Algorithms for Programming Contests SS20 - Week 10

Chair for Foundations of Software Reliability and Theoretical Computer Science, TU München

Mikhail Raskin, Tobias Meggendorfer, Stefan Jaax, A.R.Balasubramanian, Christoph Welzel

Welcome to our practical course! This problem set is due by

Tuesday, 30.06.2020, 6:00 a.m.

Try to solve all the problems and submit them at

`https://judge.in.tum.de/conpra/`

This week's problems are:

The following amount of points will be awarded for solving the problems.

Problem	A	B	C	D	E	F
Difficulty	easy	easy	medium	medium	hard	hard
Points	4	4	6	6	8	8*

* catchup problem / bonus points, do not count towards total

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request, include:

- the name of the problem (by selecting it in the subject field)
- a verbose description of your approach to solve the problem
- the time you submitted the solution we should judge

We will check your submission and award you half the points if there is only a minor flaw in your code.
If you have any questions please ask by using the judge’s clarification form.

Problem A
Meteorite

Today, the LASER (**L**aboratory for **A**dvanced **S**cientific **E**mission of **R**ays) made a huge announcement - they discovered a new element. They even found out how to synthesize it: At first, you need a meteorite that is rapidly accelerating towards earth. Then you heat it up with a high-powered laser using a special focussing crystal (because lasers are totally awesome). This causes the meteorite to be rapidly condensed into one very small lump of the new element - aptly named "meteoritium".

However that process still leaves a small problem - a super dense lump of meteoritium rapidly falling through the earth's atmosphere. They now issued a safety warning to all people living close to the calculated impact site. To her excitement, Lea is among them.

She is now itching to know if there is a chance that the meteoritium will land on her parents' property (you can think of the property as a simple polygon with no intersecting edges and no holes) so she can be one of the first people on earth to see the new element.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case begins with a line consisting of 3 integers x_{impact}, y_{impact}, the coordinates of the calculated impact site and n, the number of sides that her parents' property has. n lines follow, each containing 4 integers x_1, y_1, x_2, y_2, describing a side of the polygon connecting the points (x_1, y_1) and (x_2, y_2).

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is "jackpot" if the impact site is contained in the given polygon and "too bad" otherwise. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $3 \leq n \leq 1000$
- $-1000 \leq x_i, y_i \leq 1000$
- Every coordinate of the polygon will have exactly 2 incident sides.
- The given polygon will always be a single, connected shape.
- (x_{impact}, y_{impact}) will never lie on a side or corner of the polygon.

Sample Input 1

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

0 1 5
-1 -1 -1 2
1 1 1 0
1 0 -1 -1
0 0 1 1
-1 2 0 0
```

Sample Output 1

```
Case #1: jackpot
Case #2: too bad
```

Sample Input 2

```
3
-3 4 10
5 0 2 -2
1 -2 0 -5
2 -2 1 -2
-2 -1 -5 -1
0 -5 -1 -1
-5 -1 -5 0
-2 3 5 0
-5 0 -2 4
-2 4 -2 3
-1 -1 -2 -1

-2 0 10
-4 -1 -5 0
-1 1 0 4
-5 0 -5 1
-5 1 -1 1
5 0 3 -4
3 -4 2 -5
0 4 4 4
-2 -1 -4 -1
4 4 5 0
2 -5 -2 -1

3 -3 9
1 3 5 0
-5 0 -5 3
5 0 1 -4
-5 3 -3 3
0 3 1 3
-3 3 0 3
-3 -2 -5 -3
-5 -3 -5 0
1 -4 -3 -2
```

Sample Output 2

```
Case #1: too bad
Case #2: jackpot
Case #3: too bad
```

Problem B
Fence Posts

Lea has recently bought a new garden. The previous owner did not care very much for it, therefore it is quite a mess. The fencing is in an especially bad condition, fence posts are everywhere inside the garden and on unnecessary points of the boundary. Lea wants to fence in an area as big as possible using the existing fence posts and fences that run in straight lines between those posts. Help Lea by telling her which posts cannot be removed without altering the size of the garden.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with an integer n, the number of fence posts, n lines follow, The i-th line contains two integers x_i, y_i, the x- and y-coordinates of the i-th fence post.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is a space-separated, naturally ordered list of the indizes of those fence posts that cannot be removed without altering the size of the garden. Fence posts are indexed starting at 1. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $3 \leq n \leq 10000$
- $0 \leq x_i, y_i \leq 1000$
- All points will be distinct.
- There will be at least three non-collinear points.

Sample Input 1

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

4
0 0
2 2
1 1
0 2
```

