最佳调度问题的回溯算法

SA20225085 朱志儒

实验内容

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 。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，给出调度方案和完成时间。

data.txt 为输入文件。

程序输入：

3 2

3.1 2.0 1.5

其中，第一行中 3 表示任务个数，2 表示机器个数。第二行中，（3.1, 2.0, 1.5）分别表示 3 个任务的执行时间。

程序输出：

Machine1: <2, 2.0>, <3, 1.5>

Machine2: <1, 3.1>

总时间：3.5

其中，<3, 1.5>中的 3 表示第三个任务，1.5 表示任务执行时间。

实验目的

熟练掌握回溯算法，利用回溯算法解决相关问题。

算法设计思路

伪代码如下：

```
time[n]
minTime = MAXINT
threadTime[k]
map<work, thread>
```

schedule<work, thread>

dfs(1, 0)

```
def dfs(i, curTime):
    if i > n:
        if curTime < minTime:
            minTime = curTime
            schedule = map
        return
    if curTime > minTime:
        return
    for j = 1 to k:
        threadTime[j] += time[i]
        map[i] = j
        curTime = max(threadTime)
        dfs(i + 1, curTime)
        threadTime[j] -= time[i]
```

源码+注释

```
int n, machine;
double time[100], minTime = INT_MAX, machineTime[100];
map<int, int> temp, schedule;

double max(double array[]) {
    double MM = 0;
    for (int i = 1; i <= machine; ++i)
        if (array[i] > MM)
            MM = array[i];
    return MM;
}

void dfs(int i, double curTime) {
    if (i > n) {
        if (curTime < minTime) {
```

```
            minTime = curTime;
            schedule = temp;
        }
        return;
    }
    if (curTime > minTime)
        return;
    for (int j = 1; j <= machine; ++j) {
        machineTime[j] += time[i];
        temp[i] = j;
        curTime = max(machineTime);
        dfs(i + 1, curTime);
        machineTime[j] -= time[i];
    }
}

int main() {
    ifstream infile("data.txt");
    infile >> n >> machine;
    for (int i = 1; i <= n; ++i)
        infile >> time[i];
    dfs(1, 0);
    for (int i = 1; i <= machine; ++i) {
        cout << "Machine" << i << ':';
        for (int j = 1; j <= n; ++j) {
            if (schedule[j] == i) {
                cout << '<' << j << ", " << time[j] << ">, ";
            }
        }
        cout << endl;
    }
    cout << "总时间: " << minTime;
    return 0;
}
```

算法正确性测试

对于输入 data.txt 程序运行的结果如下：

```
Machine1:<2, 2>, <7, 3.5>,
Machine2:<3, 3.2>, <5, 1.4>, <10, 1.3>,
Machine3:<1, 1>, <4, 1.5>, <9, 3.3>,
Machine4:<6, 3.2>, <8, 2.8>,
总时间: 6
```

实验过程中遇到的困难及收获

在本次实验中遇到了一个小小的问题，上次作业我用伪代码表示，访问数组的下标是从 1 开始的，但在实现时下标是从 0 开始的，所以会出一点点问题，其实只要修改实现代码，让它从 1 开始即可。