Alan's father has bought him a box of toy soldiers, k red soldiers, k green soldiers, and *one* gold soldier. There is a $m \times n$ board, each square has a height $h_{i,j}$, and all squares are big enough to hold all the soldiers. At first, each soldier is placed in a square (not neccesarily all different), and t goal squares are chosen, each assigned with an 'importance value' r_i. The goal is to move soldiers on these t squares so that the i-th square has exactly r_i soldier(s) on it. Each soldier has to be on one of the t squares, so it's guaranteed that the sum of all r_i is equal to $2k + 1$.

Each time, a soldier can move to an adjacent (north, south, east or west) square, but it can't move outside the board. Red soldiers can only *climb up*, so it can move only if the target square is not lower than the current square; Green soldiers can only *jump down*, so it can move only if the target square is not higher than the current square; The gold soldier can move freely on the board.

Since it may be impossible to achieve the goal, Alan is allowed to use a kind of magic. Every time he uses the magic, he can do a *permutation* of all the soldiers (that is, he can do an arbitary number of exchanges, but he cannot move any soldier).

Help Alan to use least possible number of magic to achieve the goal.

Input

The first line of the input contains the number of test cases t ($1 \le t \le 10$). Each test case begins with a line containing 4 integers m, n, k, t ($2 \le m, n \le 100, 1 \le k \le 50, 1 \le t \le 2k + 1$). The second line contains $2k + 1$ pairs (x_i, y_i) indicating the initial positions of the soldiers. The first k pairs describe red soldiers, the following k describe the green soldiers, and the last one describe the gold soldier. The third line contains t triples (x_i, y_i, r_i) indicating the positions and importance value of the goal squares. The following m lines each contains n integers, indicating the heights of squares. The i-th integer of the j-th line is the height of square (x_i, y_j). Heights are integers between 0 and 100.

Output

For each test case, print on a single line the least number of magic needed.

Sample Input

```
3
4 6 2 5
1 1 1 5 4 1 4 5 3 3
1 2 1 2 6 1 3 2 1 3 6 1 4 3 1
3 2 6 1 3 5
2 1 7 4 4 6
2 3 1 4 3 4
4 3 4 3 2 3
4 3 3 7
1 1 1 2 1 3 4 1 4 2 4 3 1 1
1 1 1 2 1 1 2 2 1 2 3 1 3 1 1 3 2 1 3 3 1
1 1 1
2 2 2
3 3 3
4 4 4
8 11 3 7
1 1 1 5 1 9 8 1 8 5 8 9 4 5
1 3 1 1 7 1 1 11 1 4 5 1 8 3 1 8 7 1 8 11 1
9 2 3 1 9 2 3 1 9 2 3
1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 9 9 9 9 9
1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 9 9 9 9 9
1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 9 9 9 9 9
1 8 7 9 1 8 7 9 1 8 7
```

Sample Output

```
1
0
2
```