Bài toán tìm đường đi trong mê cung sử dụng duyệt theo chiều rộng với hàng đợi

|  |
| --- |
| 6 8  0 1 1 1 1 1 0 0  1 0 0 0 0 1 0 0  0 1 0 0 0 1 1 0  0 0 1 1 0 0 1 1  1 0 0 0 0 0 0 0  1 1 1 1 1 1 1 0 |

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define MAX 100    typedef struct Node{  int row,col;// chỉ số hang và cột của trạng thái hiện tại  int step; // số bước di chuyển để đi từ trạng thái xuất phát đến trạng thái hiện tại  struct Node\* next; // con trỏ đến phần tử tiếp theo trong hàng đợi  struct Node\* parent;// con trỏ trỏ đến trạng thái sinh ra trạng thái hiện tại  }Node;      int n,m;  int r0,c0;  int A[MAX][MAX];    // di chuyen theo 4 huong  const int dr[4] = {1,-1,0,0}; //(dr[0],dc[0]) -->(1,0) = di chuyen sang phai 1 o  const int dc[4] = {0,0,1,-1};  Node\* finalNode;    int visited[MAX][MAX];      // con tro dau va cuoi hang doi  Node\* head, \*tail;    // CODE thua, khong can phai dung den bo nho phu nay!  Node\* listNode[MAX\*MAX];// mảng lưu các phần tử được cấp phát động, để giải phóng BN  int szList = 0;// số phần tử của listNode      void input(){  FILE\* f = fopen("maze.txt","r");  fscanf(f,"%d%d",&n,&m);  for(int i = 0; i < n; i++){  for(int j = 0; j < m; j++){  fscanf(f,"%d",&A[i][j]);  }  }  fclose(f);  }    void printCurrMaze()  {  printf("size %d x %d\n",n,m);  for(int i = 0; i < n; i++){  for(int j = 0; j < m; j++){  printf("%d ",A[i][j]);  }  printf("\n");  }  printf("\n");  }    // tao nut moi cua danh sach lien ket  Node\* makeNode(int row, int col, int step, Node\* parent){  Node\* node = (Node\*)malloc(sizeof(Node));  node->row = row; node->col = col; node->next = NULL;  node->parent = parent; node->step = step;  return node;  }    void initQueue(){  head = NULL; tail = NULL;  }  int queueEmpty(){  return head == NULL && tail == NULL;  }  void pushQueue(Node \* node){  if(queueEmpty()){  head = node; tail = node;  }else{  tail->next = node; tail = node;  }  }  // lay va tra ve phan tu dau hang doi  Node\* popQueue(){  if(queueEmpty()) return NULL;  Node\* node = head; head = node->next;  if(head == NULL) tail = NULL;  return node;  }      // ham lien quan den thuat toan  // cai nut hien tai co the them vao hang doi  int legal(int row, int col){  return A[row][col] == 0 && !visited[row][col];  }    // den cai o co the thoat ra me cung chua?  // cac o ma o bien la co the thoat ra  int target(int row, int col){  return row < 1 || row >= n-1 || col < 1 || col >= m-1;  }    // giai phong bo nho  void finalize(){  for(int i = 0; i < szList; i++){  free(listNode[i]);  }  }    // dung de giai phong bo nho  void addList(Node\* node){// them phan tu vao listNode de thuc hien giai phong bo nho  listNode[szList] = node;  szList++;  }    int main()  {  printf("MAZE\n");  input();  printCurrMaze();    // gan cac o la chua tham  for(int r = 0; r < n; r++)  for(int c = 0; c < m; c++)  visited[r][c] = 0;    // toa do o bat dau  r0 = 2; c0=3;    initQueue();  Node\* startNode = makeNode(r0,c0,0,NULL);  addList(startNode);  pushQueue(startNode);  visited[r0][c0]= 1;  while(!queueEmpty()){  Node\* node = popQueue();  printf("POP (%d,%d)\n",node->row,node->col);  for(int k = 0; k < 4; k++){  int nr = node->row + dr[k];  int nc = node->col + dc[k];  if(legal(nr,nc)){  visited[nr][nc] = 1;  Node\* newNode = makeNode(nr,nc,node->step + 1, node);  addList(newNode);  if(target(nr,nc)){  finalNode = newNode; break;  }else  pushQueue(newNode);  }  }  if(finalNode != NULL) break;// found solution  }  Node\* s = finalNode;  while(s != NULL){  printf("(%d,%d) ",s->row,s->col);  s = s->parent;  }  finalize();    return 0;  } |

