## 回溯与分支限界作业

**1. 分派问题：给 n 个人分派 n 件工作，把工作 j 分配给第 i 个人的成本为 COST(i,j)。设计一个回溯算法，在给每个人分派一件不同工作的情况下使得总成本最小。**

解：

给每个人、每件工作均按 1，2，…,n 的顺序编号，从而问题的解可用 n 元组 X[1..n]表示问题的解，X[i]是第 i 个人完成的工作的编号。X 是 n 个数的排列，总共有 n!个元组。

记 totalcost 为目前获得的最小总成本，初始值 totalcost=+∞

curcost 是计算过程中已经发生的成本。

检索：从第一个人开始，按序搜索。对当前正在考虑的第 k 个人，在前面 k-1 个人完成相应工作的基础上，看第 k 个人能做剩下的哪些工作，相应成本是多少，然后继续搜索第 k+1 人…等其他人的工作，构造深度优先搜索过程。

剪枝条件：对当前正在考虑的第 k 个人，若当前考虑完成作业 j，1)j 作业以前没人做，2）若有 curcost+cost(k,j)>totalcost，即当前部分解成本已经大于已知的最好成本，则剪枝，否则深度优先搜索。

```
backtrack_job(k)
    for j← 1 to n do
        if J(j) =0 then //目前还没人做的作业。对之前已经有人做的作业，就不再考虑了
            if curcost +cost(k,j)< totalcost then //如果 curcost+cost(k,j)>
                                                  //totalcost，剪枝，不再继续 k
                                                  //以后的搜索
                curcost = curcost +cost(k,j);
                X(k)=j  //构造部分解
                J(j)=1;  //设置 j 作业被选用状态
                if k=n then  //如果已经获得问题的一个解，判断是否更优
                    if curcost<totalcost then
                        Y = X;  //保存当前最好解
                        totalcost = curcost
                    endif
                else backtrack_job(k+1) //否则，继续搜索，此时 curcost 等做了修正
                X(k)=0  //回到本层，恢复初始状态，以便于搜索其他作业
                J(j)=0
                curcost = curcost - cost(k,j)
            Endif
         endif
      repeat
end
```

```
procedure Job()  //总控程序
    global totalcost = +∞;
    global curcost = 0;
    X←0
    Y←0 //记录最优解
    Backtrack_job(1);//backtrack_job 退出后，Y 就是最优解，totalcost 是相应
                //的代价
    print Y, totalcost
end
```

**2、设 W=(5,7,10,12,15,18,20)和 M=35，使用过程 SUMOFSUB 找出 W 中使得和数等于 M 的全部子集并画出所生成的部分状态空间树。**