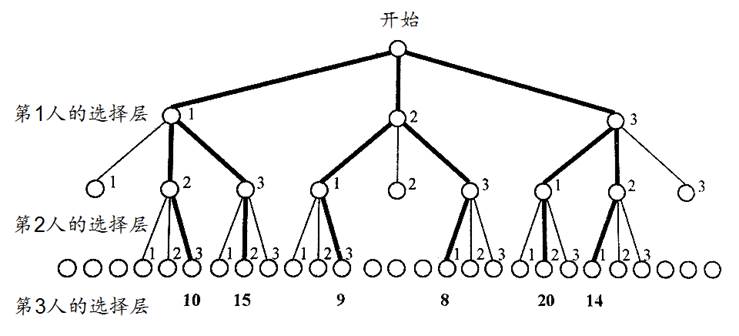
**问题描述**

有n个人和m项课题任务（n≥m），其中第i个人承担第j项课题任务的经费消耗记做COST(i, j)，总体经费消耗情况由n×m的矩阵给定。假设每个人只能承担其中一项课题任务，则可以有不止一种任务分配方案；如果各个人完成不同课题消耗的经费不同，则一定存在一种“最佳”分配方案，使得完成这些课题时消耗的总经费达到最小。

1.首先要设计出一个n+1层的、每个非叶子结点有m个子结点的树的结构，这个结构用多重链表实现，可以参考下图。



伪码:

//以m个人，n项任务为例

typedef struct CTNODE{

double cost; //存放个人经费消耗

int person\_num; //记录是第几人的选择层，值等于当前结点所在层次数-1

int child\_num; //记录是第几个任务，值等于母结点的从左往右的第几个子结点数

char travellist[m] //记录是遍历过哪些结点，遍历的时候会用上

double allcost //记录总的费用，遍历的时候会用上

CTNODE\* child[n]; //子结点

};

2.设计完以后，需要给整棵树的结点初始化： person\_num，child\_num，cost这三个变量需要初始化。

注意，叶子结点的子结点需要初始化为NULL。

伪码:

void inittree(CTNODE& rootnode, COST矩阵) { //需要提供一个根节点的引用，和cost矩阵

rootnode.cost = 0; rootnode.person\_num = 0; rootnode.task\_num = 0; //根结点的数据成员无实际意义，全设为0

…利用cost矩阵给结点的cost赋值

}

完成这两步以后。从树的根结点深度优先往下遍历到叶子结点，于此同时，把遍历过结点的cost加起来存入allcost，遍历到叶子结点的时候叶子结点中allcost的每一个值就是一个分配方案的总经费消耗。

3.遍历算法函数部分，需要提供一个根节点的引用。

需要记录遍历过哪些结点，存入travellist[m]，可以把上一个结点的travellist[m] 插入当前结点的信息（形式为”cost[person\_num，child\_num]”）,来实现。

需要把遍历过结点的cost加起来存入当前结点的allcost，可以把上一个结点的allcost与当前结点的cost相加来简单实现。

需要把所有的叶子结点存入一个新的辅助CTNODE数组（n个人和m项课题任务有n!/(n-m)!种方案，所以数组容量大小动态分配为n!/(n-m)!），可以用判断子结点是否为空来简单实现。这样做可以只遍历一遍树。

void traveltree(CTNODE& rootnode){

}

完成第三步以后，从所有叶子结点中的allcost寻找最小值即可，最小值理论上可以有多个，按照雨虹上的要求需要输出所有的allcost的最小值及对应叶子结点的travellist[m]。

4.排序部分，需提供辅助的CTNODE数组

void sortcost(叶子结点数组的引用 ){

}

5.主函数部分，需要输入cost矩阵并储存，依次调用初始化，遍历算法，排序。