











解：往届同学答案，可用贪心算法改善f初值;亦可以根据X.t[]用贪心算法估计在任务1,2,…，X.length分派后，该路径总执行时间的上界，与f比较来剪枝(程序中用X.Time与之比较，并未实现下述算法思想中的限界函数。当然，其贪心法求k机调度问题的上界比较复杂，可以用更简单的贪心规则：按执行时间降序排列，依次分配给机器1,2,..k,k,k-1…1,1,2…)。

算法思想：

**将n个任务按照所需时间非递减排序，得到任务序列1，2，3，4…….,n，满足时间关系t[1]<t[2]<……<t[n].**

限界函数：

将n个任务中的前k个任务分配给当前k个机器，然后将第k+1个任务分配给最早完成已分配任务的机器，依次进行，最后点所需的时间超过这个已知的最优值，则删掉以此节点为根的子树。否则更新最优值。找出这些机器最终分配任务所需时间最长的，此时间作为分支限界函数。

优先级：哪台机器完成当前任务的时间越早，也就是所有机器中最终停机时间越早， 优先级就越高，即被选作最小堆中的堆顶，作为扩展节点。

设task[n]用来记录最优的调度顺序

每个节点具有信息：

**{Parent**：父亲节点，**Level**：节点所在深度加1，**Ctime**：运行到当前节点所用时间**}**

①当**level<=k**时（不能用界限函数来剪枝，也不需要判断优先级），由于机器还未装满（即前面的k个任务其实是同时加入的），可以令Ctime=0，

②当**level>k**时（需要界限函数来剪枝，需要判断优先级），Ctime就是运行到当前状态所用的总时间，Ctime作为优先级函数，即从最小堆中选取Ctime最小的节点优先。

当找出第一个解节点时，记录此时的Ctime作为目标函数值，以后生成的节点的Ctime大于该目标函数值时，就可以剪掉该节点，如此下去一直到最小堆为空为止。

上述就是最佳调度问题的分支限界算法。

解空间树的节点包括以下信息:

Node{

int Path[n]; //节点对应的解空间树的路径，即到该节点为止的策略记录（分支限界一定要有记录）

int T[k];//在本策略下的每台机器的运行时间（每）

int Time; //本策略的总执行时间，为每台机器运行时间的最大值（max优化的目标）

int length;//本节点的深度,即当前处理的作业（扩展）

}

算法实现：

Proc BestDispatch(int n,int k,int t[]){

Node Boot,X,P,result; //Boot为根节点，result保存最优解

int f;//记录当前最优解的执行时间

f=n\*max(t[]); //初始化

Boot.T[n]={0}; //每台机器的运行时间

Boot.Time=0; // 机器运行时间的最大值

Boot.Path[n]={0}; //解空间树的路径，即到该节点为止的策略记录

Boot.length=0; //初始化根节点（本节点的深度,即当前处理的作业）//f赋初值

AddHeap(Boot);//根节点加入堆中，堆中元素按照Time值由小到大排序

While (!Heap.empty()) do{

P=DeleteMinHeap();//P为当前优先级最高的点

for i=1 to k do//扩展P的k个子节点

X=Newnode(P.Path[],P.T[],P.length+1);

X.Path[X.length]=i;

X.T[i]=X.T[i]+t[X.length];

X.Time=max(X.T[]);

if X.length==n then//X为叶节点

if X.Time<f then//X的执行时间小于已知最优解

f=X.Time;//将X设为最优解

result=X; //当前最优解的执行时间

end{if}

else //X为中间节点

if X.Time<f then

AddHeap(X);

end{if}//X的当前执行时间小于已知最优解则加入堆中，否则剪去

end{if}

end{for}

end{while}

end{ BestDispatch }

二、

1. 子集合问题：设n个不同的正数构成的集合S，求使得和为某数M的S的所有子集。(回溯算法)

解：

设S={a1,a2,..,an}，求S满足条件的所有子集A。用回溯算法，解向量为<x1,x2,…,xn>，xi=0,1，其中xi=1当且仅当ai∈A。部分向量<x1,x2,…,xk>表示已经考虑了对a1,a2,…ak的选择，结点分支的约束条件为B(i)=（且 ak+1+,如果先将S按升序排列的话）。算法描述略。最坏复杂度O(2n)。





a=0：选第一个元素

①b=1时，从b=1直到b=n开始选，看看之后的和有没有合适的（不选的情况，加起来大于了）

②b=2时，从b=2开始选，（即选第0个，不选第一个）

A=1： 不选第一个，选第二个

2. 如图所示，一个4阶Latin方是一个4X4的方格，在它的每个方格内填入1,2,3或4，并使得每个数字在每行、每列都恰好出现一次。用回溯法求出所有第一行为1,2,3,4的所有4阶Latin方。将每个解的第2行到第4行的数字从左到右写成一个序列。如图中的解是<3,4,1,2,4,3,2,1,2,1,4,3>。

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 3 | 4 | 1 | 2 |
| 4 | 3 | 2 | 1 |
| 2 | 1 | 4 | 3 |



1. 分派问题：给n个人分派n件工作，给第i人分派第j件工作的成本是C(i,j) ，试用分枝限界法求成本最小的工作分配方案。算法最坏情况下的复杂度T(n)=O(n.n!)。





