French-Australian Regional Informatics Olympiad
14th March, 2008

Duration: 4 hours

4 questions

All questions should be attempted

Problem 1
Graffiti

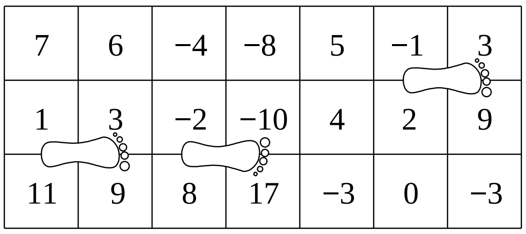
Time and Memory Limits: 1 second, 32 Mb

Along your daily walk to school, there is a long pavement tiled with concrete slabs. The pavement is three slabs wide and rather long (it always feels longer on the walk home). One day you discover that some irresponsible youth has written graffiti on all these slabs. On closer inspection, you notice that it's mathematical graffiti—each slab has been marked with an integer.

This time, curiosity does not get the better of you. Instead you decide to play a game with your friends on the way to school. The game is played as follows:

- You begin by standing just before the start of the graffitied pavement.
- Each step along the pavement must land on a corner where four concrete slabs meet. By stepping on a corner, you earn points. The number of points you earn for the step is the sum of the numbers on the four slabs you have stepped on.
- You cannot step on a corner if it would include a slab that you have already stepped on before. In other words, once you have stepped on one corner of a slab, you cannot step on any other corner of that slab.
- The total number of points you earn is added up and becomes your final score.

For example, the pavement below is three slabs wide and seven slabs long. The best possible score is 50, obtained by stepping in the corners shown.



7	6	−4	−8	5	−1	3
1	3	−2	−10	4	2	9
11	9	8	17	−3	0	−3

$$(1+3+11+9)+(-2-10+8+17)+(-1+3+2+9)=24+13+13=50$$

Note that you may leave any number of slabs between steps, and in fact if all slabs have negative points associated with them, you can simply opt to step on no slabs and retain a score of 0.

After several days of playing this game with your friends and consistently losing, you decide to whip out your trusty laptop to secure your place as the high scorer. Your task is to determine the highest score that can be achieved on a graffitied pavement, given the length of the pavement and the numbers on each slab.

Constraints

- $1 \le L \le 500\,000$, where L is the length of the pavement.

Input

Your program must read from standard input. The first line of input will contain a single integer L, the length of the pavement.

Following this will be L lines in the form $x\ y\ z$, describing the numbers written on each of the three slabs in the given row. These three numbers will be given in order from the left side of the pavement to the right (as seen by a person who is walking), and they will all be integers between $-100\,000$ and $100\,000$, inclusive.

Output

Your program must write a single line to standard output. This line must contain a single integer, giving the largest possible score you can obtain from walking along the pavement. You are guaranteed that the answer will never exceed $2\,000\,000\,000$.

Sample Input

```
7
7 1 11
6 3 9
-4 -2 8
-8 -10 17
5 4 -3
-1 2 0
3 9 -3
```

Sample Output

```
50
```

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.