Sample Output 1

```
Case #1: 1 3 4
Case #2: 1 2 4
```

Sample Input 2

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

6
6 6
4 5
2 4
9 1
7 2
9 4

4
0 4
8 0
8 1
8 5

6
5 7
1 6
8 1
4 2
2 0
6 5

8
2 2
6 7
0 1
0 4
8 3
10 6
9 6
3 1

3
0 3
9 3
9 8
```

Sample Output 2

```
Case #1: 1 2 3 4
Case #2: 1 3 4 6
Case #3: 1 2 4
Case #4: 1 2 3 5
Case #5: 2 3 4 5 6 8
Case #6: 1 2 3
```

Problem C
Surveillance

Burglars are around! Lea's aunt fears that they will get to her house next and asks Lea to stay with her all night to defend her belongings. Lea has naturally better plans for tonight, so she has a another proposal. Lea will buy and install some surveillance cameras together with her aunt.

The cameras they buy are cutting-edge technology: They scan their environment using quantum technology and can detect any intruder with ease. To do so, they send out some quantum waves that are not blocked by anything: buildings, plants, not even by the great aquarium Lea's aunt installed in her bathroom. Unfortunately, the waves disturb each other, so no two cameras may watch the same area of the property.

The area a camera watches is a perfect circle around the point where the camera is installed. The software to control the cameras allows for only one size of the surveillance radius for all cameras. This is a huge restriction, so Lea spent some time hacking the software. She got bored with the awful code of the software, so she just implemented a way to set the radius of the first camera individually while the other cameras still need to have the same surveillance radius. A radius can never be set to 0 to switch the camera off but can be as small or large as required.

Lea wants to set the radii in a way that all but the first one are the same, no surveillance area overlaps with another and the total area seen by the cameras is maximal. Can you help her compute this area?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case starts with a line containing an integer n, the number of cameras. n lines follow describing the cameras. The i-th of them contains two space-separated real numbers x_i and y_i, the coordinates of the i-th camera. The radius of the first camera can be chosen arbitrarily, the radii of all other cameras have to be the same.

Output

For each test case, output one line containing "Case #i: a" where i is its number, starting at 1, and a is the maximum area that can be watched by the cameras. Each line of the output should end with a line break.

Your output must have an absolute or relative error of at most 10^{-6}.

Constraints

- $1 \leq t \leq 20$
- $2 \leq n \leq 10^4$
- $-100.0 \leq x_i, y_i \leq 100.0$ for all $1 \leq i \leq n$

Lea's aunt's property is very big, so we can assume it to be an infinite two-dimensional plane.

Sample Input 1

```
3
7
0.0 0.0
-2.0 0.0
2.0 0.0
-1.0 1.732
1.0 1.732
-1.0 -1.732
1.0 -1.732

3
-1.0 0.0
0.0 0.0
1.0 0.0

2
0.0 0.0
1.0 0.0
```

Sample Output 1

```
Case #1: 21.99018096
Case #2: 3.14159265
Case #3: 3.14159265
```

Sample Input 2

```
6
3
85.89749 8.041092
13.502266 80.74486
1.2792587 28.864044

2
10.871872 98.02222
-10.477295 -23.74871

5
6.6840286 99.15872
63.19284 37.411926
-99.12375 6.2733994
13.110519 -79.276566
51.04373 -90.824745

5
41.790314 22.764038
69.2648 -50.786854
0.49011993 -74.49112
5.434021 75.42493
18.820465 -99.197754

4
96.94888 -62.5437
-52.94235 -28.406075
-36.57235 4.2853622
27.117096 -20.070099

5
-76.06155 -0.68807983
12.408653 -34.555946
-44.162727 -99.990585
79.7007 -27.338135
-56.049953 22.013031
```

Sample Output 2

