2015 Canadian Computing Olympiad
Day 1, Problem 1
Hungry Fox

Time Limit: 2 seconds

Problem Description
It's dinner time for your pet fox! His meal consists of N crackers, with the ith cracker having a temperature of T_i degrees Celsius. He also has a large dish of water, which has a temperature of W degrees Celsius.

After taking an initial sip of water, your fox begins his meal. Every time he eats a cracker, its tastiness is equal to the absolute difference between its temperature, and the temperature of the last thing he ate or drank (be it the previous cracker he ate, or a sip of water, whichever he consumed most recently). He can drink some water whenever he wants, and can eat the crackers in any order.

Depending on the order in which your fox eats and drinks, the total tastiness of the crackers consumed may vary. What are the minimum and maximum values it can have?

Input Specification
The first line contains two integers, N ($1 \leq N \leq 100\,000$) and W ($0 \leq W \leq 10^9$), representing the number of crackers and the water's temperature. On the next N lines, there is one integer, T_i ($0 \leq T_i \leq 10^9$ for $1 \leq i \leq N$), representing the temperature of the ith cracker.

For at least 30% of the marks for this problem, $W = 0$.

Output Specification
The output is one line containing two integers: the minimum and maximum total tastiness your fox can experience during his meal, respectively.

Sample Input
```
3 20
18
25
18
```

Output for Sample Input
```
7 16
```

Explanation of Output for Sample Input
To minimize the total tastiness, the fox might drink water, eat the first cracker, eat the third cracker, drink more water, and finally eat the second cracker. He will then experience temperatures of 20, 18, 18, 20, and 25 degrees Celsius, and the crackers will have tastiness values of $2 + 0 + 5 = 7$.

To maximize the total tastiness, the fox might drink water, and then eat the crackers in order. He will then experience temperatures of 20, 18, 25, and 18 degrees Celsius, and the crackers will have tastiness values of $2 + 7 + 7 = 16$.

2015 Canadian Computing Olympiad
Day 1, Problem 2
Artskjid

Time Limit: 2 seconds

Problem Description

There are many well-known algorithms for finding the shortest route from one location to another. People have GPS devices in their cars and in their phones to show them the fastest way to get where they want to go. While on vacation, however, Troy likes to travel slowly. He would like to take the *longest* route to his destination so that he can visit many new and interesting places along the way.

As such, a valid route consists of a sequence of distinct cities $c_1, c_2, ..., c_k$, such that there is a road from c_i to c_{i+1} for each $1 \leq i < k$.

He does not want to visit any city more than once. Can you help him find the longest route?

Input Specification

The first line of input contains two integers n, m, the total number of cities and the number of roads connecting the cities ($2 \leq n \leq 18; 1 \leq m \leq n^2 - n$). There is at most one road from any given city to any other given city. Cities are numbered from 0 to $n-1$, with 0 being Troy's starting city, and $n-1$ being his destination.

The next m lines of input each contain three integers s, d, l. Each triple indicates that there is a one way road from city s to city d of length l km ($0 \leq s \leq n-1; 0 \leq d \leq n-1$; $s \neq d$; $1 \leq l \leq 10000$). Each road is one-way: it can only be taken from s to d, not vice versa. There is always at least one route from city 0 to city $n-1$.

For at least 30% of the marks for this problem, $n \leq 8$.

Output Specification

Output a single integer, the length of the longest route that starts in city 0, ends in city $n-1$, and does not visit any city more than once. The length is the sum of the lengths of the roads taken along the route.

Sample Input

```
3 3
0 2 5
0 1 4
1 2 3
```

Output for Sample Input

```
7
```

Explanation of Output for Sample Input

The shortest route would be to take the road directly from 0 to 2, of length 5 km. The route going from 0 to 1 to 2 is $4 + 3 = 7$ km, which is longer.

2015 Canadian Computing Olympiad
Day 1, Problem 3
Solar Flight

Time Limit: 15 seconds

Problem Description
A new era of aviation is upon us - the first solar-powered jumbo jets are about to be made available for public travel! However, this cutting-edge technology raises some safety concerns, as the rays of sunlight which power these planes can be blocked by other objects in the sky. As such, some statistics must first be calculated concerning the planned initial flights.

