CS211 Data Structures and Algorithms
Aug -- Nov, 2019
Project Details

This is a team-based project. The problem you are supposed to solve is known as ‘Lumberjack’. The description of the problem is given at the end of this document.

Team: Every student must be a part of a team. A team must contain at least two and at most three students.

Evaluation: Maximum marks for this project is 100. The following is a tentative distribution of marks over various components of the evaluation.

First evaluation: 10 marks
Second evaluation: 10 marks
Third evaluation: 10 marks
Fourth evaluation: 10 marks
Fifth evaluation: 50 marks
Report, and/or presentation, and/or viva: 10 marks

The due-dates and detailed description for each of the evaluation will be intimated separately. For the first to fifth evaluation, the marks you obtain will be proportional to the total profit you make out of all the inputs available in Optil.io (currently there are 31 inputs). The team which obtains maximum (among all teams in the class) profit gets full marks in an evaluation.

The problem: The following details of the problem ‘Lumberjack’ are taken from Optil.io.

You are a lumberjack in a roadside forest. Every day you would like to cut down trees of the maximum value in the limited time that you have. We represent a forest as a rectangular grid of dimensions **n** by **n** points. Each edge between the neighboring grid points has a length of 1. At the position of each point, there could be a tree. Each tree is described using the following parameters:

- height **h**,
- thickness **d**,
- unit weight **c**,
- unit value **p**.

We calculate the value of each tree according to the formula: $p \cdot h \cdot d$. To cut down the tree, you have to reach it first. You can move only along the edges between the neighboring grid points. To cut down the tree at the position that you occupy you have to select one of four directions (along grid lines) in which the tree will fall. At the beginning of the day, you start in the lower-left corner of the forest, which is located at the point (0,0). You can assume that there is no tree there. If the tree that you cut falls on the other tree, which is lighter than the one that is falling, then this tree will also fall without being cut directly. We say that the tree is heavier if the total weight calculated as $c \cdot d \cdot h$ is greater than the total weight of the second tree. In this way, one can achieve a domino effect by pushing one tree onto another. We assume that a tree falls on the other when the distance between them is less the height of this tree. Weights of trees do not cumulate - we always take into account only the weight of the tree, which directly fell onto another. If for example tree A falls onto tree B and its weight is greater than weight of tree B then it can push tree B onto another tree C located in the same direction if the distance between B and C is smaller to height of B and, moreover, tree C is lighter than tree B. When comparing trees B and C we do not take into account tree A that lays on tree B. Moreover, if tree B is not lighter than tree A then it blocks tree A and it will not have a chance to fall onto tree C.

Calculate the biggest profit that you can reach during **t** time units and provide a corresponding sequence of movements. Assume that the walk between the two neighboring grid points takes 1 unit of time, and cutting the tree takes **d** units of time.

INPUT

The first line of input contains three integers: 0 < **t** <= 5,000 representing the time limit for the lumberjack, 0 < **n** <= 1,000 representing the size of the grid, and 0 < **k** <= 10,000 denoting the number of trees. In the following **k** lines there is a description of consecutive trees consisting of six integer numbers 0 <= **x**, **y** < **n**, 0 < **h**, **d** <= 20, 0 < **c**, **p**<= 500 representing the position of the trees on the grid, their height, thickness, weight and value. Your program should read input from the standard-in.

OUTPUT

On the output specify a list of moves (using **move** prefix) and cuts (using **cut** prefix) with the suffix denoting the direction of the action (**up**, **right**, **down** or **left**), each on a separate line. The output of your program should be in standard-out.

SCORING

For each test case, your program will receive a score equal to the total value of trees that were cut down by the lumberjack. You should maximize this value. For each test with **n** <= 25, your program should execute below 1 second, for 25 < **n** <= 100, your program should execute below 10 seconds, and for **n** > 100 your program should execute below 60 seconds. For each test the available memory is limited to 1 GB.

EXAMPLE INPUT

```
11 10 5
2 3 4 5 2 2
2 6 3 1 1 3
2 7 2 2 2 4
5 5 10 3 1 5
4 3 5 5 2 6
```

EXAMPLE OUTPUT

```
move right
move right
move up
move up
move up
cut up
move right
```