Problem 2
Sledge Track

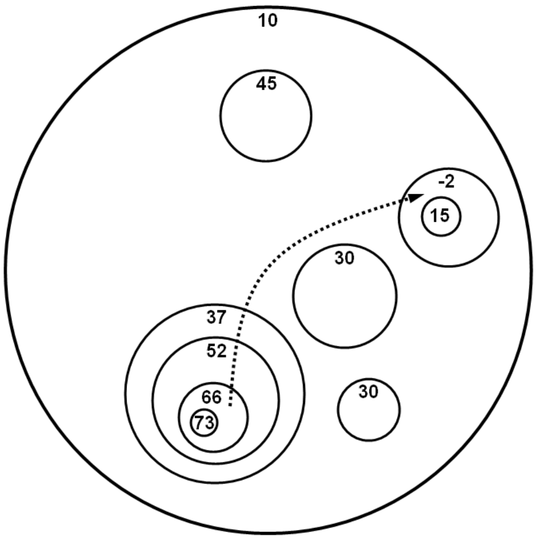
Time and Memory Limits: 1 second, 40 Mb

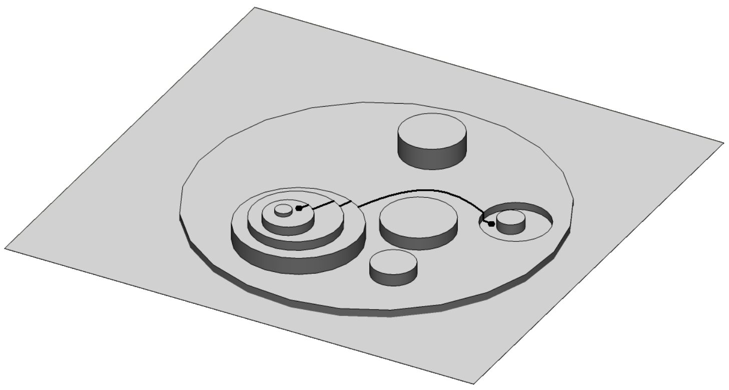
For the first time in many years, probably as a consequence of climate change, it has snowed very hard in your area. You decide to take this opportunity to organize a sledge competition in the hills behind your house. You are looking for a nice spot for a track and want to find one that would allow the sledges to reach high speeds, even though there aren't too many steep slopes in the area.

You have found a topographic map of the area, which is drawn using *contours*. These contours are perfect circles, and the map indicates the altitude of the area touching the inner edge of each circle. The area that is outside all of the circles has an altitude of zero. As these contours are the only information you have about the shape of the hills, you may assume the areas between contours are flat. Two contours never intersect or even touch one another.

Using your map, your goal is to find a track that crosses no more than K circles for some given number K (you don't want the track to be too bumpy), and such that the total difference in altitude between the starting point and the finish point is as large as possible. Your track can have uphill parts, but should never reach an altitude higher than your starting point.

For example, the diagram below shows one such map. If the maximum number of circles you are allowed to cross is $K = 4$, then the track with the largest difference in altitude is shown by the dotted line, which has an altitude difference of $66 - (-2) = 68$. The second diagram gives a clearer view of the landscape described by this map (not to scale).





Constraints

For 95% of the available marks:

- $1 \leq C \leq 2000$, where C is the number of circles on the map;
- $1 \leq K \leq 200$, where K is the maximum number of circles your track can cross;
- $-1000 \leq X, Y \leq 1000$, where X and Y form the co-ordinates of the center of a circle;
- $1 \leq R \leq 2000$, where R is the radius of a circle;
- $-1000 \leq A \leq 1000$, where A is the altitude of an area.

For 30% of these marks (which form part of the 95% mentioned above):

- $1 \leq C \leq 100$, where C is the number of circles on the map;

Furthermore, for the remaining 5% of the available marks:

- $1 \leq C \leq 40\,000$, where C is the number of circles on the map.

Input

Your program must read from standard input. The first line of input will contain the integers C K, where C is the number of circles on the map, and K is the maximum number of circles your track can cross.

The following C lines describe the circles, given in increasing order of their radius. Each of these lines describes a circle using four space-separated integers: X_i, Y_i, R_i and A_i, where (X_i,Y_i) are the coordinates of the center of the circle, R_i is the radius and A_i is the altitude of the area touching the inner edge of the circle.

Output

Your program must write a single line to standard output. This line must contain a single integer, giving the maximum possible difference in altitude you can obtain between the starting point and end point of a track with the constraints described above.

Sample Input

```
10 4
38 61 2 73
69 34 3 15
61 59 4 30
40 60 5 66
58 44 6 30
71 34 6 -2
47 21 6 45
41 58 8 52
41 57 11 37
48 40 33 10
```

Sample Output

```
68
```

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.

