Topic 1: Geometry: Review, Half-plane Intersection, Adaptive Simpson **Lecture Notes**
CS 41100 - CP3 Competitive Programming III (Spring 2024) Purdue University
Instructor: Zhongtang Luo **Date:** February 3, 2024
Notes Preparer: Pedro Sugiyama

1 Considerations

1.1 Team Problem Considerations

This week, the contest will be solved in teams of up to three members. This is to help practice for official competitions, where entering as a team is common. Below are some of the questions each team member should ask themselves before and during team contests:

- How many problems can be reasonably solved within the time limit?
- Who should solve which problem?
- How can I best help my teammates?
- **Reminder:** each teammate is allowed to code one problem in class.

Remember: these factors are NOT static can change as time goes on / as your team solves problems.

2 Takeaways from Upsolve

Here's some of the things we learned from analyzing last week's problems:

- When the constraints are very small, do not try to waste time with any optimization.
- When solving a Simulation problem, tear down the problem into smaller steps. This will give a clearer view on the different functions you need to write.
- Utilize symmetry to reduce the amount of computations and edge cases to consider.

3 Geometry!

3.1 Initial Considerations

When solving Geometry problems, a reliable template is necessary. However, one must consider a few things when analyzing their template:

3.1.1 Completeness

How complete is your template, as in, how many questions can your template alone solve? Good examples of this are Voronoi Diagrams, discussed as possible solutions to example problems below, and the use of Neural Networks to solve very specific problems. If these approaches were not part of the template, solving such problems would prove much more difficult.
A complete template will allow a user to solve all problems that necessitate a very specific data structure or algorithm to be solved.

3.1.2 Understanding

How well do you know your template? Given that a template has the correct approach to solve the problem, how quickly can you 1. identify and 2. find the code in your template? This is important because knowing what your template allows you to do will broaden the possibilities you consider when analyzing a problem.

3.1.3 Content

How large is your template? What should you include / exclude? Ultimately, it is up to the user. If you know how to code Dijkstra's algorithm from memory, including it in your template will make it unnecessarily long and polluted. The best way to narrow down what is essential to your template is by making and using your own!

3.2 Precision and Speed Considerations

In most (if not all) Geometry problems, some level of error is expected, as we are dealing with floating point numbers. Although normally double precision is enough, repeatedly using some operations such as line intersections can lead to rapid loss of precision. However, using more precision requires a time sacrifice:
A double can support 15 digits, a long double, 30, and a __float128, 60. The performance degrades 4x per precision increase (i.e. double is 16x slower than __float128).
In general, using a double should suffice. Figuring out, however, if the problem requires more precision or has an implementation issue requires a lot of skill! For an example where more precision was necessary, read this article.

3.3 Point-based approach

A line / region contains infinite points. Trying to consider all possibilities within that line / region, therefore, is tremendously inefficient and practically impossible. However, if we can define a few key points of interest, the problem becomes enumerable. In last week's problem Alice and Bomb, there were three types of points that had to be considered, making checking all of them a possibility. As long as the amount of points isn't too large, the problem becomes computable within the time constraints. Another example is the problem below:

> **Sample Problem:** Birthday Cake
> **Link:** `https://vjudge.net/problem/Gym-104114B`

Intuition. For this problem, we can realize that the optimal cut must lie on a strawberry. If an optimal cut does not intersect any strawberry nodes, it can be translated until it intersects a strawberry without decreasing its chocolate chip count. However, we do not have to consider all strawberries. Consider a convex hull of the strawberries. If the cut crossed the hull at two points, it would divide the hull into two polygons, where it is guaranteed that a strawberry vertex exists in each polygon. As such, there would be strawberries in both resulting slices. As such, we only need to consider intersections with the strawberries that define the hull.

Implementation. First, create a convex hull of strawberries. For each strawberry vertex in the hull, calculate the optimal cut at that strawberry. To do this, we can employ the sliding window technique. Since each cut divides the cake in half, we are looking for the 180 degree window that satisfies two conditions:

- Does not contain the previous or next points in the strawberry hull, as this would mean the cut intersects with the hull at another point.
- Contains the maximum number of chocolate chips as possible.