We consider a set of N flight paths, all travelling East from one city to another. A plane can be considered as a single point. The sky through which they pass can be modelled as a Cartesian plane, with x-coordinates representing distance East from an arbitrary fixed point, and y-coordinates representing altitude. We are interested only in the section of the sky with x-coordinates in the inclusive range $[0,\ X]$ ($1 \leq X \leq 10^9$), in which all flight paths are straight lines. The ith plane flies from (0, A_i) to (X, B_i) ($1 \leq A_i, B_i \leq 10^9$). All A values are distinct, as are all B values. The planes travel at unknown, possibly non-constant speeds along their flight paths, so at any point in time, a plane may be anywhere along its path. However, it is known that the planes will never crash with one another, so if two flight paths cross one another, both planes won't reach the intersection point at precisely the same time.

Planes also have an inteference factor: each plane i has an interference factor C_i ($1 \leq C_i \leq 10^9$), which is a measure of how much plane i would negatively impact the solar absorbing capability of any plane below them.

The solar panels on each plane are rather strange, in that they can only collect energy from directly above the plane. This means the sunlight that a given plane can absorb can be obstructed by other planes which occupy the same x-coordinate as it, and have a larger y-coordinate than it. In particular, their effectiveness is reduced based on the sum of the interference factor of all such planes.

Given this information, as well as a fixed distance constant K ($1 \leq K \leq X$), you must answer Q queries regarding the possible impact on planes' solar panels at various times. The ith query asks for the largest possible amount by which the plane P_i's solar panels could be obstructed at a single moment in time, at any point while the plane's x-coordinate is in the inclusive range $[S_i,\ S_i + K]$ ($0 \leq S_i \leq X - K$).

Input Specification
The first line contains four space-separated integers: X ($1 \leq X \leq 10^9$), the maximum x-coordinate to consider; K ($1 \leq K \leq X$), the fixed distance constant; N ($1 \leq N \leq$ 2000), the number of flight paths; Q ($1 \leq Q \leq$ 800 000), the number of queries.

Each of the next N lines contain three integers, A_i, B_i, and C_i, for $i = 1..N$ ($1 \leq A_i, B_i, C_i \leq 10^9$), representing, for plane i, the starting y-coordinate, the ending y-coordinate, and the interference factor (respectively).

Each of the next Q lines contain two integers, P_i and S_i, for $i = 1..Q$ representing the query concerning plane P_i while its x-coordinate is in the range $[S_i, S_i + K]$.

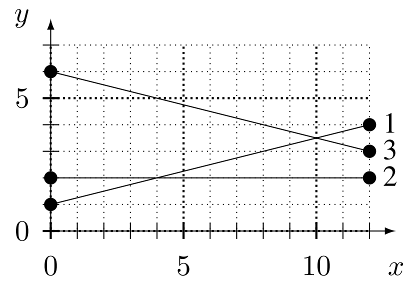
For 40% of the marks for this problem, $Q \leq 1000$.

Output Specification
The output consists of Q lines, each with the integer answer to the ith query, for $i = 1..Q$.

Sample Input
```
12 4 3 3
1 4 5
2 2 3
6 3 6
2 1
1 8
3 0
```

Output for Sample Input
```
11
6
0
```

Explanation of Output for Sample Input
Below is a diagram of the planes' flight paths:



The first query is for plane 2 over x-coordinates in the inclusive range $[1, 5]$. While this plane is at an x-coordinate smaller than or equal to 4, it might be covered by plane 3, but definitely not by plane 1. However, when it's at an x-coordinate larger than 4, it might be covered by both other planes. Therefore, the answer to this query is the sum of the other planes' interference factors, $5 + 6 = 11$.

For the second query, plane 1 could be covered by plane 3 while it has x-coordinate less than 10, and will not be covered by anything while it has x-coordinate greater than or equal to 10. Thus, it is only possibly interfered with by plane 3 with interference factor 6.

Neither plane 1 nor plane 2 can possibly be directly above plane 3 until it reaches x-coordinate 10, so the answer to the final query is 0.

2015 Canadian Computing Olympiad
Day 2, Problem 1
Cars on Ice

Time Limit: 5 seconds

Problem Description

Guarding a bank during Christmas night can get very boring. That's why Barry decided to go skating around the parking lot for a bit. However the parking lot isn't empty as the other security guards have their cars parked there. So Barry decides to push their cars out of the parking lot. He notices that their cars are parked facing either North, South, East or West. Since the parking lot is frozen, pushing a car will make it slide until it has left the parking lot or hit another car. Cars can only be pushed in the direction which they are facing. Not wanting to crash the cars, Barry enlists your help to find out what order he has to push the cars so as to clear the parking lot.

Input Specification

The first line contains two integers N and M ($1 \leq N, M \leq 2000$) representing the number of rows and columns of the parking lot. The next N lines each contain M characters representing the parking lot itself, where '.' represents an empty spot, while 'N', 'S', 'E' and 'W' each represent a car (facing North, South, East or West, respectively).