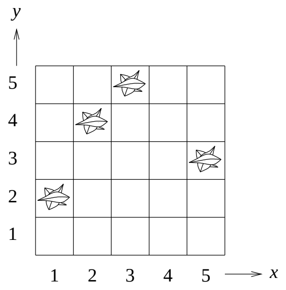
Problem 3
Leaf Blower

Input File: *leaf*.k.*in*
Output File: *leaf*.k.*out*

Output Only Task

Autumn has arrived, and the leaves are falling from the trees onto your tiled courtyard. Although the leaves are pretty, your courtyard is prettier, and so you decide to clean the leaves away.

Your courtyard is formed from a grid of tiles, as illustrated below; these tiles can be described by (x, y) coordinates. The leaves have fallen into n piles, each on a different tile.

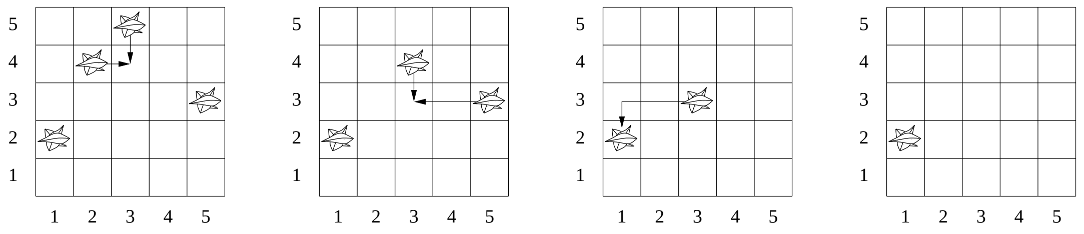


You have a leaf blower that allows you to move these piles around. The leaf blower can move a pile of leaves horizontally or vertically from one tile to another. If any two piles meet, they combine to form a single (larger) pile, which you can continue to blow around as before.

You wish to combine the leaves into a single pile using as few *pile movements* as you can. One pile movement involves blowing a pile of leaves (of any size) to an adjacent tile.

As an example, consider the figure below. The leaves begin in four piles, located in tiles $(1, 2)$, $(2, 4)$, $(3, 5)$ and $(5, 3)$, as seen in the first diagram.

- To begin, you could blow the two piles at $(2, 4)$ and $(3, 5)$ into the tile $(3, 4)$. This requires two pile movements. The result is that both piles join into a single pile, as shown in the second diagram.
- You could then blow the new pile at $(3, 4)$ down to $(3, 3)$, requiring one pile movement, and also blow the far right pile at $(5, 3)$ across to $(3, 3)$, requiring two more pile movements. As a result both piles merge into a single pile at $(3, 3)$, as seen in the third diagram.
- Finally, you could blow the new pile at $(3, 3)$ two tiles left and one tile down, so that it joins the pile at $(1, 2)$. This requires another three movements. At this point all of the leaves have been gathered into a single pile, as seen in the fourth diagram, and so we are done.



By following these steps, a total of $2 + 3 + 3 = 8$ pile movements are used, which is the smallest number possible.

You are not required to find the best possible solution. Instead you must simply use as few pile movements as you can. Your solution will be compared against other solutions, and better solutions will score more points. See the *Scoring* section for details.

Input

You are given ten input files `leaf.`*k*`.in` ($1 \leq k \leq 10$).

The first line of each file contains the single integer n, giving the number of piles ($2 \leq n \leq 500$). Following this are n lines, each of the form $x\ y$, giving the coordinates of a single pile of leaves. All x and y coordinates are integers in the range $1 \leq x, y \leq 1\,000$.

Output

For each input file `leaf.`*k*`.in`, you must produce an output file `leaf.`*k*`.out` that contains your solution.

Each solution file should contain one line for each pile movement, given in order from the first movement to the last. Each line should be of the form $x\ y\ p\ q$, which indicates that the pile at tile (x, y) is blown to the tile (p, q). These two tiles must be either horizontally or vertically adjacent, and all tile coordinates must remain in the range $1 \leq x, y, p, q \leq 1\,000$.

