French-Australian Regional Informatics Olympiad

Sunday 16th May

2004

Duration: 4 hours

3 Questions

All questions should be attempted

Problem 1
Stargazing

Time Limit: 1 second

You decide to benefit from the good weather by making a campsite with your friends. After a slow day of fishing, followed by a good meal around a wood fire during which you exhaust your entire repertoire of songs, you each lie back, relax and stare at the stars. Unfortunately you know very little about the constellations that you see.

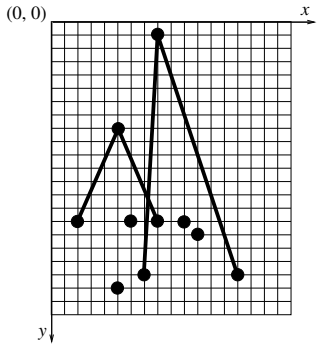
For you, the stars do not form splendid creatures or characters, or even so much as a saucepan. You always found it absurd to memorise the names and positions of the different constellations — after all, by staring hard enough at several dots in space you can make them look like anything really. So you decide to seek out your own shapes in the stars.

To spice things up a little, you propose a small contest with your friends: the goal is to find the largest possible arrow in the night sky. There are no limits on what you can use, so while your friends are gazing upwards you secretly pull out your digital camera and laptop to guarantee victory.

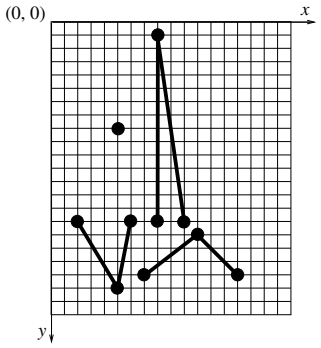
You must write a program to find the largest possible arrow formed from precisely three stars. Your arrow must be pointing upwards[1], and the two stars at the left and right sides of the arrow must be at precisely the same height. The star that forms the tip of the arrow must have its x-coordinate strictly between the other two, and its y-coordinate must be strictly above them.

Furthermore, the width of the arrow must be smaller or equal to its height. Note that the width of the arrow is defined to be the difference in x-coordinates between the leftmost star and the rightmost star, and the height of the arrow is defined to be the difference in y-coordinates between the tip of the arrow and the two stars beneath it.

Thus, if the three stars making up the arrow are at coordinates (x_1, y_1), (x_2, y_2) and (x_3, y_3), we must have $x_1 < x_2 < x_3$ and $y_2 < y_1 = y_3$. The width of the arrow is $x_3 - x_1$ and the height of the arrow is $y_1 - y_2$, and we must therefore also have $x_3 - x_1 \leq y_1 - y_2$. The following diagrams illustrate several correct and incorrect arrows.



Correct arrows



Incorrect arrows

The size of an arrow is determined as its width plus its height. In seeking the largest arrow, you are therefore seeking the arrow whose width plus height is as large as possible.

[1] Of course, the French contestants will be searching for arrows pointing downwards.

Input

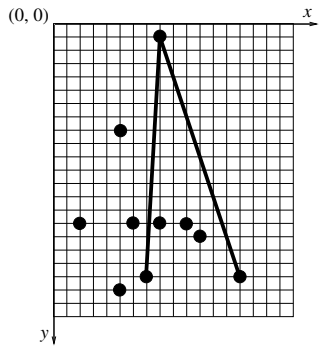
Your program should read directly from standard input. The first line of input will contain a single integer: N, the number of visible stars in the sky ($1 \leq N \leq 2\,000$).

Following this will be N lines each describing a single star. Each of these lines will contain two positive integers separated by a space: the x and y coordinates of the star ($0 \leq x, y \leq 1\,000\,000$). You are guaranteed that no two stars will be at the same location in the sky.

Output

Your program must write directly to standard output. Your output must consist of a single integer: the size of the largest arrow (i.e., its width plus its height). You are guaranteed that at least one correct arrow can be found.

Sample Input



The following input represents the night sky illustrated above, where the largest arrow is marked in the diagram. The three stars forming this arrow are at coordinates $(7, 19)$, $(8, 1)$ and $(14, 19)$. This arrow has width $14 - 7 = 7$ and height $19 - 1 = 18$, giving a total size of $7 + 18 = 25$.

```
10
6 15
14 19
2 15
5 8
8 1
8 15
11 16
7 19
10 15
5 20
```

Sample Output

```
25
```

Problem 2
Blue

Time Limit: 5 seconds

As an Arts student at heart, one of your favourite pastimes is strolling around the local art gallery and contemplating with awe the creations of the imagination. One day you imagine how nice it would be to tile the entrance of your mansion to look like one of these artworks, and so you pull out your digital camera and photograph one of your favourite paintings. Unfortunately, being the arty person you are, you accidentally left your blue filter on your camera, and so your photo has turned out to be coloured entirely in various shades of blue.