For at least 70% of the marks for this problem, $N \leq 100$ and $M \leq 100$.

Output Specification

Output the order in which the cars have to be pushed so as to clear the parking lot without crashes. Output each car in the form (r, c), where r and c are the car's coordinates on the parking lot (where $(0, 0)$ is the top leftmost spot and $(N-1, M-1)$ is the bottom rightmost spot).

You can assume there will always be at least one valid solution.

If there are multiple possible solutions, output any valid solution.

Sample Input 1

```
5 5
.....
.E.S.
.....
.....
.N.E.
```

Output for Sample Input 1

```
(4,3)
(1,3)
```

```
(1,1)
(4,1)
```

Explanation for Output for Sample Input 1
The only car that isn't initially blocked by another car is the one at $(4, 3)$. After that's pushed out to the right side of the parking lot, then the car above it (at $(1, 3)$) can be pushed, and then the one at $(1, 1)$, and finally the car at $(4, 1)$, clearing the parking lot.

Sample Input 2
```
4 3
...
.N.
.S.
...
```

Output for Sample Input 2
```
(1,1)
(2,1)
```

Explanation for Output for Sample Input 2
Either car could be pushed first to clear the parking lot, so this output is acceptable (as would the output with the lines outputted in reverse order).

2015 Canadian Computing Olympiad
Day 2, Problem 2
Timpanist

Time Limit: 1 second

Problem Description
Computer scientists don't often help percussionists, but today, that will change. Since we cannot help all percussionists at the same time, we focus on timpanists first. By way of terminology, the *timpani* is the plural of *timpano* and the player of the timpani is a *timpanist*.

A timpano is a large drum which can be tuned to a certain pitch, and a timpanist uses an ordered set of D timpani. On this occasion, they're playing a piece which has N notes. Note i occurs T_i seconds into the piece, and has pitch P_i. P_i is one of the following twelve notes:

{ F, F#, G, G#, A, A#, B, C, C#, D, D#, E }

At a given time, a timpano can only be used to play the pitch it is currently tuned to, and thus the timpanist can play a note i if and only if one of the timpani is tuned to pitch P_i at time T_i.

Every note in this piece is in the range of a single octave, from F up to E, which means that the above list of possible notes is in ascending order of pitch. In order to make your computation slightly easier, we will use integers from 1 to 12 to indicate these 12 tones:

1	2	3	4	5	6	7	8	9	10	11	12
F	F#	G	G#	A	A#	B	C	C#	D	D#	E

(i.e., F will be represented by 1, F# by 2, . . ., E by 12).

These are the only pitches to which timpani can be tuned.

Before the piece starts, the timpanist can freely tune each timpano to any pitch they'd like. However, during the piece, they may need to quickly retune them in between notes in order to be able to play the required pitches at the correct times. The drums are numbered from 1 to D. At every point in time, the drum $i + 1$ must be kept tuned to a pitch higher than drum i. Retuning a single drum must be done in an uninterrupted interval of time, in which no notes are being played and no other drums are being retuned. Because this is not an easy process, the timpanist would like to give themselves as much time as possible. In particular, they'd like to maximize the amount of time they have for the fastest retuning they must perform within the piece.

Input Specification
The first line contains two integers, N and D, the number of notes to be played and the number

of drums available to be played ($1 \leq N \leq 100$; $1 \leq D \leq 4$). The next N lines each contain two integers T_i and P_i representing the time and pitch of the ith note played ($0 \leq T_1 < T_2 < ... < T_{N-1} < T_N \leq 10^9$; $1 \leq P_i \leq 12$ for $1 \leq i \leq N$).

For 60% of the marks for this problem, $N \leq 50$ and $D \leq 3$.

Output Specification
The output is one line containing one real number (rounded off to 2 decimal places), which is the maximum amount of time (in seconds) that the timpanist can have for their fastest retuning, or `0.00` if no retunings are necessary

Sample Input 1
```
7 1
100 1
120 3
130 5
140 6
150 8
165 10
170 12
```

Output for Sample Input 1
```
5.00
```

Explanation of Output for Sample Input 1
With just 1 drum, the timpanist must retune it after every note in order to play the following one.

With 2 drums, the answer would instead be 10.00 (achieved by leaving the higher drum tuned to pitch 12).

Sample Input 2
```
12 4
0 1
2 1
3 3
6 1
9 6
12 5
21 1
23 1
24 3
27 1
30 8
```

```
33 6
```

Output for Sample Input 2
```
4.50
```

Explanation of Output for Sample Input 2
The first 6 notes include only the 4 pitches 1, 3, 5, and 6. Similarly, the last 6 include only 1, 3, 6, 8.

