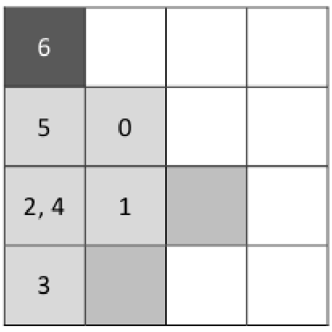
This problem consists of traversing deterministically a discrete surface represented by a matrix of $n \times n$ cells ($n > 2$), starting at a *source* cell, ending at a *target* cell, and considering the existence of static obstacles in the form of *unavailable* cells. The key idea is that from the *source* cell, the position of the *target* cell is unknown; thus, a path starting from *source* cell must be created on the run until *target* cell is found.

Only vertical and horizontal movements are allowed. The choice of the next cell visited is given by the following list of priorities.

1. Go to the *target* cell, if it is adjacent vertical or horizontally
2. Go down, if the bottom cell is available and not visited yet
3. Go to the right, if the right cell is available and not visited yet
4. Go to the left, if the left cell is available and not visited yet
5. Go up, if the top cell is available and not visited yet
6. Go back to the previous cell.

If either *source* cell or *target* cell is trapped in a dead end, eventually the path will return to the start point. In such a case, the travel must end.

For a better understanding of this problem, consider the first test case from Sample Input, where *source* cell is at (2, 2), *target* is at (1, 1), and cells (3, 3) and (2, 4) are unavailable. As can be seen in the following figure, the path from *source* to *target* considering the priorities enlisted above is as follows: 1) go down, 2) go to the left, 3) go down, 4) go back, 5) go up, 6) go to the *target*.



Input

The first line contains an integer $N > 0$ denoting the number of test cases.

The next N lines contain a space-separated list starting with an integer $n > 1$ denoting the number of rows (or columns) in the surface, and followed by $m \geq 2$ pairs of integers (i, j), such that:

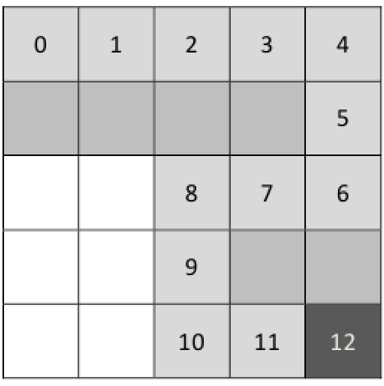
a) i is the column index
b) j is the row index
c) $1 \leq i, j \leq n$
d) The first pair (i_1, j_1) denotes the position of the *source* cell
e) The second pair (i_2, j_2), such that $(i_2, j_2) \neq (i_1, j_1)$, denotes the position of the *target* cell
f) Every pair (i_k, j_k), such that $k \geq 3$, $(i_k, j_k) \neq (i_1, j_1)$, and $(i_k, j_k) \neq (i_2, j_2)$, denotes an *unavailable* cell (obstacle)

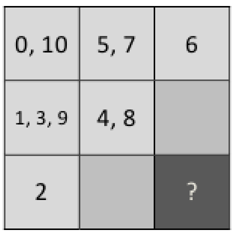
Output

The output consists of N lines containing the path produced at each test case. The path is specified by means of a space-separated list of m pairs of integers (i, j), such that:

a) $1 \leq i, j \leq n$
b) The first pair (i_1, j_1) denotes the position of the *source* cell
c) Every pair (i_k, j_k), where $1 < k < m$, denotes the position of a cell employed in the path, such that (i_k, j_k) is not an *unavailable* cell and is horizontally or vertically adjacent to (i_{k-1}, j_{k-1})
d) The last pair (i_m, j_m) denotes either
    a. The position of the *target* cell, if it was successfully reached
    b. The position of the *source* cell, otherwise

Note: Images from test cases 2 and 3





Sample Input

```
3
4 (2,2) (1,1) (3,3) (2,4)
5 (1,1) (5,5) (1,2) (2,2) (3,2) (4,2) (4,4) (5,4)
3 (1,1) (3,3) (3,2) (2,3)
```

Sample Output

```
(2,2) (2,3) (1,3) (1,4) (1,3) (1,2) (1,1)
(1,1) (2,1) (3,1) (4,1) (5,1) (5,2) (5,3) (4,3) (3,3) (3,4) (3,5) (4,5) (5,5)
(1,1) (1,2) (1,3) (1,2) (2,2) (2,1) (3,1) (2,1) (2,2) (1,2) (1,1)
```