```
Case #1: 23856.75320794
Case #2: 48015.93349141
Case #3: 22009.72272370
Case #4: 12864.66353914
Case #5: 20987.36209114
Case #6: 11508.33809639
```

Problem D
Azrieli Towers

Elinor is in love with math and architecture. The only thing that can excite her even more is a combination of both. Therefore, it is not surprising that for a long time Elinor's favorite building was the ancient Pyramid of Cheops in Egypt due to its clear mathematical structure. However, a couple of days ago Elinor visited Tel Aviv, where she found a modern-style architectural counterpart to the ancient pyramid in Egypt - the Azrieli towers. This complex of buildings consists of three towers whose bases are in shape of a square, circle, and perfect triangle respectively.

Elinor was so amazed by this brilliant geometric design that she decided to construct her very own version of this landmark. In order not to be accused of plagiarism, she wants to build only two towers - one with a rectangular basis and the other one with an arbitrary triangular basis. Neither the orientation of the two towers, nor the base area matter to Elinor. However, she does not want the bases of her two towers to overlap or touch each other as this would completely destroy the beauty of the geometric design.

Unfortunately, the location at which Elinor wants to erect her two towers is known to be quite swampy so that the positioning of the towers is somewhat restricted. Luckily, Elinor is married to Nathan, a distinguished geologist, who already inspected the area thoroughly and came up with a list of specific coordinate sets at which the cornerstones of both buildings can be placed without facing problems with the statics. Given such a set of coordinates, Elinor has to decide now if her two towers can be built such that each edge of both buildings coincides with one of Nathan's cornerstone coordinates. Since this task is very time-consuming and repetitive, she wants you to write a computer program that does the job for her.

Input

The first line of the input contains an integer t. t cornerstone sets follow, each of them separated by a blank line.

Each cornerstone set starts with one integer n, where n is the number of cornerstone coordinates in the particular set. n lines follow describing the coordinates of the cornerstones. Each line contains two integers x_i and y_i where x_i is the x-coordinate and y_i is the y-coordinate of the i-th cornerstone position. One can safely assume that any three of the n cornerstone coordinates are non-collinear.

Output

For each test case, output one line containing "Case #i: s" with i being the number of the test case starting at 1, and s being "possible" if Elinor can construct her two towers with respect to the given cornerstone positions, and "impossible" otherwise. In case of "possible", output seven more lines each containing two integers x_i and y_i. The first four lines contain the x- and y-coordinates of the cornerstones of the rectangular tower. The latter three lines contain the x- and y-coordinates of the cornerstones of the triangular tower. Both coordinate sets can be given either in clockwise or counterclockwise order. If there are multiple possibilities to place the two towers, any of them will be accepted.

Constraints

- $1 \leq t \leq 50$
- $7 \leq n \leq 10$
- $0 \leq x_i, y_i \leq 1000$ for $1 \leq i \leq n$
- Any three of the n cornerstone coordinates are non-collinear

Sample Explanation

The first cornerstone set consists of 10 cornerstones. There are multiple possibilities to construct the two towers. One of them is illustrated in figure F.1.

In the second test case there are only 7 cornerstone coordinates given (see figure D.2). Even though it is possible to place a rectangle and a triangle with respect to the cornerstones, Elinor cannot erect her two towers because they would overlap.

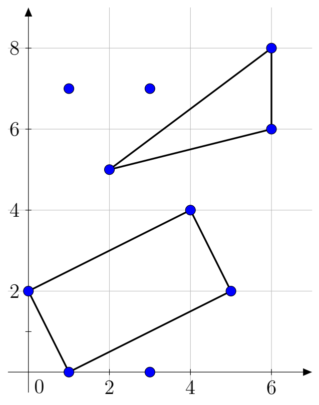

Figure D.1: First sample test case.

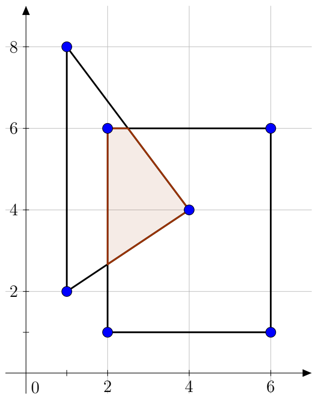

Figure D.2: Second sample test case.

Sample Input 1

