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)

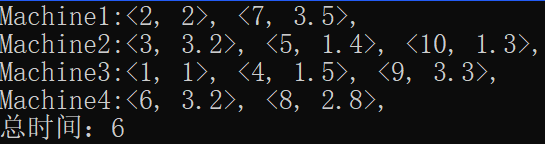
threadTime[j] -= time[i]

## 源码+注释

1. **int** n, machine;
2. **double** time[100], minTime = INT\_MAX, machineTime[100];
3. map<**int**, **int**> temp, schedule;
5. **double** max(**double** array[]) {
6. **double** MM = 0;
7. **for** (**int** i = 1; i <= machine; ++i)
8. **if** (array[i] > MM)
9. MM = array[i];
10. **return** MM;
11. }
13. **void** dfs(**int** i, **double** curTime) {
14. **if** (i > n) {
15. **if** (curTime < minTime) {
16. minTime = curTime;
17. schedule = temp;
18. }
19. **return**;
20. }
21. **if** (curTime > minTime)
22. **return**;
23. **for** (**int** j = 1; j <= machine; ++j) {
24. machineTime[j] += time[i];
25. temp[i] = j;
26. curTime = max(machineTime);
27. dfs(i + 1, curTime);
28. machineTime[j] -= time[i];
29. }
30. }
32. **int** main() {
33. ifstream infile("data.txt");
34. infile >> n >> machine;
35. **for** (**int** i = 1; i <= n; ++i)
36. infile >> time[i];
37. dfs(1, 0);
38. **for** (**int** i = 1; i <= machine; ++i) {
39. cout << "Machine" << i << ':';
40. **for** (**int** j = 1; j <= n; ++j) {
41. **if** (schedule[j] == i) {
42. cout << '<' << j << ", " << time[j] << ">, ";
43. }
44. }
45. cout << endl;
46. }
47. cout << "总时间：" << minTime;
48. **return** 0;
49. }

## 算法正确性测试

对于输入data.txt程序运行的结果如下：



## 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。