Felicity Threads 2k14

Strange Loop

Problem Set

Contest Rules and Format

1. This document is the problem statement for Strange Loop, the combinatorial search and AI contest as part of Felicity Threads 2014.
2. There are three problems as part of the contest. Problems description and specifications are provided below.
3. The Online Judge for the contest will go live on 18th January, 2014 at 1800 hours IST (UTC + 5:30)
4. Further updates will be provided on the event portal. Link to event portal: http://felicity.iiit.ac.in/threads/strangeloop/. You need to register on threads site, if you haven't registered before on order to submit solutions. Use this link to register: http://felicity.iiit.ac.in/threads/register/.
5. For any queries or doubts, mail us at strangeloop@felicity.iiit.ac.in

Ruby Challenge : Magnets are Forever

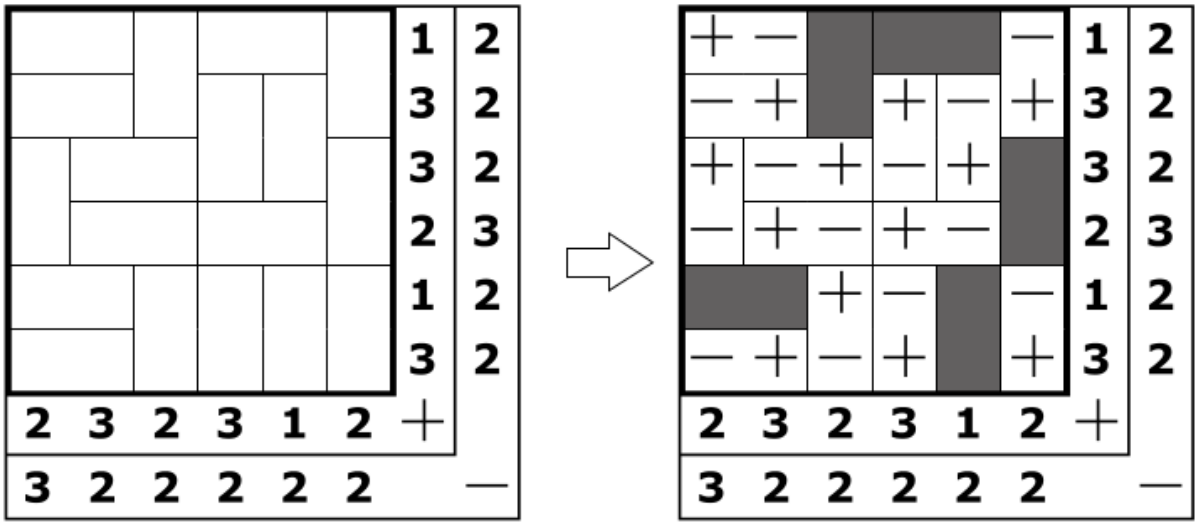
Magnets is an intriguing and quite a simple puzzle. You are given a square board of dimensions NxN. In this board, you have a domino tiling. As a result of the tiling, the tiles in the board have to be filled by magnets. Filling up a tile is done by placing a north pole at one space and the south pole at the other space of the tile. Or, you could choose to not place a magnet in the given spot. But, there's always something to make the game harder. The rightmost column of the diagram represents the number of south poles (-), i.e., the i-th entry of the column has the number of south-poles in the i-th row of the board. Similarly the 2nd column from the right represents the number of north poles (+). The bottom most row represents the number of south poles (-), i.e., the i-th entry of the row contains the number of south poles in the i-th column of the board. And similarly the 2nd row from the bottom that represents the number of north poles (+).

Your task is the populate the board with magnets.

Observe the image and read the description to obtain greater clarity of thought.

Oh, and in case you didn't notice, no two like poles are in contact. Ever!

Also note that a puzzle could have multiple solutions, but you only need to generate one feasible solution.



Puzzle Credits: Andrey Bogdanov

Input Format:

The first line contains *t*, the number of test cases. This is followed by *t* sets of lines. The first line in this set is *n*, i.e., the size of the puzzle. The next *n* lines contain the details of the table. So, each of these *n* lines is *n* integers long. The integers are in order from left to right and they denote the the magnet_no of the magnet that the space belongs to. After these *n* lines, there are 4 lines. The first line contains the number of north poles in each row and the second

contains the number of south poles in each row. The third line contains the number of north poles in each column and the fourth line contains the number of south poles in each column.

Output Format:

Print an NxN grid. A '+' is printed in place of a north pole and a '-' in place of a south pole. A neutral spot is filled by 0.

Sample Input:

This input corresponds to puzzle in the picture.

```
1
6
1  1  2  3  3  4
5  5  2  6  7  4
8  9  9  6  7  10
8  11 11 12 12 10
13 13 14 15 16 17
18 18 14 15 16 17
1 3 3 2 1 3
2 2 2 3 2 2
2 3 2 3 1 2
3 2 2 2 2 2
```

Sample Output:

```
+-000-
-+0+-+
+-+-+0
-+-+-0
00+-0-
-+-+0+
```

Scoring:

1. There will be three different test files on which your code will be tested. The files will be of increasing difficulty and size - Easy, Medium, and Hard. Each file will contain several test cases and you will be given partial marks for the problem only if your solution passes all the test cases of a given file.
2. Each test file will contain some partial marks, say 20 marks for Easy file, 30 for Medium file and 50 for Hard file. This is just tentative distribution, actual distribution of marks may be different from this.
3. After passing 1 or more rounds, your code will be assigned a score. This score will now be scaled down by the ratio of time taken by your solution to the time taken by the

best solution of the contest (in terms of performance). Say your code got 50 marks, i.e. passed Easy and Medium tests. If it took X seconds to complete the solution and the most optimal solution took Y seconds, your score will be scaled down to (50 * X)/Y.