Sample Input

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

Sample Output

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

Scoring

There is no "best solution" that you are required to achieve. Instead, your score for each input scenario will be determined relative to the other contestants whom you are competing against (as well as one of the judges' solutions).

For each input scenario, the contestant who finds a valid solution using the fewest pile movements will be identified. Suppose that this contestant uses p pile movements in total. Assuming your output also contains a valid solution, it will be scored as follows:

- If your solution uses between p and $1.1 \times p$ pile movements, it will be scored according to a linear scale between 100% and 50%, where 100% corresponds to p movements and 50% corresponds to $1.1 \times p$ movements.
- If your solution uses between $1.1 \times p$ and $2 \times p$ pile movements, it will be scored according to a linear scale between 50% and 10%, where 50% corresponds to $1.1 \times p$ movements and 10% corresponds to $2 \times p$ movements.
- If your solution uses more than $2 \times p$ pile movements, it will be scored precisely 10%.

For example, consider an input scenario for which the best solution found by any contestant uses 100 pile movements. The scoring for a valid solution would be as follows:

Movements	100	102	104	106	108	110	140	170	200	300	900
Score	100%	90%	80%	70%	60%	50%	37%	23%	10%	10%	10%

A solution that makes an invalid movement (for instance, moving to a non-adjacent square, or moving outside the allowed coordinate range of 1..1000) will score zero. Likewise, a solution that does not combine all of the leaves into a single pile will score zero.

If a solution makes an otherwise legal movement from an empty tile (i.e., there are no leaves to blow from that tile), this irrelevant movement will be silently ignored (but will add to your total number of pile movements).

Submitting for an Output Only Task

To submit your output files, you must do the following:

- Create a zip file containing all of your output files. You may include as many or as few output files as you like (i.e., you do not need a solution for every input scenario). On a GNU/Linux system you can use a command like the following:

  ```
  zip mysolutions.zip leaf.*.out
  ```

  On Windows systems you can create a zip file by selecting *File* → *New* → *Compressed (zipped) Folder* from within Windows Explorer, and then you can copy your output files into this new zip file.

- Submit this zip file to the FARIO website, in the same way that you would submit an ordinary solution.

The website will look inside the zip file and ensure that the output files are named correctly, although it will not check the formatting or layout of these files.

If you resubmit a different zip file, **the old zip file will be deleted**. For instance, suppose you submit a zip file with solutions to the first five scenarios, and later on you find a solution to the sixth. Your new submission must contain **all six output files**, since when you submit your new solutions the old ones will be thrown away.

Please contact the contact organisers at `fario@fario.org` if you have any questions about this procedure.

Problem 4
Rock Climbing

Time and Memory Limits: 1 second, 64 Mb

For reasons not yet understood, indoor rock climbing is becoming a popular pastime amongst many ex-philatelists, philanthropists and philosphers. You too have decided to follow suit and begin your career aspirations as the world's best rock climber. Indoor rock climbing involves getting from the bottom of a wall to the top using only your strength to pull yourself up, and trying your utmost not to fall off. Near the top of the wall is a bell that you ring to boast your feat to the world.

