Snake Bastions

The Reply Code Masters Team

9th March 2023

Abstract

The ingenious Reply engineers have devised an innovative way to deploy defences on our clients' systems. They created the Snake Bastions, which protect multiple system parts in one shot, provided they're topologically contiguous.

They also invented an easy way to deploy the Snakes by representing a client's network as a matrix in the metaverse. Every cell is a single component in the system and is marked with a number indicating its relevance in the overall system. Furthermore, the network switches are represented as wormholes, capable of teleporting Snakes to guard a different part of the network as if they were contiguous.

As a skilled Reply cyber security engineer, you (the Player!!) have to find the best way to deploy Snakes on the client's system to maximise the relevance of protected parts overall.

Image created with Dall•e 2

1 Problem Statement

You must plan how to lay out the available S Snakes to cover as many highly relevant components as possible.

You can deploy each snake on the system, represented by a $R \times C$ matrix. Each snake is positioned by placing it on a cell and commanding it to slither to an adjacent cell either directly up, down, to the left, or to the right of the one just visited, until the snake reaches its total length. Snakes cannot overlap.

Thanks to the wonders of the metaverse, the boundaries of maps wrap around. For example, if a snake moves out from the grid by crossing the left border, it will magically reappear on the same row of the right border. Similarly, if a snake leaves the map from the upper border, it will reappear on the same column at the bottom border; and so on.

Some cells in the grid don't represent components, but network switches. These are wormholes where a snake can enter and teleport to any other wormhole in the map. Passing through a wormhole counts as a single move. Multiple snakes can use the same wormhole for both entering and exiting, provided they don't overlap their bodies before or after teleporting. Please, remember:

- Snakes cannot overlap with other Snakes or with themselves.
- Passing through a wormhole takes only one Snake segment.
- A Snake cannot start from, or end, in a wormhole.
- Multiple snakes can use the same wormhole, as long as they don't overlap on adjacent non-wormhole cells.
- If the head of a Snake moves beyond the western boundary of the map, it will reappear on the eastern boundary in the same row.
- If the head of a Snake moves beyond the eastern boundary of the map, it will reappear on the western boundary in the same row.
- If the head of a Snake moves beyond the northern boundary of the map, it will reappear on the southern boundary in the same column.
- If the head of the Snake moves beyond the southern boundary of the map, it will reappear on the northern boundary in the same column.

Your total score is the sum of the relevance of the cells covered by your Snakes.

The solution is valid only if the total score is greater than 0.

2 Input format

The input file is a regular ASCII text file. Each line of the input file is separated by a single "`\n`" character ("UNIX-style"). If a line has multiple data, each value is separated by a single whitespace character. The first row of the input file will have 3 integer numbers:

- C, indicating the number of columns of the system grid
- R, indicating the number of rows of the system grid

- S, indicating the number of Snakes available to the player

The second row is composed of S integer values, with each one corresponding to the length of a Snake. Then, R lines follow, with each one consisting of C values. Each value could either be:

- V_{rc}, the value of the relevance of the component in that position
- An asterisk (*), representing the presence of a wormhole in that position

3 Output format

The output file must be a regular ASCII text file. Each line of the output file must be separated by a single "`\n`". The i-th line of this file holds the layout of the i-th Snake. Each line consists of:

- two integers, c and r, representing the column and the row in which the head of the Snake is placed initially
- one uppercase letter – "U", "D", "L", "R" – indicating the Snake slithering one cell up, down, left, or right respectively
- two integers, c_w and r_w of the wormhole from which the Snake emerges if the Snake enters a wormhole.

All the values on a single line are separated by a single space character.

If, for any reason, you don't want to use a Snake, simply leave the line empty.

4 Scoring rules

Given a list S of deployed Snakes, S_i is the i-th Snake of length $l(S_i)$. Each Snake S_i can be seen as a sequence of $l(S_i)$ segments, S_{ij} with $0 < j < l(S_i)$.

Let's define $V(S_{ij})$ as the relevance of the system part covered by the segment j of the snake i, as defined in the input map.

