C++方向编程题答案

> **答案说明：**
>
> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：鲍松山 qq ： 365690203
>
> 代码题：鲍松山 qq：365690203

第四周

day24

> **题目ID：26020-年终奖**
>
> 链接：https://www.nowcoder.com/practice/72a99e28381a407991f2c96d8cb238ab?tpId=49&&tqId=29305&rp=1&ru=/activity/oj&qru=/ta/2016test/question-ranking
>
> **【题目解析】：**
>
> 本题题意明确
>
> **【解题思路】：**
>
> 本题需要使用动态规划求解。
>
> 定义f(i,j)表示从左上角走到坐标(i，j)处能获得的最大奖励。
>
> 搜索所有从左上角走到右下角的路径，找到最优路径。
>
> f(i,j)分三种情况：
>
> 第一列：f(i, 0) = f(i-1, 0) + board(i, 0)
>
> 第一行：f(0,j) = f(0, j - 1) + b(0, j)
>
> 其它位置：f(i, j) = max{f(i-1, j), f(i, j - 1)} + board(i, j)
>
> 最后返回右下角的值。
>
> **【示例代码】**

```
class Bonus {
public:
 int getMost(vector<vector<int> > board)
 {
     // write code here
     int row = board.size();
     int col = board[0].size();

     vector<vector<int>> allPrice(row, vector<int>(col, 0));
     allPrice[0][0] = board[0][0];
```

比特就业课

```
    for(int i=0; i<row; ++i)
    {
           for(int j=0; j<col; ++j)
        {
             //如果是起点坐标，不做任何处理。
               if(i==0 && j==0)
                   continue;
            if(i == 0) //在第一行，只能往右走
                allPrice[i][j] = allPrice[i][j-1] + board[i][j];
            else if(j == 0) //在第一列，只能往下走
                   allPrice[i][j] = allPrice[i-1][j] + board[i][j];
               else
                  //除去两个临界边，剩下的就是既能向右走，也能向下走，
                  //这时候就要考虑走到当前点的所有可能得情况，也就是走到当前点
                  //各自路径的和是不是这些所有到达该点路径当中最大的了
                   allPrice[i][j] = max(allPrice[i][j-1], allPrice[i-1][j]) + board[i][j];
           }
       }
       // 返回最后一个坐标点的值，它就表示从左上角走到右下角的最大奖励
    return allPrice[row-1][col-1];
 }
};
```

题目ID:36867-迷宫问题

链接：https://www.nowcoder.com/practice/cf24906056f4488c9ddb132f317e03bc?tpId=37&&tqId=21266&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking

【题目解析】：

本题题意明确

【解题思路】：

本题可用回溯法求解 具体步骤为：

1. 首先将当前点加入路径，并设置为已走
2. 判断当前点是否为出口，是则输出路径，保存结果；跳转到4
3. 依次判断当前点的上、下、左、右四个点是否可走，如果可走则递归走该点
4. 当前点推出路径，设置为可走

【示例代码】

```
#include<iostream>
#include<vector>
using namespace std;

int ROW, COL;
vector<vector<int>> maze;
vector<vector<int>> path_tmp; //临时路劲
vector<vector<int>> path_best; //最佳路劲

void MazeTrack(int i, int j)
{
```

比特就业课

```
    maze[i][j] = 1; //代表 (i,j) 已经走过
    path_tmp.push_back({i,j});

    //判断是否到达出口
    if(i==ROW-1 && j==COL-1)
    {
        //寻找最短路劲
        if(path_best.empty() || path_best.size()>path_tmp.size())
            path_best = path_tmp;
    }

    //向右走
    if(j+1<COL && maze[i][j+1]==0)
        MazeTrack(i, j+1);
    //向左走
    if(j-1>=0 && maze[i][j-1]==0)
        MazeTrack(i, j-1);
    //向上走
    if(i-1>=0 && maze[i-1][j]==0)
        MazeTrack(i-1, j);
    //向下走
    if(i+1<ROW && maze[i+1][j]==0)
        MazeTrack(i+1, j);

    maze[i][j] = 0; //回溯 恢复路劲
    path_tmp.pop_back();
}

int main()
{
    while(cin >> ROW >> COL)
    {
        maze = vector<vector<int>>(ROW, vector<int>(COL, 0)); //开辟迷宫空间
        path_tmp.clear();
        path_best.clear();
        //首先输入迷宫
        for(int i=0; i<ROW; ++i)
        {
            for(int j=0; j<COL; ++j)
                cin>>maze[i][j];
        }

        MazeTrack(0, 0); //从起始点 (0, 0) 开始走

        //输出路径
        for(int i=0; i<path_best.size(); ++i)
        {
            cout<<"("<<path_best[i][0]<<","<<path_best[i][1]<<")"<<endl;
        }
    }
    return 0;
}
```

比特就业课

比特就业课