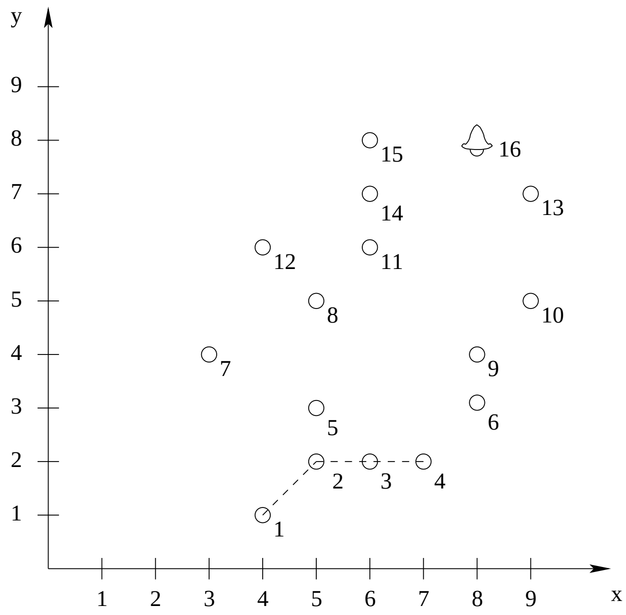
The wall is a flat two-dimensional surface with handles at various points. To hang onto the wall, you must have a hold on three distinct handles when you are stationary, or two distinct handles when you are moving—any fewer and you will fall off and embarass yourself!

To move yourself around and up the wall, you can let go of one of these three handles and then take hold of a different handle. The time required to perform the move is equal to the sum of the horizontal and vertical distance moved between the handle you let go of and the handle you take hold of.

As you can only reach so far, each of the three handles you are attached to must be at most R metres from some other handle you are attached to, as measured by the sum of the horizontal and vertical distances between two points. Note that they do not *all* have to be within R metres of each other. For example, the starting configuration in the diagram below (shown by the dotted line) has distance 2 between points 1 and 2, and distance 2 between points 2 and 4, but distance 4 between points 1 and 4. This is a valid configuration even if $R = 2$.

In order to be the world's fastest rock climber, you need to determine a set of moves that will get you to the bell as fast as possible. The bell is located on one of the handles (but is not necessarily the highest handle). You are told which handles to begin your climb from and you are guaranteed that these will all be within reach of each other.

Given (i) the set of handles on a wall described as (x, y) co-ordinates on a plane and (ii) your starting handles, your task is to find the smallest time in which you can get from the starting position to the bell on the wall.



For example, consider the diagram above with $R = 2$. Each circle represents a handle. There are 16 handles (including the one with the bell). Suppose we were required to start by having a hold on handles 1, 2 and 4 as shown. Then one possible way to reach the bell would be to:

- release handle 4 and hold onto handle 3 (1 second);
- release handle 1 and hold onto handle 5 (3 seconds);
- release handle 2 and hold onto handle 8 (3 seconds);
- release handle 3 and hold onto handle 11 (4 seconds);
- release handle 5 and hold onto handle 15 (6 seconds);
- release handle 8 and hold onto handle 16 (6 seconds);

This takes a total of $1 + 3 + 3 + 4 + 6 + 6 = 23$ seconds, and is in fact the fastest way to reach the bell.

Constraints

- $1 \leq N \leq 50$, where N is the number of handles on the wall;
- $1 \leq R \leq 6$, where R is how far you can reach (i.e. the maximum sum of horizontal and vertical distance between each point you are holding onto and its nearest other point you are holding onto);
- $-1000 \leq x, y \leq 1000$, where x and y form the co-ordinates of a handle.

For 50% of the marks, $N \leq 20$.

Input

Your program must read from standard input. The first line of input will contain the integers N R, separated by a space. The N handles will be numbered $1 \ldots N$. The second line will contain the three distinct integers A B C that denote the three handles you begin from $(1 \leq A, B, C, \leq N)$.

Following this will be N lines in the form x_i y_i, where the ith of these lines describes the location of the ith handle. No two handles will have the same location. The bell is always attached to the Nth handle.

Output

Your program must write a single line to standard output. This line must contain a single integer, giving the smallest possible time to get from the starting position to the bell. You are guaranteed that it will always be possible to reach the bell.

Sample Input

```
16 2
1 2 4
4 1
5 2
6 2
7 2
5 3
8 3
3 4
5 5
8 4
9 5
6 6
4 6
9 7
6 7
6 8
8 8
```

Sample Output

```
23
```

Scoring

The score for each input scenario will be 100% if the correct answer is written to the output file, and 0% otherwise.