Experiment—*Optimal Scheduling* problem

PB18010496 杨乐园

Introduction

设计算法实现最佳调度问题：

设有n个任务，由k个可并行工作的机器来完成，完成任务i需要时间为t_i。试设计一个算法找出完成这n个任务的最佳调度，使完成全部任务的时间最早。输出任务所需最短时间，以及给出最佳调度方案。

Purpose

实验目的：熟悉并掌握基于深度优先的回溯法设计算法。

Idea

这是基本的回溯法求解问题，所设计的函数需要用到递归，并层层返回，解空间树是一个n叉树，运用深度优先搜索，每次搜索到叶子结点即更新一次所需的时间，从而在其中选取完成时间最少的一条路径即可。我们只需逐任务进行分配，对当前需要被分配的任务$task_i$，若分配给第j个机器，则更新第j个机器的结束时间，向下继续搜索，直至到叶子节点分配阶数，选取最优解方案即可，再逐层回溯。

Algorithm

根据上述算法设计与思路，直接进行相关代码实现：

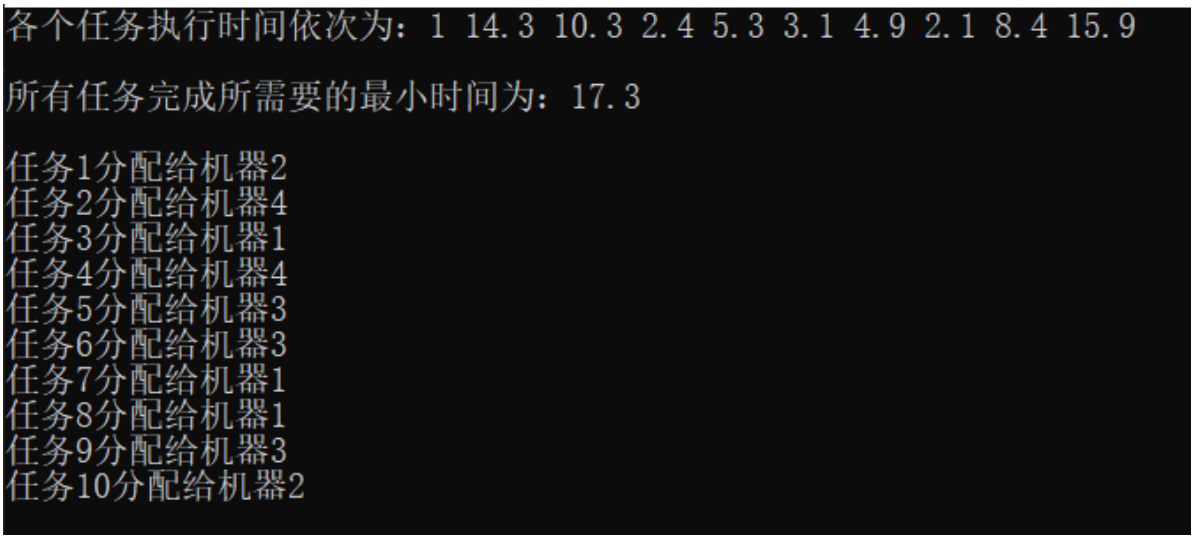
```
#define NUM_TASK 10
#define NUM_MACHSHINE 3

int x[NUM_TASK] = { 0 };                                //记录分配，即x[task]表示任务task分配给机器x[task]
int best_x[NUM_TASK] = { 0 };                           //存储最优分配方案
int min_time = 1024;                                    //执行任务所需最小时间
int time_task[NUM_TASK] = { 1,7,4,0,9,4,8,8,2,4 };//每个任务所需时间
int time_machine_end[NUM_MACHSHINE] = { 0 };        //每个机器运行结束时间

//获取当前已分配任务的完成时间
int getmaxTime(int time_mac[])
{
    int max_time = time_mac[0];
    for (int i = 1; i < NUM_MACHSHINE; i++)
        if (time_mac[i] > max_time)
            max_time = time_mac[i];
    return max_time;
}

//回溯法求解
void BackTrack(int task)
{
```

```
    if (task >= NUM_TASK)
    {
        int current_time = getmaxTime(time_machine_end);//当前已分配任务的完成时间
        if (current_time < min_time)
        {
            min_time = current_time;
            for (int i = 0; i < NUM_TASK; i++)
                best_x[i] = x[i];
        }
    }
    else
    {
        for (int i = 0; i < NUM_MACHSHINE; i++)
        {
            x[task] = i;
            time_machine_end[i] += time_task[task];
            if (time_machine_end[i] < min_time)
                BackTrack(task + 1);
            time_machine_end[i] -= time_task[task];
        }
    }
}
```

从而我们只需从最开始调用函数$BackTrack(0)$即可。

Results

通过运行程序与测试数据，我们有如下输出结果：



我们可以看到，该算法给出了正确的配时方案。

Code

具体完整代码，参看附件文件$Scheduling$。