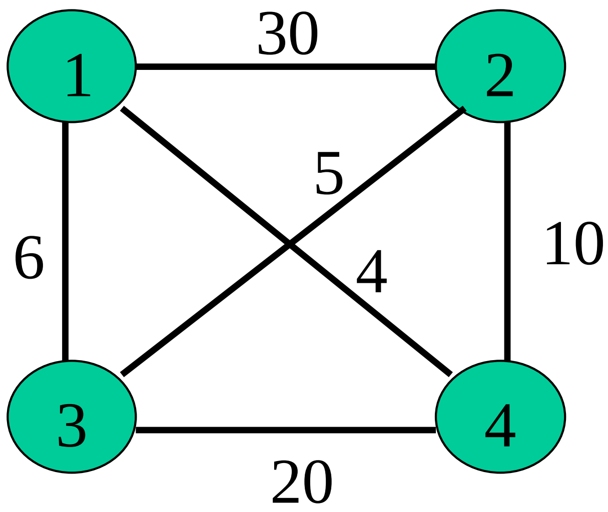


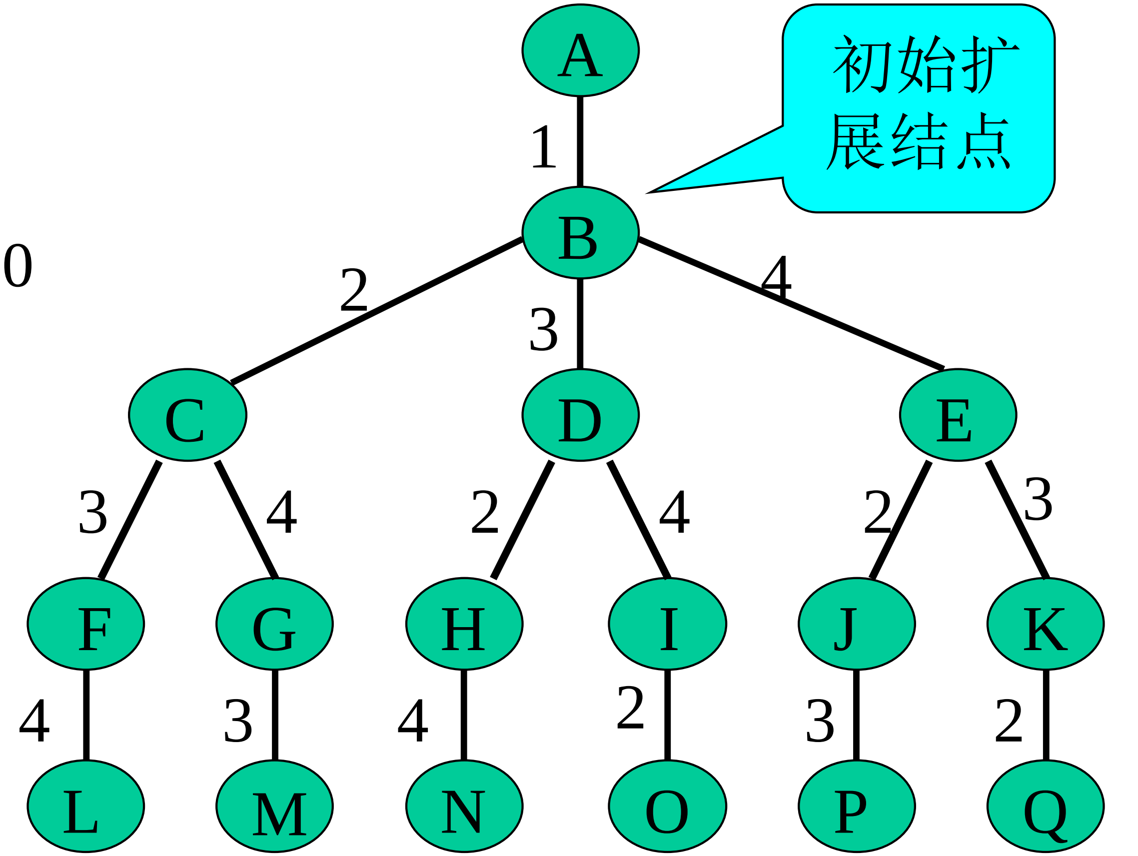
解空间：排列树

回溯法剪枝策略：

当前路径的权重＋下一个路径的权重<当前的最小权重，则搜索该路径

6.3分支限界法求旅行商问题





C	D	E	F	G	H	I	J	K	L

费用为59

6.4装载问题

将第一艘轮船尽可能装满等价于选取全体集装箱的一个子集，使该子集中集装箱重量之和最接近。由此可知，装载问题等价于以下特殊的**0-1**背包问题。

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

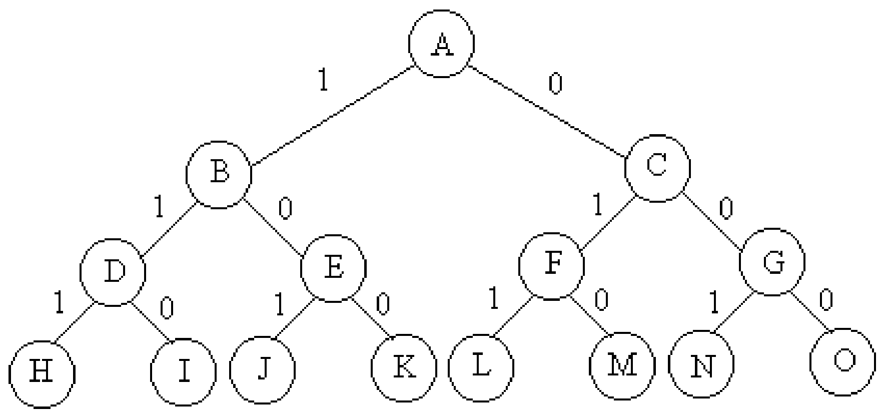
$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \le c_1$$

$$x_i \in \{0,1\}, 1 \le i \le n$$

例如：

$$W = <10,8,5>$$

$$C = 16$$



6.4 装载问题

问题分析：

- 解空间：$X=(x_1,x_2,\ldots,x_n)$，$x_i \in S_i=\{0, 1\}$，$i=1,2,\ldots,n$
- 约束函数：$\sum_{i=1}^{n} w_i x_i \le c_1$
- 目标函数：$\max \sum_{i=1}^{n} w_i x_i$
- 限界函数：$\begin{cases} \text{左孩子：} Ew+w[i] <= c_1 \\ \text{右孩子：} Ew+\sum_{j=i+1}^{n} w_j > bestw \end{cases}$

i+1层结点的剩余重量

6.4 装载问题

Ew: 扩展节点的载重量 W: 重量数组 Q: 活节点队列
bestw: 当前最优载重量 i: 当前处理到的层数 n: 总货物数

```
while (true) {
  // 检查左儿子结点
  Type wt = Ew + w[i]; // 左儿子结点的重量
  if (wt <= c) {   // 可行结点
     if (wt > bestw) bestw = wt;
     // 加入活结点队列
     if (i < n) Q.Add(wt);
  }
  // 检查右儿子结点
  if (Ew + r > bestw && i < n)
      Q.Add(Ew);     // 可能含最优解

  Q.Delete(Ew);    // 取下一扩展结点