This doesn't faze you though, as you're quite comfortable with the idea of a blue-tiled entrance. After taking a look around your local tile shop, you find a number of blue-shaded tiles that you can use. All of the tiles are square, with sizes 1m × 1m, 2m × 2m, 3m × 3m or 4m × 4m. Each tile is coloured in a single shade of blue, which you can measure using an integer ranging from 0 (very dark blue) to 255 (very light blue). Each pixel of your photograph can also be measured from 0 to 255 on this same scale.

Your task is to lay out a collection of these tiles in the entrance to your mansion, where a 1m × 1m portion of tile corresponds to a single pixel of your photograph. The *error* for each pixel is the absolute difference between the shade of the pixel and the shade of the corresponding piece of tile. The *total error* for the entire picture is the sum of the individual errors for every pixel. You must make your entrance resemble the photograph as closely as possible, i.e., you must make the total error as small as possible.

Tiles may not overlap each other, and they may not extend beyond the boundary of the picture. You may not leave any holes (i.e., pixels of your photograph for which there is no corresponding piece of tile). Although the shop only offers a few different types of tile, you may purchase as many of each type of tile as you like.

Input

Your program must read from standard input. The first line will be a single integer n ($1 \leq n \leq 20$), denoting the number of different types of tiles that the shop sells. These types are numbered $1, 2, \ldots, n$.

Following this will be n lines each describing a single type of tile that the shop sells. Each of these lines will have the form $s\ k$, where s represents the side length (in metres) of the square tile and k is an integer describing the shade of blue that the tile is coloured. You are guaranteed that $1 \leq s \leq 4$ and that $0 \leq k \leq 255$. There will always be at least one tile of dimension 1m × 1m in the input.

After this will be a line consisting of two integers $h\ w$, where h and w are the height and width of your photograph measured in pixels ($1 \leq h, w \leq 200$). Following this will be h lines of w integers each (separated by spaces), describing the shade of each pixel of your photograph. Each of these integers will be between 0 and 255 inclusive.

Output

Your program must write the best solution it can find to standard output. Your output should contain one line for each tile that you place. Each of these lines should have the form $r\ c\ t$, where r and c are the row and column of the top-left corner of the tile ($1 \leq r \leq h$ and $1 \leq c \leq w$), and t is an integer describing which type of tile you are using ($1 \leq t \leq n$). You may output these tiles in any order you like.

After the tiles have been written, your program must write one additional line of output. This line must contain a single integer representing the total error for your solution.

Sample Input

```
3
1 10
2 15
1 20
3 4
16 15 10 25
14 15 14 30
10 10 30 11
```

Sample Output

```
1 1 2
3 1 1
3 2 1
1 3 1
1 4 3
2 3 2
42
```

The following three diagrams demonstrate the sample data above. The diagram on the left is the photograph you are required to replicate with as little error as possible. The central diagram is one possible tiling of this, using only the tiles given in the sample input. The diagram on the right shows the resulting error for each pixel of the photograph.

16	15	10	25
14	15	14	30
10	10	30	11

15	15	10	20
15	15	15	15
10	10	15	15

1	0	0	5
1	0	1	15
0	0	15	4

The sum of all errors for each pixel is $1+0+0+5+1+0+1+15+0+0+15+4 = 42$, hence the total error for this tiling is 42.

Scoring

There is no particular "best solution" that you are required to achieve. Instead, your score will be determined relative to the other contestants whom you are competing against (as well as the judges' solution). For each test case, the contestant who achieves the least total error is identified. Your score for this test case will then be:

- 100% if your program finds a solution with this same least total error;
- 10% if your program simply matches each pixel of the photograph with the 1m × 1m tile closest to that pixel's shade (ignoring all tiles of size 2m × 2m or above);
- 0% if your program generates an incorrect solution;
- otherwise determined by a linear scale according to your total error, with the 100% and 10% marks on the scale corresponding to the solutions described above. If your output contains a correct tiling, no matter how large your total error is, you will be guaranteed at least 5%.

For example, for the above sample data, the total error achieved by matching each pixel with the closest 1m × 1m tile is 48. If the best solution found by any contestant (or the judges) has a total error of 32, then the scoring scale for a *correct* solution would be as follows:

Total error	32	34	36	38	40	42	44	46	48	50	52	54
Score	100%	89%	78%	66%	55%	44%	33%	21%	10%	5%	5%	5%

Problem 3
Chariot Race

Time Limit: 1 second

It's been a slow century up on Olympus, and the gods are bored. No wars to influence, no affairs to meddle in — the humans are actually getting on rather well with each other. So you decide to have a chariot race.