The single optimal strategy involves pre-tuning the 4 drums to 1, 3, 5, and 6. After the sixth note, the timpanist can take 4.5 seconds to retune the highest drum to an 8, and then another 4.5 seconds to retune the second-highest drum to a 6, finishing just in time to play the seventh note.

Sample Input 3
```
2 4
40287 8
20338194 8
```

Output for Sample Input 3
```
0.00
```

Explanation of Output for Sample Input 3
This is a more typical timpani part, involving just one note, and thus no retuning.

2015 Canadian Computing Olympiad
Day 2, Problem 3
Eggscavation

Time Limit: 10 seconds

Problem Description
It's time for a vacation! You are sick and tired of C shells, so you decide to become a seashell collector.

For your vacation, you have decided to visit the beautiful island nation of Cartesia. It is well-known for having a lovely square beach that is composed of an $N \times N$ grid of square cells. You have brought your trusty shovel, which is able to dig up a $K \times K$ square subgrid of the beach. Your shovel, being trusty, can only dig up a $K \times K$ subgrid that is entirely within the beach.

There are M undiscovered species of shells hidden under certain grid cells. Specifically for each i, there are S_i shells from the ith species in S_i grid locations, with $1 \leq S_i \leq 4$. For each distinct species that you dig up and bring back home, you can sell it to a scientist back home for one dollar. Multiple shells of the same species don't add extra value.

Complicating matters is a glorious dodo bird running around the beach. At a given moment, it may decide to bury an egg in a grid cell (including grid cells that have eggs or shells already). The bad news is that if the $K \times K$ subgrid dug up by your shovel includes a dodo egg, the scientists will become annoyed that you are harming endangered species, and nobody will pay you anything.

After all is said and done, you decide to sit back, go back to programming, and simulate the digging instead. You will be computing the probability that your scoop, when chosen uniformly from all valid possible scoops, will make at least a given minimum profit (to pay off your student loans) at different points in time. Who wants to get all sweaty from shoveling on a beach anyway?

Input Specification
The first line of input contains two integers N and K, ($1 \leq N \leq 2500$; $1 \leq K \leq N$), the size of the beach and the size of the shovel, respectively.

The second line of input contains the integer M ($0 \leq M \leq 10^5$), the number of species of shells. The next M lines each represent species i and consist of the integer S_i ($1 \leq S_i \leq 4$) followed by $2 * S_i$ more integers, which represent the grid positions (between $(1, 1)$ and (N, N)) where the S_i shells of that species are buried.

The next line contains T ($1 \leq T \leq 10\,000$). Each of the next T lines represent one specific point in time (from oldest to newest) and each line has one of the following two forms:

- `1` A_i B_i ($1 \leq A_i, B_i \leq N$), meaning that the dodo just buried an egg in the grid cell (A_i, B_i); or

- $2\ V_i$ ($1 \leq V_i \leq 10^9$), meaning we would like to calculate the probability that a random dig at this point in time has profit in dollars $\geq V_i$. Note that no shells or eggs are actually removed or added when this probability is calculated.

For at least 15% of the marks for this question, $N \leq 40$ and all update operations occur before all query operations.

For an additional 25% of the marks for this question, all update operations will occur before all query operations.

Output Specification
For each query operation, output on one line the probability that a random scoop would contain at least the desired number of different types of shells.

Your answer must be within 10^{-4} of the judge's answer.

Sample Input
```
4 2
3
3 1 1 2 3 4 2
3 1 4 2 3 3 2
4 2 1 2 4 4 1 4 4
6
2 2
2 3
1 2 3
2 2
2 3
2 4
```

Output for Sample Input
```
0.88889
0.33333
0.44444
0.11111
0.00000
```

Explanation of Output for Sample Input
Initially, we have the following arrangement of shells (with the first shells in the input being labelled as 1, and so on):

1			2
3		1, 2	3
	2		
3	1		3

and our shovel will scoop up a 2×2 square of cells. Thus, there are 9 possible scoops.

With no eggs present, 8 of the 9 scoops contain at least two species of shells and 3 of the 9 scoops contain three species of shells.

An egg is then dropped into the cell that contains $1, 2$.

Then, 4 of the 9 scoops contain at least two species of shells and no eggs and only 1 scoop (the bottom-leftmost scoop) would contain all three species of shells and no eggs. The final output indicates there are no scoops which contain 4 different species of shells.