4. Ranks will be calculated by sorting the codes in the descending order of their scores.

Sapphire Challenge : The Revenge of the Cubes

You are given a 4x4x4 Rubik's Cube (a.k.a. Rubik's Revenge), which you must solve. By solving we mean that you must reach a configuration where every square on each face has the same colour (here, represented by a digit between 1 and 6, inclusive). The moves you can perform are as follows:

pFn
pBn
pUn
pDn
pLn
pRn

where F = front, B = back, U = up, D = down, L = left, R = right.
(Orientation: Image that the cube is suspended in front of you. The face that you can see is the front face.)
p = the number of layers being rotated,
and *n* = the number of clockwise quarter turns performed to complete the move
(Thus $0 < p < 3$ and $0 < n < 4$)

For example, 1F2 denotes the frontmost layer being turned clockwise (by 90 degrees) twice.
Similarly, 2F1 denotes the front two layers being turned clockwise (by 90 degrees) once.
Given a configuration of the cube, you need to print the moves you would use to solve it. The challenge is to minimize the number of moves you make.
One move of the form 'pKn' is considered as a single move. Basically, you are not evaluated by the number of quarter turns, i.e., 2F2 counts as a single move. (i.e. Face turn metric)

Input Format:

A cube's configuration is given to you as six 4x4 matrices of the form:

```
N1  N2  N3  N4
N5  N6  N7  N8
N9  N10 N11 N12
N13 N14 N15 N16
```

where each N*n* is a single digit between 1 and 6 (inclusive) representing the colour of that square.
There will be a blank line after every 4x4 matrix. The faces will be given in the order F, R, B, L, U, D.

Output Format:

For a cube, you must print each `pKn` move on a new line.

Sample input file:

```
2 2 2 2
1 1 3 3
1 1 3 3
1 1 3 3

1 1 3 3
2 2 2 2
2 2 2 2
2 2 2 2

2 2 2 2
3 3 1 1
3 3 1 1
3 3 1 1

1 1 3 3
4 4 4 4
4 4 4 4
4 4 4 4

6 6 6 6
6 6 6 6
5 5 5 5
5 5 5 5

6 6 5 5
6 6 5 5
6 6 5 5
6 6 5 5
```

Sample Output:

```
2R2
1U1
```

Scoring:

1. Scoring will be done by ranking the user submitted codes in decreasing order of the number of moves required to solve the puzzle. Another constraint is that, the code should terminate within some specific time and space which will be announced later.

Emerald Challenge : Fighters of the Lost Othello

Reversi is an intriguing puzzle. It is played on an 8x8 board. Let the rows be named from a to h and columns from 1 to 8. The top left corner is denoted as a1 and the bottom right as h8. To play the game, one requires two-sided coins, black on one side and white on the other. At the start of the game, there are 4 coins on the board, two (placed at d4 and e5) are white-up whereas the other two (placed at e4 and d5) are black up.

This is a two player game where each player is assigned a colour. On the board, if there's a horizontal, vertical or diagonal line of coins of a certain colour with a coin of the opposite colour at the end of the line, then if a coin of the above opposite colour is placed at the opposite end of the line (i.e., the line is "sandwiched"), then the line that got sandwiched gets flipped to the opposite colour.

Black moves first, and and at each move the player in turn must put a coin, with their color facing up, on one of the empty cells, adjacent to a coin of opponent's color, such that a line segment (vertical, horizontal, diagonal) of opponent's color is sandwiched by the move, leading to a flip.

If a player cannot make a move, then it is required that the other player keep making move(s) till the first player can make a move. When neither player can make a move, the game ends.

The player with highest number of coins of his colour that face upward wins. But, wait! That's the usual Reversi. This is Reverse Reversi. So, the player with least number of coins of his colour that face upward wins.

Input Format:

The first line of the input is the colour of the coin assigned to you (W or B). The input is an 8x8 board with the state of the coins. 'B' denotes that the black side of the coin faces up whereas 'W' indicates that the white side of the coin faces up. '0' (zero) denotes that that grid has no coin and that a coin may be placed there. Yes, coins are only placed on the empty cells!

Test Case #1

```
W
000000W0
WBBBBW00
000BBBW0
000BBBW0
00WWW0B0
BBBWW000
0000WWB0
WWWB0000
```

Test Case #2

```
B
000000W0
WBBBBW00
000BBBW0
000BBBW0
00WWWBB0
BBBWW000
0000WWB0
WWWB0000
```

Output Format:

You are given the state of the board and your program must output the optimal next move, in terms of the position where your coin should be places, so as to maximize your chances of winning.

The output is an alphanumeric string of 2 characters with the first a letter that denotes the row and the second a number that denotes the column number.

For example: d3 or any other alphanumeric string, K, where

K[0] >= ‘a’ and K[0] <= ‘h’

K[1] >= 1 and K[1] <= 8

Scoring:

1. The agents submitted by the users will be competed against one another and the best agent will be declared winner.
2. Final contest will be conducted using the last submitted agents be the users. Tournament will be of round robin format. If there are any ties in the top positions after completion of all the rounds, tie breaking will be done by conducting playoffs among the same ranked bots.
3. To give the users an idea of how their agents are performing with respect to other agents, a tentative rank-list will be published nearly after every 3 hours. This rank-list will be generated using double elimination on user submitted agents.