|  |
| --- |
| MAZE  size 6 x 8  0 1 1 1 1 1 0 0  1 0 0 0 0 1 0 0  0 1 0 0 0 1 1 0  0 0 1 1 0 0 1 1  1 0 0 0 0 0 0 0  1 1 1 1 1 1 1 0    POP (2,3)  POP (1,3)  POP (2,4)  POP (2,2)  POP (1,4)  POP (1,2)  POP (3,4)  POP (1,1)  POP (4,4)  POP (3,5)  POP (4,5)  POP (4,3)  POP (4,6)  (4,7) (4,6) (4,5) (4,4) (3,4) (2,4) (2,3) |

Bài tập

* In ra đường đi cho đúng thứ tự xuôi
* In ra độ dài đường đi
* Sửa lại điều kiện thoát ra, chỉ cho thoát ra nếu đạt tới 1 ô có tọa độ cho trước nào đó (rt, ct nhập từ bàn phím). Nếu mà KHÔNG thể đạt tới ô đó thì in ra thông báo là KHÔNG tìm được đường đi thỏa mãn. ==> hàng đợi rỗng

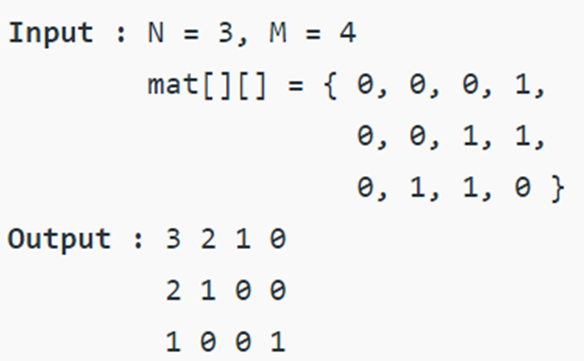
|  |
| --- |
| #include <stack>  using namespace std;  Node\* s = finalNode;  stack<Node\*> path;  printf("Do dai duong di: %d\n",s->step);    while(s != NULL){  path.push(s);  //printf("(%d,%d) ",s->row,s->col);  s = s->parent;  }  while(!path.empty())  {  Node\* s = path.top();  path.pop();  printf("(%d,%d) ",s->row,s->col);  } |

Nếu muốn tìm đường đi dài nhất ?

In ra tất cả các đường thoát?

=============================================

**Bài tập về nhà 1.**



Cho ma trận nhị phân chứa n hàng và m cột đọc vào từ file txt

|  |
| --- |
| 3 4 0 0 0 1  0 0 1 1  0 1 1 0 |

Cần in ra đường đi ngắn nhất từ các ô tới ô có giá trị 1  
Nếu ô hiện tại là số 1 thì độ dài đường đi là 0  
Chỉ có thể đi theo trái, phải, lên trên, xuống dưới

* Cách 1 dùng duyệt theo kiểu vét cạn --> O(N2M2)
* Cách 2: Dùng hàng đợi

Với ma trận ban đầu

|  |
| --- |
| 0 0 0 1  0 0 1 1  0 1 1 0  Các ô ban đầu chưa tính sẽ nhận giá trị -  Các ô ban đầu là 1 sẽ nhận giá trị là 0  - - - 0  - - 0 0  - 0 0 -  Thêm các ô giá trị 0 vào hàng đợi  Lần lượt lấy từng ô ra khỏi hàng đợi, xét các ô kề với ô hiện tại (có thể có tối đa 4 ô kề) mà chưa được gán giá trị ==> gán giá trị bằng giá trị ô hiện tại +1 (đẩy vào queue)  Lặp lại cho tới khi hàng đợi rỗng  - - - 0  - - 0 0  - 0 0 -  Xét ô (0,3)  Ô này kề với 1 ô mà chưa được xét là ô 0,2 --> gán giá trị cho ô 0,2 là 0+1 = 1  - - 1 0  - - 0 0  - 0 0 -  =====  - - 1 0  - 1 0 0  1 0 0 1  ========  - 2 1 0  2 1 0 0  1 0 0 1  =======  3 2 1 0  2 1 0 0  1 0 0 1  T(n) = O(MN) |