**最佳调度问题的回溯算法实验报告**

1. **实验目的**

通过实践，加深对回溯算法的理解，提高编程能力。

1. **实验内容**

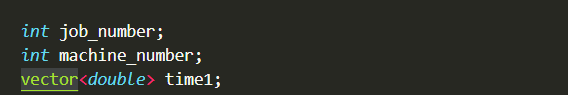
1.从data.txt文件获取数据。

2.设计调度方案，找出完成n个任务的最佳调度，使得完成全部任务的时间最早。这n个任务可由k个可并行工作的机器来完成，完成任务i需要时间为。

1. **实验步骤**

**1.读取数据**

读取数据方法与前几次实验类似，通过getline函数对.txt文件按行读取然后进行字符串分割并进行转化可得到任务数、机器数以及一个存储着n个任务执行时间的一维数组，分别如下所示：



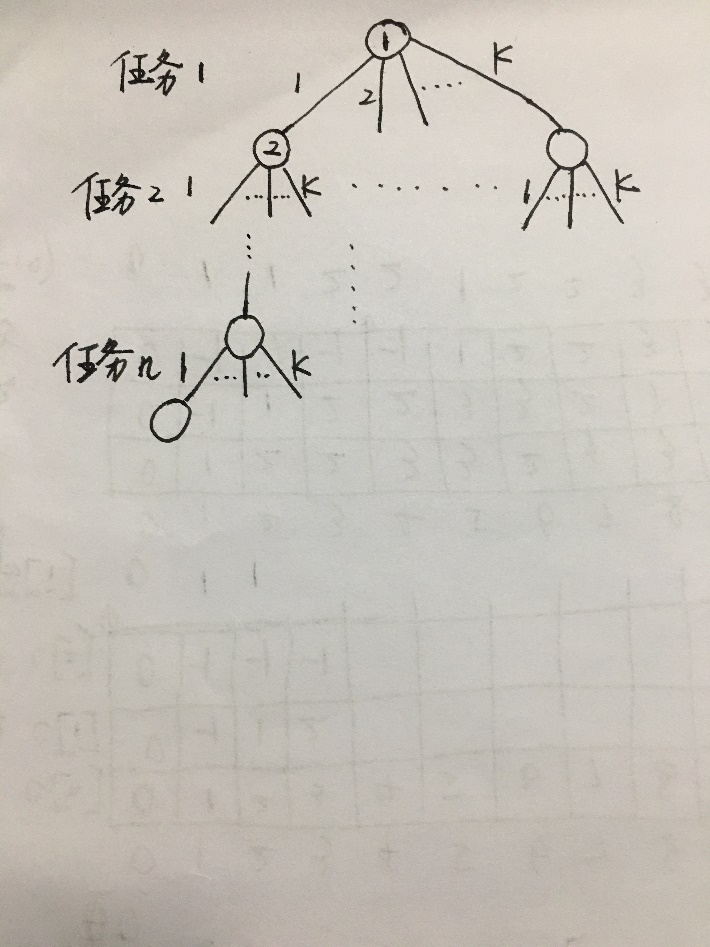
**2.设计回溯算法**

**（1）针对问题描述，定义问题的解空间**

由于是将n个任务放到k个机器上执行，故可定义一个10元组（x1,x2,…,xn）,xi = 1,2,…,k, xi的值j表示第i个任务可以在第j台机器上执行。

**（2）确定易于搜索的解空间组织结构**

解空间树：



**（3）使用深度优先搜索解空间，并使用限界函数进行剪枝**

采用子集树的回溯算法框架编写程序，使用了一个二维数组machine[k][],其中machine[j]是一个一维数组，在搜索解空间过程中，若将第j个任务放到第i台机器上执行，就将j放到machine[i]中。另外，使用如下限界方法进行剪枝：在判断任务j是否可以加到机器i上执行时，首先看当前所有要在机器i上执行的任务执行时间之和加上任务j的执行时间后是否小于等于当前最优调度方案的总执行时间best\_time(因此将best\_time的初始值设为INT\_MAX),若小于best\_time,则任务j可以加到机器i上执行，继续往深层迭代，否则任务j可以加到机器i上。

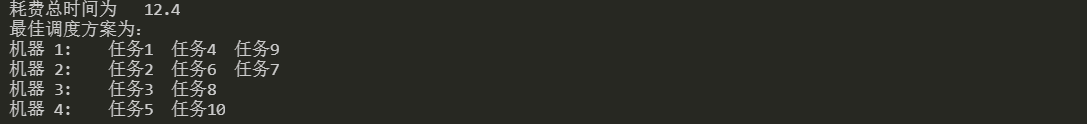
当所有的任务都已经分配到这k台机器上之后，再判断执行总时间最长的那台机器的执行时间是否小于best\_time,如果小于，则可以更新最优解。

上述算法的具体实现如下所示：



1. **实验结果**

对于data.txt文件中的n个任务和k台机器，采用回溯法求得的最佳调度方案如下所示：



1. **实验心得**

使用回溯算法在求解问题时，只要这样的步骤：定义问题的解空间->确定解空间的组织结构->深搜+约束+限界，并结合子集树和排列树的回溯算法框架来设计回溯算法，就可以很快地对问题进行求解。