If the solution respects all constraints, then the final score is:

$$\sum_{i=0}^{S-1} \sum_{j=0}^{l(S_i)-1} V(S_{ij})$$

5 Constraints

- All the indices, loops, and iterations start from 0.
- $R \leq 5.000$
- $C \leq 5.000$
- $S \leq 5.000$
- $1 \leq length(S) \leq 1.000$
- $-10.000 \leq V_{rc} \leq 10.000$

6 Example

6.1 Input file example

```
10 6 5
6 7 5 3 3
1 5 3 6 3 8 5 2 6 8
6 4 * 0 5 3 7 5 2 8
3 4 5 0 3 6 4 * 5 7
3 5 6 3 0 3 5 3 4 6
3 6 7 * 3 0 6 4 5 7
3 7 8 5 3 6 0 4 5 6
```

The system parts are laid out into a matrix with 10 columns and 6 rows, with the player counting on 5 Snakes to establish defences. The first Snake is composed of 6 cells, the second of 7, the third of 5, and the fourth and fifth of 3 cells.

In this case, the matrix is shown below:

1	5	3	6	3	8	5	2	6	8
6	4	*	0	5	3	7	5	2	8
3	4	5	0	3	6	4	*	5	7
3	5	6	3	0	3	5	3	4	6
3	6	7	*	3	0	6	4	5	7
3	7	8	5	3	6	0	4	5	6

6.2 Output file example

```
0 0 R R D 7 2 R R
6 1 L U L D L U
1 1 R 3 4 R R R
7 1 D 3 4 L
9 0 U L
```

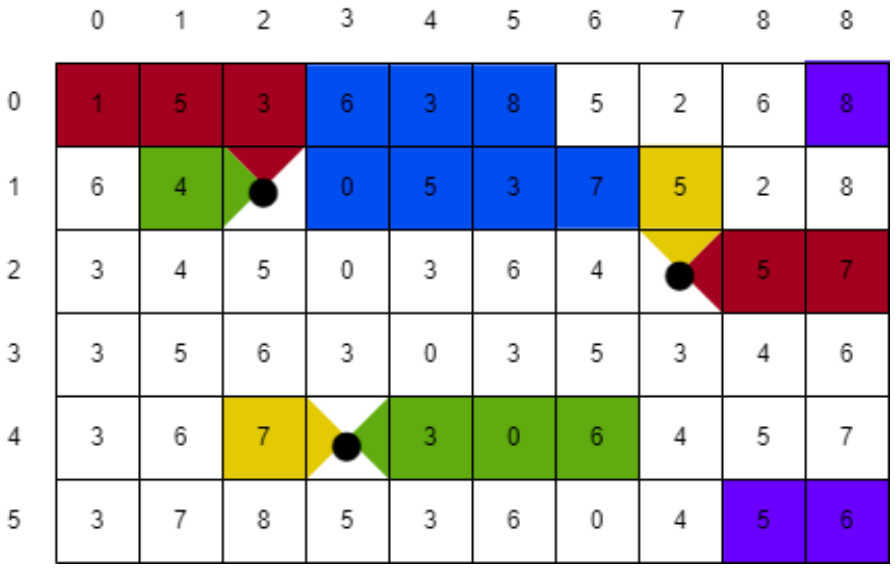
The first Snake is six segments long. It starts from the upper-left corner, moves right for 2 more cells and slithers down into the wormhole, exiting from the one in the eighth column, third row. It then slithers right for 2 other cells. 1 initial cell + 2 cells on the right + 1 wormhole traversal + 2 cells to the right = 6 segments.

The second Snake is seven segments long. By starting in the cell at the seventh column of the second row, it slithers left, up, left, down, left, and up again.

The third Snake is five segments long. It starts from the cell in the second column of the second row and moves to the right into the wormhole. It slithers out from the wormhole on the fourth column of the fifth row, then moves right for 3 cells.

The fourth Snake is three segments long. It starts in the eighth column, second row and slithers directly down into the wormhole. It intends to exit from the same wormhole as the third Snake, but given they cannot overlap, it chooses to move in the opposite direction: 1 cell to the left.

The last Snake is three segments long. It starts in the upper-right corner and slithers directly up, leaving the top border and reappearing in the lower-right corner. It then moves 1 cell on the left.



6.3 Scoring

The first Snake covers 1+5+3+5+7 = 21 anomalies.

The second Snake covers 7+3+8+3+5+0+6 = 32 anomalies.

The third Snake covers 4+3+0+6 = 13 anomalies.

The fourth Snake covers 5+7 = 12 anomalies.

The last Snake covers 8+6+5 = 19 anomalies.

The total player score would then be 21+32+13+12+19 = 97 total points.