  if (Ew == -1) {     // 同层结点尾部
     if (Q.IsEmpty()) return bestw;
     Q.Add(-1);       // 同层结点尾部标志
     Q.Delete(Ew); // 取下一扩展结点
     i++; // 进入下一层
     r-=w[i];//剩余重量}

  }
}
```

在算法的while循环中，首先检测当前扩展结点的左儿子结点是否为可行结点。如果是则将其加入到活结点队列中。然后将其右儿子结点加入到活结点队列中(右儿子结点一定是可行结点)。2个儿子结点都产生后，当前扩展结点被舍弃。

活结点队列中的队首元素被取出作为当前扩展结点，由于队列中每一层结点之后都有一个尾部标记-1，故在取队首元素时，活结点队列一定不空。当取出的元素是-1时，再判断当前队列是否为空。如果队列非空，则将尾部标记-1加入活结点队列，算法开始处理下一层的活结点。

本章小结：

（1）分枝限界法:广度优先生成树中结点并使用剪枝函数的方法.

（2）适合求解的问题:求一个可行解的问题和最优化问题

（3）分枝限界法的分类:

① FIFO分支限界法

② LIFO分支限界法

（4）分枝限界法的基本思想

在分支限界法中，每一个活结点只有一次机会成为扩展结点。活结点一旦成为扩展结点，就一次性产生其所有儿子结点。在这些儿子结点中，导致不可行解或导致非最优解的儿子结点被舍弃，其余儿子结点被加入活结点表中。此后，从活结点表中取下一结点成为当前扩展结点，并重复上述结点扩展过程。这个过程一直持续到找到所需的解或活结点表为空时为止。

（5）求解步骤

1.确定解空间结构

2.确定约束条件和目标函数

3.取限界函数U=U(T).

4.扩展根结点的所有儿子.对每一子结点x判定其是否满足约束条件，对满足约束条件的 x 计算最小耗费下界$\overline{c(x)}$，将 $\overline{c(x)} \le U$ 的x 加入活动结点表.

5. x为叶结点时,检查是否$c(x)<U$, 是则用$c(x)$更新U.

6. 取活动结点表中的第一个结点为根, 重复4.

课后作业

分别用回溯法和分支限界法求解如下0-1背包问题，

问题陈述 设有n个物体和一个背包,物体i的重量为w_i价值为p_i,背包的载荷为M, 若将物体i(1≤ i ≤n,)装入背包,则有价值为p_i .
目标是找到一个方案, 使得能放入背包的物体总价值最高.

- 设N=3, W=(16,15,15), P=(45,25,25), C=30

要求写出求解步骤，搜索过程和算法描述！
并分析两种算法的异同。