```
2
10
0 2
1 0
1 7
2 5
3 0
3 7
4 4
5 2
6 6
6 8

7
1 2
1 8
2 1
2 6
4 4
6 1
6 6
```

Sample Output 1

```
Case #1: possible
0 2
1 0
5 2
4 4
2 5
6 6
6 8
Case #2: impossible
```

Problem E
Fragile Letters

The company Lea is working at recently bought a new building to provide offices for all employees. The new building is a skyscraper situated in the city center and widely visible from all over the town. Since the company invests heavily in advertisements, the management decided to write its name on the outer wall of the new building, too. They bought big letters that glow in the dark. The letters got delivered today, but will only be mounted next week.

When the shipping company was about to unload the letters, Lea went outside to have a break in the company-owned park and stopped by to see the huge letters. The workers were debating loudly, so Lea joined them and asked what they are arguing about. It turned out they were not sure how to position the letters. They should stand vertically, due to technical reasons, but it is possible to rotate them or even turn them upside down. Obviously, the letters should be in a stable position and should not break, but they do not even know how many such positions there are. Can you help them together with Lea?

Input

Lea measured all of the letters. Each of the letters represents one test case. Since they have a very modern font, the letters are polygons (as seen when standing next to them), which means they do not contain holes and consist of straight lines only. Lea measured the position of all of the letter's vertices and computed their two-dimensional coordinates.

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with an integer n, the number of vertices. n lines follow describing the vertices. The i-th line contains two doubles x_i and y_i, the coordinates of the i-th vertex, The points are given in order, but Lea forgot whether she wrote them down clockwise or counter-clockwise. Note that due to the modern font the letters may not look like what you would expect a letter to look like. Consider them as a general simple polygon.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the number of stable positions of the letter. Each line of the output should end with a line break.

A position is considered stable if it touches the ground with exactly one edge and no vertex except the ones incident to that edge. Standing on an additional vertex or multiple edges would break the letter since it is not made for standing on the ground. Additionally, the center of mass of the letter (i.e. the centroid of the polygon) needs to be above the lowermost edge or otherwise the letter would break. For instance, the letter "V" in normal fonts (Arial, for instance) has three stable positions, but a "T" has only two.

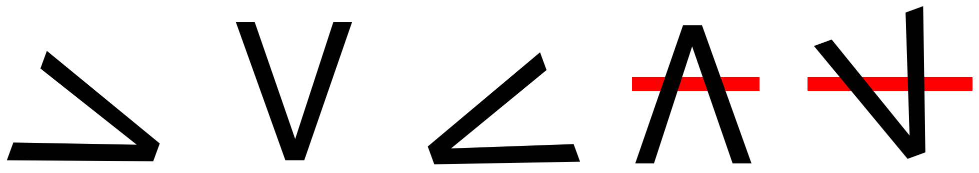

Figure E.1: The letter "V" has three stable positions.

Constraints

- $1 \leq t \leq 20$
- $3 \leq n \leq 50$
- $0 \leq x_i, y_i \leq 1000$ for all $1 \leq i \leq n$

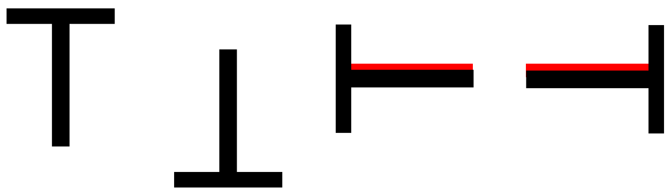

Figure E.2: The letter “T” has two stable positions.

Sample Input 1

```
3
7
0.0 2.0
1.0 2.0
2.0 1.0
3.0 2.0
14.0 2.0
13.0 0.0
1.0 0.0

5
0.0 0.0
1.0 0.0
2.0 1.0
3.0 0.0
3.0 5.0

8
1.0 0.0
1.0 2.0
0.0 2.0
0.0 3.0
3.0 3.0
3.0 2.0
2.0 2.0
2.0 0.0
```

Sample Output 1

```
Case #1: 1
Case #2: 2
Case #3: 2
```

Sample Input 2

```
3
3
69.65178077847345 476.5867758189318
821.1853328040016 88.77793357104213
647.0223390027927 271.8248148626079

4
181.44458803946185 940.3664887563629
9.073553867733452 668.9316108351632
300.5477457248187 232.847850879327
736.0950978507178 376.24666402600525

6
589.537247768286 277.42135454463437
894.9913870012382 124.39634658639598
45.423403877217 250.24312860783692
509.6408941478913 630.1328761488896
166.30830193549738 595.7205262511858
826.6580132415487 795.3440293977031
```

Sample Output 2