Again, we can use a point-based approach to simplify this check, as only chocolate chips or adjacent strawberry vertices should be considered as a starting point for the window. Now, we can repeat this process for each strawberry vertex in the convex hull to find the best possible cut.

Analysis. Worse-case, all strawberries could be a part of the convex hull. As such, we would have to check each of the n chocolate chip starting points for each of the m strawberries, taking $O(nm)$. This is fast enough to pass, as $n \leq 5 \cdot 10^5$ and $m \leq 10^2$.

> **Sample Problem:** Panda Reserve
> **Link:** `https://vjudge.net/problem/Gym-498660B`

Intuition. This problem looks like a bisection problem, as we are looking to find the minimum acceptable radius of circles to cover the polygon. Trying to calculate the total area covered by the circles, however, is quite complicated – the interlapping circles making it almost impossible to do so. How can we change this into a point-based approach? Consider a circle with radius r at some vertex v_1. If the portion of this circle that lies within the polygon overlaps at its boundary (an arc) with a portion of another circle at v_2, we can be sure that the segment v_1 and v_2 is completely covered by circles. To check a triangle with vertices v_1, v_2 and v_3, a similar calculation can be performed. For each arc, it must have some overlap at all points with another arc.

Implementation. Since this problem involves bisection, we must discuss the check function. Using our previously discussed intuition, we must calculate if each portion of each arc within the polygon has some overlap at all points with at least one other arc. This can be done by analyzing each arc one by one. For each arc, check its intersection with all other circles using the circle-circle intersection template. If at any intersection point the arc fails to overlap with at least one other circle, we must increase the radius. Otherwise, we can try to decrease it.

Analysis. Checking each intersection with all n other circles takes $O(n^2)$. Since the bisection should not take more than 10^2 iterations to narrow down an acceptable radius, this passes as $n \leq 2 * 10^3$.

4 Half Plane

A half plane is a planar region formed of points such that there exists a line where all points stand to one side of it, with no points on the other side. This can be calculated in $O(n \log(n))$, where n is the number of points considered. There are 4 problems solved by this technique: Visibility in the plane, biggest circumference inscribed in a polygon, convex polygon intersection, and 2D linear programming. For a more detailed explanation, read this article.

> **Sample Problem:** Art Gallery
> **Link:** `https://vjudge.net/problem/Baekjoon-3800`

Intuition. This is a direct implementation of the Half-Plane technique, as visibility is one of the properties guaranteed by the half-plane.
Implementation. Plug the given points into the half-plane template.
Analysis. This passes as there can be at most 1500 points.

4.1 Numerical Integration

When trying to integrate a complicated function, taking the anti-derivative can be very hard and time consuming. We can numerically approximate the result by adding smaller and smaller rectangles under the graph together. One such method is Adaptive Simpson. This has two main effects:

- **Pro:** Taking the anti-derivative is no longer necessary, and the programmer no longer needs a strong understanding of calculus to solve the problem.
- **Con:** Turns every computation into a precision problem. It becomes difficult to understand if your solution is correct or if the result is too imprecise.

Most times, however, finding the anti-derivative becomes so complicated that numerical integration is much faster to implement. A few such problems are described below:

Sample Problem: Garden of Thorns
Link: `https://naq23.kattis.com/contests/naq23-fall/problems/naq23.gardenofthorns`

Intuition. First, we must figure out a way to use a point-based approach to solve this problem. Instead of trying to calculate random placements for the circle of thorns, consider that for a plant at point p all centers within the radius r of p protect the circle. Put simply, all centers that protect p lie in a circle of radius r centered at p. If we let I be the intersection between this circle and the rectangle, the probability that a plant is protected will be equal to the area of I divided by the area of the rectangle. This process can be repeated for each plant to obtain the expected value.

Implementation. Instead of trying to find the anti-derivative for all circles centered at the different plants, use a numerical integration method such as Adaptive Simpson to approximate the result. Implementation-wise, this is simple, as we can enumerate each plant and figure out its area quite easily.

Analysis. Even though it is difficult to estimate how quickly Adaptive Simpson runs, as it depends on the desired ε, the small number of plants (10) means this will almost certainly pass given some tweaking.

