**《算法设计与分析》实验报告**

# 实验五 分支限界

|  |  |  |  |
| --- | --- | --- | --- |
| **报告书** | | | |
| 姓名 | 吴宇敖 | 指导教师 |  |
| 学号 | 176001752 | 日 期 | 2020.5.21 |
| 班级 | 17计算机软件一班 |  |  |
| **实验内容** | | | |
| **[要求先在OJ上提交通过]**  1）填格子[OJ]  2）不如走楼梯[OJ] | | | |
| **实验目的** | | | |
| 完成两个实验题目 | | | |
| **实验过程和步骤** | | | |
| **【实验题目】**  **题目描述：**  **有一个由数字 0、1 组成的方阵中，存在一任意形状的封闭区域，封闭区域由数字1 包围构成，每个节点只能走上下左右 4 个方向。现要求把封闭区域内的所有空间都填写成2 .**  **例如： 6×6 的方阵：**  **6**  **0 0 0 0 0 0**  **0 0 1 1 1 1**  **0 1 1 0 0 1**  **1 1 0 0 0 1**  **1 0 0 0 0 1**  **1 1 1 1 1 1**  **填写后如下：**  **0 0 0 0 0 0**  **0 0 1 1 1 1**  **0 1 1 2 2 1**  **1 1 2 2 2 1**  **1 2 2 2 2 1**  **1 1 1 1 1 1**  **输入要求：**  **每组测试数据第一行一个整数 n(1≤n≤30)**  **接下来 n 行，由 0 和 1 组成的 n×n 的方阵。**  **封闭区域内至少有一个0 。**  **输出要求：**  **已经填好数字 2 的完整方阵。**  **注意矩阵的每个数字后面都有一个空格**  **【程序代码及注释】**  #include<iostream>  using namespace std;  int num[100][100];//来记录每个格子应该填写的状态  int res[100][100]={0};//防止无限循环递归  int n;  void DFS(int a,int b)  {  if(num[a][b]==1||num[a][b]==-1)//当当前格子已经确定状态后，返回  return ;  if(a==0||b==0||a==n-1||b==n-1)//如果为边界，直接定义为-1（就是不需填写内容）  num[a][b]=-1;  if(a!=0&&res[a-1][b]!=1)//向上  {  res[a][b]=1;  DFS(a-1,b);  if(num[a-1][b]==-1)//说明没有被包围，填写为-1  num[a][b]=-1;  }  if(a!=n-1&&res[a+1][b]!=1)//向下  {  res[a][b]=1;  DFS(a+1,b);  if(num[a+1][b]==-1)  num[a][b]=-1;  }  if(b!=0&&res[a][b-1]!=1)//向左  {  res[a][b]=1;  DFS(a,b-1);  if(num[a][b-1]==-1)  num[a][b]=-1;  }  if(b!=n-1&&res[a][b+1]!=1)//向右  {  res[a][b]=1;  DFS(a,b+1);  if(num[a][b+1]==-1)  num[a][b]=-1;  }  }  int main()  {  cin>>n;  for(int i=0;i<n;i++)//初始化  for(int j=0;j<n;j++)  cin>>num[i][j];  for(int i=0;i<n;i++)  for(int j=0;j<n;j++)  if(num[i][j]==0)//只有为0时需要进行填写判断  {  for(int l=0;l<n;l++)//初始化res数组，来防止进行死循环  for(int k=0;k<n;k++)  res[l][k]=0;  DFS(i,j);  }  for(int i=0;i<n;i++)//修改填写的内容，使之符合题目要求  {  for(int j=0;j<n;j++)  {  if(num[i][j]==-1)  num[i][j]=0;  else if(num[i][j]==1)  num[i][j]=1;  else if(num[i][j]==0)  num[i][j]=2;  }  }  for(int i=0;i<n;i++)//进行输出  {  for(int j=0;j<n;j++)  cout<<num[i][j]<<' ';  cout<<endl;  }  //system("pause");  return 0;  }  **【实验结果及遇到的问题】**    遇到的问题：  （1）一开始忘记进行重复的判断，导致在递归时出现死循环  （2）在对于添加防止死循环判断时，忘记修改之前的代码，导致出现了许多错误，列如：遇到之前走过的格子就直接将现有的格子标记为-1，导致之后输出的除了1就是2.  **【算法分析】**  这个实验是对于一个n\*n的方正中，将被1包围的全部0填写为2，并且题目提示只有一个区域是被1所包围，所以只要将没有被1包围的0标记为-1，通过循环，遍历全部为0的点，每一个0点都使用递归的方法，将直接和间接与边界相连接的0标记为-1.之后将-1输出为0,1输出为1,0输出为2。  **【实验题目】**  **题目描述：**  **有个电梯，每一层楼都可以停，只是算法混乱了，所以你得写个补丁；第i层楼(1<=i<=N)上有一个数字Ki(0<=Ki<=N)，表示上或下的层数（相对于当前层），每层楼都可以上或下。当然，如果不能满足要求（没有的层），相应的按钮就会失灵。例如：3 3 1 2 5代表了Ki（在第一层可以上或下3层；当然下是不可能的，第三层可以上或下1层），从一楼开始。在一楼，按“上”可以到4楼，按“下”是不起作用的，因为没有-2楼。那么，从A楼到B楼至少要按几次按钮？**  **输入要求：**  **共二行。**  **第一行为3个用空格隔开的正整数，表示 N,A,B（共基层，开始层，结束层）；(1≤N≤200, 1≤A,B≤N)N,A,B(1≤N≤200,1≤A,B≤N)。**  **第二行为N个用空格隔开的非负整数，表示每层按钮的数值Ki。**  **输出要求：**  **一行，即最少按键次数；若无法到达，则输出−1。**  **【程序代码及注释】**  #include<iostream>  using namespace std;  int num[201];//记录每个楼层可以上下的层数  int res[201]={0};//记录是否到达过这个楼层  int m=0;  void DFS(int a,int b,int k)  {  if(res[a]==1)  return ;  if(a==b)//记录最小的按键次数  {  if(m!=0)  m=min(m,k);  else m=k;  return ;  }  res[a]=1;//标记已经到达过  if(a+num[a]<=b)//向上  DFS(a+num[a],b,k+1);  if(a-num[a]>0)//向下  DFS(a-num[a],b,k+1);  }  int main()  {  int n,A,B;  cin>>n>>A>>B;  for(int i=1;i<=n;i++)//初始化  {  cin>>num[i];  }  DFS(A,B,0);  if(m==0)//输出  m=-1;  cout<<m<<endl;  //system("pause");  return 0;  }  **【实验结果及遇到的问题】**    遇到问题：  一开始没有理解题目，以为是可以坐任意的楼层数，并不是每一层所能走的楼层数都是固定的  **【算法分析】**  这次实验只需要从起始点开始，进行递归，判断是否可以向上或向下，可以的话就对于向上或向下之后的点来进行递归，并且标记一下这一层，防止出现重复到达同一层的情况，通过不断的递归到达终点后，进行最小按键点的判断。 | | | |