**【算法】老鼠走迷宮問題的解答**

時間:2012-10-21 01:01來源:Internet 作者:Internet 點擊: 272 次

一．題目：       用一個m行n列的二維數組來表示迷宮。數組中每個元素的取值为0或1。其中值0表示通路，值1表示阻

|  |
| --- |
| **一．題目：**        用一個m行n列的二維數組來表示迷宮。數組中每個元素的取值为0或1。其中**值0表示通路，值1表示阻塞**，迷宮的入口在左上放（1，1）處，出口在右下方（m，n）處。要求求出從迷宮入口到出口有無通路，若有通路則指出其中一條通路的路徑，即輸出找到通路的迷宮數組，其中通路上的“0”用另外一個數字8替換，同時打印出所走通路徑上每一步的位置坐標及下一步的方向。  **二．算法說明：**      （1）以二維數組maze[m][n]表示迷宮，並設maze[1][1]處为迷宮入口，maze[m][n]處为迷宮出口，迷宮中的任一位置以maze[i][j]來表示。      （2）對於迷宮中的每個位置（i，j）處，可能移動的路線可以有八個方向，用一個二維數組move表示這八個方向上**坐標的增量**，並把這八個方向從**正東起按順時針方向編上序號**，則                     move[k][0]表示第k個方向上i的增量，              move[k][1]表示第k個方向上j的增量。              move數組的方向增量表內容如下：k  1  2  3  4  5  6  7  8                 int move[8][2]={      {0,1},    //正東   i不變   j+1  向右                                                  {1,1},    //右下    i+1   j+1                                                  {1,0},    //下  i+1  j不變                                                 {1,-1},   //                                                {0,-1},   //                                                {-1,-1},  //                                                {-1,0},   //                                                {-1,1}};  //       （3）當處於迷宮邊緣時，它的下一個位置不再有八種可能，甚至只有三種可能。所以，为了簡化邊界位置的檢測，將二維數組maze[m][n]擴充到maze[m+2][n+2]，且令其四周邊界位置的值均为1。       （4）計算機解迷宮，要用一步一試探的方法。为此在開始每一步時，都要從正東方向起，沿順時針方向檢測。當探測到某個方向上下一個位置的值为0時，就沿着此方向走一步，當這一步周圍剩下的七個方向的上的值均为1時，則退回一步重新檢測下一方向。在這過程中，建立一個mark[m+2][n+2]數組來記錄某位置是否走過，走過用1記，未走過用0記。據此，可以用遞歸的思路來解决該問題。       （5）具體的算法可以概括如下：               走迷宮過程中，如果當前位置（i，j）（初始以入口为當前位置）已到達出口，即（i，j）=（m，n），則說明已找到一條通路，返回值1，表示走通，結束遞歸過程；               如果當前位置的各個方向上都沒有找到通向出口的路徑，則遞歸返回值0，表示未走通，若此時的位置在入口處，给出“沒有通路”的信息，結束遞歸；               如果對當前位置的各個方向依次試探的過程中，發現某個方向的試探位置可以走（即maze與mark數組中該位置的值均为0），則把試探位置作为調用参數遞歸調用走迷宮過程，同時對調用的返回值進行判斷，若調用返回值为1，則表示當前位置在走通的路徑上，因此要記錄下該位置，且遞歸返回1。               在記錄走通的每步位置時，考慮到遞歸的特點，若要正序打出走通的路徑，我們則可以另入口和出口顛倒，即可實現目的。  三，源碼  #include <iostream>  using namespace std;  const int m=12,n=15;//迷宮的大小  int maze[m+2][n+2]={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,//輸入迷宮地圖  1,0,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,  1,1,0,0,0,1,1,0,1,1,1,0,0,1,1,1,1,  1,0,1,1,0,0,0,0,1,1,1,1,0,0,1,1,1,  1,1,1,0,1,1,1,1,0,1,1,0,1,1,0,0,1,  1,1,1,0,1,1,1,1,0,1,1,0,1,1,0,0,1,  1,1,1,0,1,0,0,1,0,1,1,1,1,1,1,1,1,  1,0,0,1,1,0,1,1,1,0,1,0,0,1,1,1,1,  1,0,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,  1,0,0,1,1,0,1,1,0,1,1,1,1,1,0,1,1,  1,1,1,0,0,0,1,1,0,1,1,0,0,0,1,0,1,  1,0,0,1,1,1,1,1,0,0,0,1,1,1,1,0,1,  1,0,1,0,0,1,1,1,1,1,0,1,1,1,1,0,1,  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};  int mark[m+2][n+2];//設立迷宮是否走過標志  int move[8][2]={{0,1}, //正東 i不變 j+1 向右  {1,1}, //右下 i+1 j+1  {1,0}, //下 i+1 j不變  {1,-1}, //  {0,-1}, //  {-1,-1}, //  {-1,0}, //  {-1,1}}; //  //方向指標  int SeekPath(int x,int y)  {  int i,g,h;  if(x==1&&y==1) //如果到終點則找到路徑,返回 1  return 1;  for (i=0;i<8;i++)//嘗試每一個方向  {  g=x+move[i][0];  h=y+move[i][1]; //探索地點的新坐標    if(maze[g][h]==0&&mark[g][h]==0)//如果該地點走得通且沒有被探索過  {  mark[g][h]=1;//將這一地點置为探索過  if(SeekPath(g,h))//從這一地點開始新的探索,如果成功  {  cout<<"("<<g<<","<<h<<")";//則打出這一點的坐標  if(move[i][0]==1) cout<<"North->";  if(move[i][0]==-1) cout<<"South->";  if(move[i][1]==1) cout<<"West->";  if(move[i][1]==-1) cout<<"East->";//判斷前一地點到這一地點的方向  cout<<endl;  maze[g][h]=8;//把這一點設为通路  return 1;//返回1  }  }  }  if(x==m&&y==n)  cout<<"No path!"<<endl;//如果最後回到了起點,則說明沒有通路  return 0;//返回0  }    int main()  {  int i,j;  for (i=0;i<m+2;i++)  for (j=0;j<n+2;j++)  mark[i][j]=0;//先將所有通路置为沒有走過    mark[m][n]=1;//將起點置为走過了  if(SeekPath(m,n)) //如果走通  {  cout<<"("<<m<<","<<n<<")"<<endl;//先打出起點坐標  maze[m][n]=8;//將起點設为通路  for (i=0;i<m+2;i++)  for (j=0;j<n+2;j++)  {  cout<<maze[i][j]<<" ";  if(j==n+1)  cout<<endl;  }//打印出走通後的迷宮  }  }    From:[CSDN](http://blog.csdn.net/tianshuai11/article/details/8093925) |