Sample Problem: Military Maneuver
Link: `https://vjudge.net/problem/Gym-104869G`

Intuition. For this problem, instead of calculating the expected value of the donut that starts at radius r_1 and ends at r_2, we instead consider the expected value of the outer circle of radius r_2 and subtract the expected value of the inner circle of radius r_1. These radii, however, must enclose all enemies within their bounds. In other words, The larger radius r_2 must enclose all enemies within it while the smaller one, r_1, must not enclose any. Not all pairs of given enemies fulfill this requirement. Using brute force, we can enumerate all pairs of points to figure out which two are the farthest / closest, depending on which circle we are considering. However, this is too slow, as it will take $O(n^2)$. However, we can use the slope trick to compute this in $O(n)$.

Birthday Cake

How exciting! Today is your little brother's birthday! That's why you ordered a huge (1×1)-meter cake. It is a special vanilla cake with n sweet chocolate chips and m refreshing strawberries.

You show him your awesome surprise, and... bummer! It turns out that he hates fruit! "*Of course, how could I have forgotten?*" you say. Nonetheless, he has a sweet tooth for chocolate, so he would be happy if you could cut him a piece of the cake that contains no strawberries. To make him happy, you'd want to give him a piece having as many chocolate chips as possible.

The picture above depicts the example test case.

You can only make one cut along a straight line through the cake, and you are not allowed to cut though either chocolate chips or strawberries. What is the maximum number of chocolate chips that you may give your little brother?

Note: The picture above is for illustration purposes. You should consider both chocolate chips and strawberries to be infinitesimally small.

Input

The first line of the input contains two positive integers n $(1 \leq n \leq 50\,000)$ and m $(1 \leq m \leq 100)$ — the number of chocolate chips and strawberries, respectively.

The i-th of the next $n+m$ lines contains two decimal numbers x_i and y_i, $(0 < x_i, y_i < 1)$, representing the coordinates of the i-th ingredient: the first n of the ingredients are chocolate chips, and the remaining m are strawberries.

All numbers are given with at most 6 decimal places. The locations of all $n + m$ ingredients are distinct.

Output

Output a single non-negative integer c, representing the maximum number of chocolate chips that you can give your little brother after cutting the cake exactly once.

Example

Input

```
5 2
0.2 0.6
0.8 0.6
0.6 0.2
0.1 0.2
0.6 0.8
0.6 0.6
0.5 0.5
```

Output

```
3
```

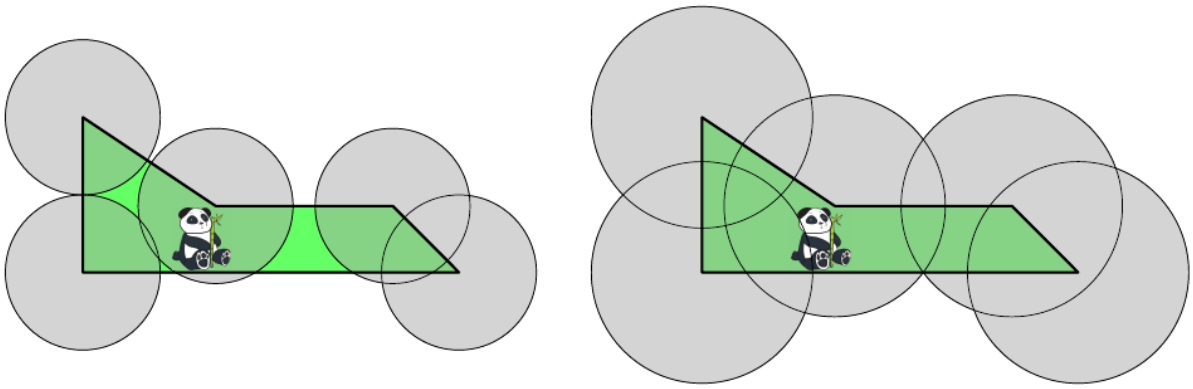
Source

2022 ICPC Southeastern Europe Regional Contest

Panda Reserve

Last month, Sichuan province secured funding to establish the Great Panda National Park, a natural preserve for a population of more than 1800 giant pandas. The park will be surrounded by a polygonal fence. In order for researchers to track the pandas, wireless receivers will be placed at each vertex of the enclosing polygon and each animal will be outfitted with a wireless transmitter. Each wireless receiver will cover a circular area centered at the location of the receiver, and all receivers will have the same range. Naturally, receivers with smaller range are cheaper, so your goal is to determine the smallest possible range that suffices to cover the entire park.

