**算法分析与设计实验报告**

**第 四 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 6.2 | 地点 | 软件大楼 | | | |
| 实验名称 | 回溯法求解工作分配问题 | | | | | |
| 实验目的 | 通过搜索资料，要求掌握一些基本问题和典型例题的处理方式 | | | | | |
| 实验原理 | 工作分配类似于n皇后问题，类似于棋子的放置，这个题的求解采用排列树框架，，判断某一个工作分配给某人是否合适，如果合适就进行下一层也就是下一个工作的分配，如果不合适就换一个人。 | | | | | |
| 实验步骤 | ①首先输入一个整数代表一共多少个人和多少份工作，并且用二维数组记录每个人对应某一项工作的花费。  ②利用递归的方式判断每一个工作分配给某个人是否合适，我们利用一个循环，循环遍历每个人对当前层也就是当前工作判断这份工作是否合适其中包括花费是否小于当前最佳（限界剪枝），当前人是否已经有工作和是否到达叶子节点，如果满足上述条件便可去判断下一份工作，否则就得换下一位工作人员，同时循环一次结束后要返回工作人员进入该循环前的状态以保证循环进入其他次数的时候不会被受到影响。  ③ 最终我们会遍历完所有的节点并得到一个最小花费，此时所得解就是最优解 | | | | | |
| 关键代码 | void getMinSumCost ( int i , int sum )// i表示当前正在分配的工作，sum表示花费  {  if (sum > min\_sum\_cost) //如果当前花费已经大于已知的最小花费，则剪枝  return;  if (i == n + 1 && sum < min\_sum\_cost)//分配完所有的工作返回结果  {  min\_sum\_cost = sum;  memcpy ( bestArrange , arrange , sizeof ( arrange ) );  return;  }  for (int j = 1; j <= n; j++)//为第1到第n个人分配工作  {  if (!visited[j]) //如果工人j未被分配工作  {  visited[j] = 1;//将其改为已分配状态  arrange[i] = j;//表示工作i分配给工人j  getMinSumCost ( i + 1 , sum+ costs[i][j] );//递归解决第i+1个工作  visited[j] = 0;//回溯  }  }  } | | | | | |
| 测试结果 | 由于是回溯法，时间复杂度为n！结合题目中的数据范围，最终可以给出结果，但超过20或者达到20就已经明显看到程序结果的缓慢  如果有的时候测试不出来请更改一下MAXN或者多运行几次,或者先测试20的规模再测试小规模。 | | | | | |
| 实验心得 | 因为采用的方法是回溯法，需要遍历所有的可能性可能路径，其实也就相当于求解n个数的各种排列，然后分别对应的值求和，我们回溯法自然而然地将所有路径都遍历，并且对路径上的每个点都进行了处理，时间复杂度达到了O(n!)，所以题目也要求了n的数据小于等于20，因此好的算法对实现一个问题的时间开销是不言而喻的，但是当我们如果找不到一个好的算法的时候，回溯法是一个不错的选择。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**