```
Case #1: 2
Case #2: 4
Case #3: 2
```

Problem F
Pathing

Lea is programming the next new hit game, Age of Conquercraft, a real time strategy game. Unfortunately, her units do not run as they are supposed to: Instead of taking the shortest path to the their target, they sometimes take a longer path or run into a dead end and have to walk back. Can you help her and write a better pathfinding algorithm?

The terrain that Lea uses is divided into squares of 1x1 cm which are either passable or impassable. The units are considerd to be a point (they do not occupy space) and can move in any direction (in particular, they do not have to move perpendicular to the grid) but cannot move into impassable squares.

Input

The first line of the input contains an integer t, the number of test cases. t test cases follow, each of them separated by a blank line.

Each test case starts with a description of the grid. The first line contains three integers w h n, where w and h are the width and height of the grid and n is the number of impassable locations. All following coordinates are 1-bases with $(1, 1)$ being the upper left corner.

n lines follow, describing the impassable locations. The i-th line contains four integers x_i y_i w_i h_i, describing an impassable square starting at grid position $(x_i,\ y_i)$ of width w_i and height h_i in the grid, extending to the right and downward. The grid description is followed by a blank line.

Two lines follow, the j-th of which contains two integers a_j b_j, the first is the start point (a_1, b_1) of the unit and the next is the target point (a_2, b_2).

Output

For each test case, print a line containing "Case #i: c" where i is its number, starting at 1, and c is a space separated list of coordinates $(X_1, Y_1)(X_2, Y_2) \dots (X_k, Y_k)$ (with brackets and comma) such that

- $(X_1, Y_1) = (a_1, b_1)$ is the start point,
- $(X_k, Y_k) = (a_2, b_2)$ is the target point,
- for every $1 \le i < k$ it is possible to move from (X_i, Y_i) to (X_{i+1}, Y_{i+1}) in a straight line without entering an impassable grid location, and
- the length of the path along the points in c is minimal.

Any optimal solution (up to relative or absolute errors of at most 10^{-4}) will get accepted. Each line of the output should end with a line break.

Constraints

- $1 \le t \le 20$
- $5 \le w, h \le 5000$
- $0 \le n \le 100$
- $1 \le x_i < x_i + w_i \le w$ for all $1 \le i \le n$.
- $1 \le y_i < y_i + h_i \le h$ for all $1 \le i \le n$.
- $1 \le a_j \le w$ for all $1 \le j \le 2$.
- $1 \le b_j \le h$ for all $1 \le j \le 2$.

- The start point and the target points will neither touch nor be contained in an impassable location.
- The impassable locations will not touch or overlap with each other or the border.

Sample explanation

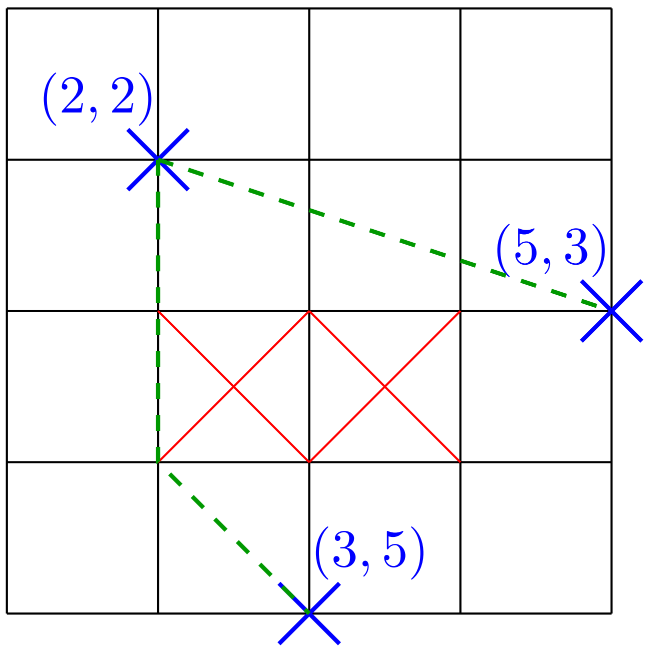

Figure F.1: Illustration of the first sample input

In the first sample, depicted in Figure F.1, in the first case it is possible to move from the starting point to the target in a straight line. In the second case, this is not possible because of the impassable locations, therefore we move via the point $(2, 4)$. Another accepted solution would be $(2, 2)$ $(2, 3)$ $(2, 4)$ $(3, 5)$. The solution $(2, 2)$ $(4, 3)$ $(4, 4)$ $(3, 5)$ would be wrong because the path is longer then the optimal one.

Sample Input 1

```
2
6 6 1
2 3 2 1
2 2
5 3

6 6 1
2 3 2 1
2 2
3 5
```

Sample Output 1

```
Case #1: (2,2) (5,3)
Case #2: (2,2) (2,4) (3,5)
```

Sample Input 2

```
4
7 6 2
2 2 2 1
3 4 3 1
3 1
4 6

7 7 3
2 3 2 1
3 5 3 1
5 2 1 2
3 2
5 7

7 7 2
4 2 2 1
3 2 1 3
2 1
3 1

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

Sample Output 2

```
Case #1: (3,1) (2,2) (2,3) (3,5) (4,6)
Case #2: (3,2) (4,3) (5,4) (6,5) (6,6) (5,7)
Case #3: (2,1) (3,1)
Case #4: (4,8) (1,8)
```