As an example, Figure G.1 shows the park described by the first sample input. Notice that a wireless range of 35 does not suffice (a), while the optimal range of 50 covers the entire park (b).



(a) An insufficient range for covering the park.

(b) The minimal range for covering the park.

Figure G.1: Illustration of Sample Input 1.

Input

The first line of the input contains an integer n ($3 \leq n \leq 2000$) specifying the number of vertices of the polygon bounding the park. This is followed by n lines, each containing two integers x and y ($|x|, |y| \leq 10^4$) that give the coordinates (x, y) of the vertices of the polygon in counter-clockwise order. The polygon is simple; that is, its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex.

Output

Display the minimum wireless range that suffices to cover the park, with an absolute or relative error of at most 10^{-6}.

Example

Input

```
5
0 0
170 0
140 30
60 30
0 70
```

Output

```
50
```

Input

```
5
0 0
170 0
140 30
60 30
0 100
```

Output

```
51.538820320
```

Input

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

Output

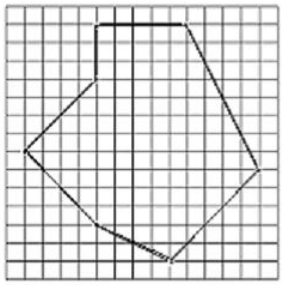
```
1.581138830
```

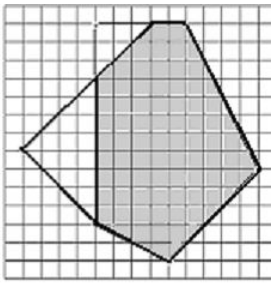
Source

2018 ACM-ICPC World Finals

Art Gallery

The art galleries of the new and very futuristic building of the Center for Balkan Cooperation have the form of polygons (not necessarily convex). When a big exhibition is organized, watching over all of the pictures is a big security concern. Your task is that for a given gallery to write a program which finds the surface of the area of the floor, from which each point on the walls of the gallery is visible. On the figure 1. a map of a gallery is given in some co-ordinate system. The area wanted is shaded on the figure 2.





Input

The number of tasks T that your program have to solve will be on the first row of the input file. Input data for each task start with an integer N, $5 \leq N \leq 1500$. Each of the next N rows of the input will contain the co-ordinates of a vertex of the polygon — two integers that fit in 16-bit integer type, separated by a single space. Following the row with the co-ordinates of the last vertex for the task comes the line with the number of vertices for the next test and so on.

Output

For each test you must write on one line the required surface — a number with exactly two digits after the decimal point (the number should be rounded to the second digit after the decimal point).

Example

Input

```
1
7
0 0
4 4
4 7
9 7
13 -1
8 -6
4 -4
```

Output

```
80.00
```

Source

Southeastern Europe 2002

Garden of Thorns

Eddy owns a rectangular garden and has noticed some trespassers stomping through his garden. There are some plants that he wants to protect. He hires an assistant, Zyra, to patrol and protect his garden.

Zyra cannot be bothered to monitor his garden, so she plants a circle of thorns centered at a randomly chosen location within the boundaries of his garden. A plant is considered protected if it is strictly inside the circle of thorns - that is, the distance from the plant to the center of the circle of thorns is less than the circle's radius. The circle of thorns may extend outside of the boundary of the rectangular garden, though all plants will be inside or on the boundary of the garden.

Given the random nature of the placement of Zyra's circle of thorns, compute the expected value of the plants that will be protected. Note that Zyra's circle of thorns does not have to be centered at integer coordinates.

Input

The first line of input contains four integers n ($1 \leq n \leq 10$), r ($1 \leq r \leq 2000$), w and h ($1 \leq w, h \leq 1000$), where n is the number of plants in Eddy's garden, r is the radius of Zyra's circle of thorns, w is the width of Eddy's garden and h is the height of the garden.

Each of the next n lines contains three integers x ($0 \leq x \leq w$), y ($0 \leq y \leq h$) and v ($1 \leq v \leq 1000$), where (x, y) denotes the position of a plant from the lower left corner of Eddy's garden, and v is the value of that plant. No two plants will be at the same position.

