There is an $n * n$ matrix. Each number can be increased or decreased. If we increase a number by $x$ (which may be non-integer), the cost is $c * x$. If we decrease a number by $x$, the cost is $d * x$, where $c$ and $d$ are two non-negative integers.

The theoretical goal is to make every number equal to $F$, but in practice we only do $Q$ tests, each test is to specify a square, then adding all the numbers in the same row or column, if the difference between the sum and $(2n - 1)F$ is at most $e$, the test is successful.

Your task is to survive all the tests with minimal cost. It's not hard to see that the actual new matrix might be very different from the theoretical goal.

## Input

The first line contains $T$ ($T \leq 100$), the number of test cases. Each of the following lines begins with six integers $n$, $c$, $d$, $F$, $e$, $Q$ ($1 \leq n \leq 12$, $0 \leq c, d \leq 100$, $-10 \leq F \leq 10$, $0 \leq e \leq 5$, $1 \leq Q \leq n^2$). Then an $n * n$ matrix of integers followed. Each integer will have an absolute value of no greater than 10. Then $Q$ lines followed. Each line contains two integers $x$, $y$ ($1 \leq x, y \leq n$), that means we make a test on the square at row $x$, column $y$. Rows are numbered 1 to $n$ from top to bottom, columns are numbered 1 to $n$ from left to right. Each test case is terminated by a blank line.

## Output

For each test case, print the minimal cost, to five digits after the decimal point.

## Sample Input

```
2
5 12 23 2 1 2
0 0 1 0 0
1 2 3 1 1
3 1 5 3 3
0 0 1 0 0
0 0 1 0 0
2 3
3 3

2 0 1 0 0 3
1 -1
-1 -2
1 1
2 1
2 2
```

## Sample Output

```
58.00000
0.50000
```