EECS 204002 Data Structures 資料結構

Project #2: Floor Cleaning Robot 掃地機器人

1. Project Objective:
   Apply the knowledge learned from the course of data structures and implement an efficient robot floor cleaning algorithm.

2. Project Description
   You are to design an optimal floor cleaning algorithm to drive the robot. The floorplan is described as an $m * n$ matrix, in which each cell is marked with one of the following status.
   a. "1" indicates an obstacle, which normally is a wall.
   b. "0" indicates a free space to be cleaned.
   c. "R" indicates where the robot is placed initially. This is also where the robot can be recharged. Note that this cell can only be connected for recharge and cannot be passed through.

   The robot is to follow these rules:
   a. Always starts from the cell marked "R".
   b. Needs to clean every free cell "0" on the floor.
   c. Cannot move into or cross any obstacle cell "1" or move outside the floor boundary.
   d. Can only move *up, down, left* and *right*.
   e. The robot is initially charged with a battery life "B", which means the robot can move *B* steps before it runs out of power. Moving from one cell to next cell is counted as one step.
   f. Needs to return for recharge before running out of battery. Each recharge fully restores the battery life "B".
   g. After cleaning all free cells, needs to return to the cell "R" for recharge.

3. Test Case
   Every student has to design and submit a test case prepared as a text file, named "**floor.data**". The first line of the file contains three numbers, specifying the number of rows (*m*), the number of columns (*n*) and the battery life (*B*). Following the first line, there shall be *m* lines and each line shall have *n* numbers, each represents the corresponding cell status.

a. Example:

```
7  10 30
1  1  1  1  1  1  1  1  1  1
1  0  0  0  1  0  0  0  0  1
1  0  0  0  1  0  0  0  0  1
1  0  0  0  1  0  0  0  0  1
1  0  1  1  1  1  1  1  0  1
1  0  0  0  0  0  0  0  0  1
1  1  1  1  R  1  1  1  1  1
```

This case describes a floorplan with cells of 7 rows and 10 columns. The battery life allows the robot to move 30 steps for each fully charged run. We assume the row index and column index all starts from 0. Therefore, initially the robot is placed at cell (6, 4). Note that *R* can be at any, but only one, location.

b. Valid Test Case

Make sure your test case is valid. A valid test case should have at least one solution to clean every free cell. Invalid test case gets no score. The size of the floorplan should be no more than 1000*1000.

c. Test Case Competition

TAs will use all test cases collected from the class to evaluate your algorithm.

4. Result output file

Output your optimized final floor cleaning path into a file named "**final.path**". The first line should be the number of steps your algorithm used to finish the cleaning task and for the following lines, each should have two numbers *i* an *j*, representing the cell (*i*, *j*).

Example:

```
6
5  4
5  5
5  6
5  5
5  4
6  4
```

This example shows that the robot moves 6 steps. Obviously, this is not a complete floor cleaning

path and will receive no score.

5. Project Submission Rules
   a. Submit your test case and program to ILMS.
   b. Test case submission: the deadline will be one week before the program submission date.
   c. Final submission: submit your program and project report.
   d. Please use GitHub for source code control and show your program development history. Please follow the version control rules when doing programming.

6. Grading Policy
   a. Algorithm Quality (60%)
      - Basic Test (30%): TAs will provide three open test cases. For each test case your program can pass and find the correct solution, you receive 10% credit.
      - Quality Test (30%): Your implementation quality will be evaluated according to the following formula. For each test case *j*, and, we compute the numbers of execution time and memory usage wins for a person *i* to be $t_{i,j}$ and $m_{i,j}$. Simply, if your program executes faster or uses less memory than *w* other people, then the number of wins is *w*. The quality scores for a person *i* are calculated as the following.
        - Execution time score: $20\% * (\sum_j \frac{(t_{i,j}+1)}{\max_k t_{k,j}})/N$;
        - Memory usage score: $10\% * (\sum_j \frac{(m_{i,j}+1)}{\max_k m_{k,j}})/N$,

          where N is the total number of students in this class.
      - Illegal implementations receive zero scores: If your program cannot compile or execute on our testing platform, then you get no score. Also, you should use only standard C++11 library, and should not use assembly codes. Note that our testing platform is a simple machine which supports only standard CPUs, supports no GPU, nor other non-CPU instructions, and is not connected to internet.
   b. Test cases (20%)
      - Illegal test cases receive zero scores. Please make sure the matrix contains the correct number of elements in correct format, and all elements are of legal long integer numbers.
      - Your test case is to be measured according to how well it can differentiate good algorithms from bad algorithms in the following formula.
        $20\% * (1 - 10^{\sum_{i=1}^{n}(1-T_i/T_{min})/n})$,
        where *n* is the number of students.
   c. Report & Demo (20%)
      1. The report file should be named "**report.pdf**".
      2. Each person should reserve a 15-min demo with TA. During the 1-on-1 demo, TA's will

ask you questions related to your project report, test case and your implemented code. TA may ask you to compile and execute your program on spot.

3. Your project report is recommended to follow this outline:

1) Project Description 1-1) Program Flow Chart 1-2) Detailed Description 2) Test case Design 2-1) Detailed Description of the Test case

Note: The project report is limited to 10 pages.

Note: Your report can be either in Chinese or in English, or mixed.

Etiquette

a. **Do not plagiarize others' work, or you will fail this course.**
b. **No acceptance of late homework.**

Please frequently check the class website announcements for possible updates.