Output

Output a single real number, which is the expected value of plants which will be protected by Zyra's circle of thorns. Any answer within an absolute or relative error of 10^{-6} will be accepted.

Example

Sample Input 1

```
3 50 100 100
30 10 3
40 10 7
50 90 8
```

Sample Output 1

```
8.419064869324508038062049308 79
```

Sample Input 2

```
2 5 3 4
0 0 10
3 4 15
```

Sample Output 2

```
25.0
```

5 Source

North American Qualifier 2023

Military Maneuver

A military maneuver is going on a two-dimensional Cartesian plane, and n enemy targets are hiding somewhere on the battlefield, whose locations are known to our headquarters.

Our headquarters will airdrop a beacon in a rectangular region with sides parallel to the coordinate axes uniformly at random to expose all the enemy targets to our troops on the battlefield so that our troops can surround all the enemy targets. The bottom-left corner of the region is at coordinate (x_l, y_l) while the top-right corner is at coordinate (x_r, y_r).

After being dropped, the beacon will firstly receive two parameters r and R that satisfy $0 \le r \le R$ from our headquarters, then scan an annulus region, that is, the region lying between two concentric circles, where the radius of the inner circle is r and that of the outer circle is R, and finally mark those enemy targets hiding in the scanned region (including the boundary).

However, the beacon can only scan a unit area in a unit of time, and the commander would like to know the expected minimum time for the beacon to scan the designated annulus region so that it can mark all the enemy targets.

Input

The first line contains four integers x_l, y_l, x_r, and y_r ($-10\,000 \le x_l, y_l, x_r, y_r \le 10\,000$, $x_l < x_r$, $y_l < y_r$), denoting the coordinates of the bottom-left and the top-right corners of the rectangular region where the beacon will be dropped.

The second line contains a single integer n ($2 \le n \le 2\,000$), denoting the number of enemy targets on the battlefield.

Each of the following n lines contains two integers x and y ($-10\,000 \le x, y \le 10\,000$), denoting an enemy target located at coordinate (x, y).

It is guaranteed that no two enemy targets share the same locations.

Output

Output a single real number, indicating the expected minimum time for the beacon to scan the designated annulus region.

Your answer is acceptable if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Example

Input

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

Output

```
8.377580409572781970
```

Input

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

Output

```
37.699111843077518863
```

Note

In the first sample case, if the beacon is dropped to $(0.5, 1.5)$, the minimum time as well as the minimum area of the feasible annulus region is 4π. The expected minimum time when the beacon dropped in the rectangular region uniformly at random is $\frac{3}{8}\pi$.

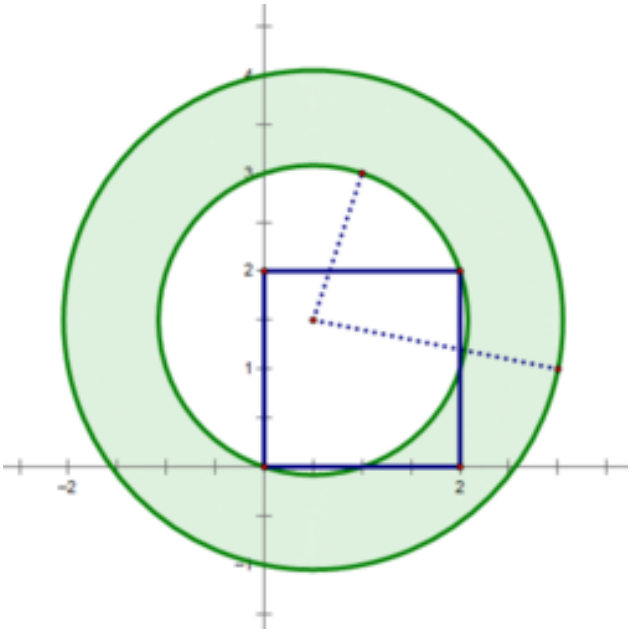


Figure: The feasible annulus region for the beacon at $(0.5, 1.5)$

Source

The 2023 ICPC Asia Shenyang Regional Contest (The 2nd Universal Cup. Stage 13: Shenyang)