Since you are gods, these are of course no ordinary chariots. You are racing through the heavens from star to star. Your aim is to be the first to reach the Serene Olive Grove of Alpha Centauri, and Hera has promised the winner a bottle of the famous Sweet Nectar of Elysium. For a prize this magnificent, you are determined to come out victorious.

But even for the gods there are rules. You cannot fly directly to Alpha Centauri, since otherwise Zeus (who has the fastest chariot) would be the clear winner. Instead you have each been given a map of the galaxy telling you which stars you are allowed to fly between.

To add further excitement to the race, some stars are joined by wormholes. Travelling through a wormhole actually takes you back in time. Hera has given Zeus strict instructions not to use the wormholes (being the leader of the gods he must be a good sport), but the rest of you are allowed to use the wormholes as often as you like.

Assume that the race starts at time 0. If you enter a wormhole at time t minutes, then you come out the other end of the wormhole at time $t/2$ minutes. Since the gods have not yet invented fractions, if t is an odd number then you must round down. For example, if you enter a wormhole at time 10 minutes then you exit at time 5, and if you enter a wormhole at time 15 minutes then you exit at time 7.

Your task is to plot a course through the heavens that allows you to arrive at the Serene Olive Grove of Alpha Centauri at the earliest time possible. Note that you are allowed to pass through Alpha Centauri more than once, e.g., you are allowed to pass through Alpha Centauri at a later time, travel through one or more wormholes and then return at an earlier time to win the race.

Input

Your program should read directly from standard input. The first line of input will contain a single integer N representing the number of stars ($1 \leq N \leq 100$). These stars will be numbered $1, 2, \ldots, N$.

The second line of input will contain two integers S F, where S is the star at which you begin the race and F is the star at which you end the race (i.e., Alpha Centauri). You are guaranteed that $1 \leq S, F \leq N$.

The third line of input will contain a single integer P representing the number of ordinary paths that you are allowed to take from star to star. Following this will be P lines each describing a single path. Each of these lines will contain three integers A B T, meaning that you are allowed to travel from star A to star B and that the journey will take you precisely T minutes. You are guaranteed that $1 \leq A, B \leq N$, that $A \neq B$ and that $1 \leq T \leq 1000$.

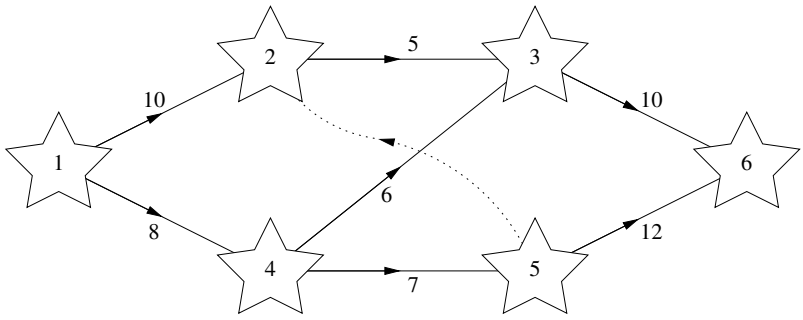
The next line of input will contain a single integer W representing the number of wormholes. Following this will be W lines each describing a single wormhole. Each of these lines will contain two integers A B, meaning that the wormhole moves you from star A to star B (and takes you back in time). You are guaranteed that $1 \leq A, B \leq N$ and that $A \neq B$.

Note that all paths and wormholes are one-way. That is, if the input includes a path or wormhole from star A to star B, you cannot use that path or wormhole to move from star B to star A (though of course, some other path or wormhole from B to A might appear elsewhere in the input). You are guaranteed that no two paths or wormholes will take you from the same star A to the same star B, and that it is actually possible for you to eventually reach Alpha Centauri.

Output

Your program must write directly to standard output. Your output should consist of a single integer describing the earliest time at which you can arrive at Alpha Centauri.

Sample Input



The following input represents the map illustrated above. Regular paths are marked by solid lines, and the wormhole is marked by a dotted line. The beginning of the race is at star 1 on the left, and the end of the race is at star 6 on the right.

```
6
1 6
7
1 2 10
1 4 8
2 3 5
3 6 10
4 3 6
4 5 7
5 6 12
1
5 2
```

Sample Output

```
22
```

Explanation

The shortest route using ordinary paths from start to finish is $1 \to 4 \to 3 \to 6$, taking a total time of $8 + 6 + 10 = 24$ minutes. However, if we make use of the wormhole then we can arrive at the finish earlier.

If we travel along the regular path $1 \to 4 \to 5$, we arrive at star 5 at time 15. Travelling down the wormhole $5 \to 2$ takes us back to time 7, and then we can finish our journey along the path $2 \to 3 \to 6$, arriving at the final star at time $7 + 5 + 10 = 22$.