Programming Homework - Poison Maze

ver 1.1

2021/12/11

1 Problem Description

The poison maze is a square room full of poisonous liquid, every step you go, you get hurt. The maze can be described as an N×N grid. For better understanding of the input data, we define +x direction is **RIGHT** and +y direction is **DOWN**. Depending on the density of poison, walking on different grid points results in different amounts of damage.

You start at point $(1, 1)$ (top left), and there is an exit at the diagonal corner (N, N) (bottom right). The damage at each grid point is given. You need to escape from this maze with as little damage as possible.

In this problem, you need to use **C/C++** to implement a search algorithm to find a path from $(1, 1)$ to (N, N) with the smallest total damage.

2 Search Algorithm

In general shortest path problems, We consider arbitrary graphs, not only grids, with weighted edges. These weight can be considered as distance between 2 nodes. There is a famous shortest-path algorithm called Dijkstra's Algorithm. The basic concept of Dijkstra's Algorithm is the following:

The algorithm keeps a "distance" on every node denoting the length of the current shortest path from $(1, 1)$ to this point. Initially, set the distance of (1,1) to its own number and distances of every other nodes to ∞ and push the starting point into a list (called the **openlist**). Nodes in the open list are reachable but not explored yet. The algorithm keep popping nodes with current smallest distance from the **openlist** one at a time, check if it is the target point, if not, update all its neighbors' distance to starting point if you found a shorter path and then push updated nodes into the **openlist**. Keep doing this until you pop the target point from the **openlist**. The algorithm ensures that the distance recorded at the target point will be the shortest distance between the starting point and the target. Since you need to pop nodes with smallest distance every time, it is common to use a MIN-heap to maintain the **openlist**.

If you use other data structure aside from MIN-heaps to construct the openlist, you can get different results with different time and space complexity. For example, if the openlist is implemented by a queue, then the algorithm will give you a path with the least number of steps. The algorithm is called Breadth-First Search (BFS for short). If the openlist is implemented by a stack, the algorithm will become Depth-First Search (DFS for short). Using key values different from the "distance" defined above also lead to different algorithms. Examples include Best-First Search or the A* Algorithm.

For this homework, you need to find the entire shortest path instead of just the shortest distance. To do this, each node needs to remember its predecessor in the path (besides the distance mentioned above). Furthermore, the weights are on the grid points instead of the edges.

3 Input

There will be 10 test data in 3 different sizes.

The first line of each input file contains an integer N, the maze will be a $N \times N$ Grid. Then there will be N lines following, every lines contain N integers. For the i-th line, the j-th number denote the damage of the point (i, j) when you walk on it.

4 Constraints

Small test data:
N=10, with known minimal damage.
Damage in each node is no more than 64

Median test data:
N=100
Damage in each node is no more than 256

Large test data:
N=1000
Damage in each node is no more than 65536

5 Output

For each test, please output a file with 2 lines:

In the first line, please output 2 integer C, S. C is the smallest possible damage (length of the shortest path), and S is the number of steps of this path.

In the second line, please output a string of $S - 1$ characters, consist of 4 characters: {'U', 'D', 'L', 'R'} (denoting "up", "down", "left", "right") describing every step from $(1, 1)$ to (N, N).

Please output your answers with filename: ID_test# , without any extensions

Example: B09901288_1

Please save your code with filename: ID_src.cpp/cc/c or other C/C++ compatible extension

Example: B09901288_src.cpp

Then put all output file alone with your code **directly** in a **ZIP** named ID_PH.zip without any folders in the zip file.

Example: B09901288_PH.zip

Finally, upload it to NTU COOL.

6 Samples

Sample Input

```
3
1 1 4
5 1 4
8 1 0
```

Sample Output

```
4 5
RDDR
```

7 Related Materials

A video describing Dijkstra's algorithm is provided on the class website.