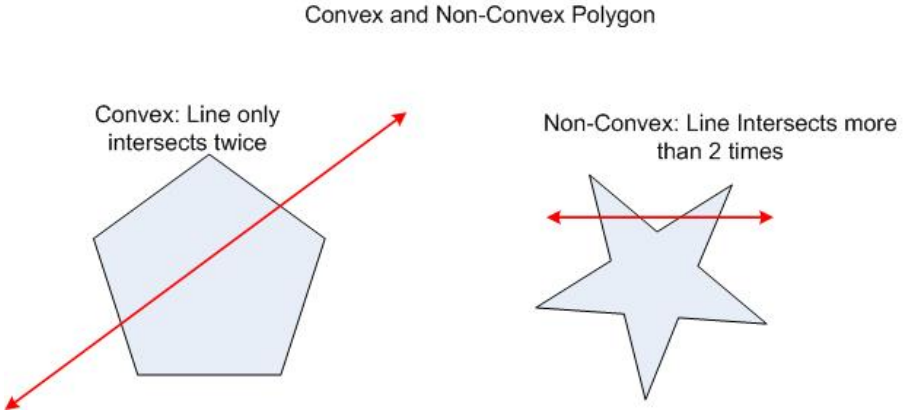
January 2023 CSE 106
Offline Assignment on Divide and Conquer and Greedy Algorithms

Deadline: Saturday, 2 September 2023, 11:55 PM
Last modified : Saturday 26th August, 2023 16:48

Task-1: Optimal 2D Fence Building

Convex Polygon

A convex polygon is a closed figure where all its interior angles are less than 180° and the vertices are pointing outwards. All the lines across the outline are straight and they point outwards.



Scenario

Imagine you are a land developer and you are tasked with enclosing a set of trees on a 2D plot of land. Your objective is to construct a fence that fully encloses the trees while minimizing the length of the fence.

You are presented with a landscape containing various trees, each at distinct 2D coordinates. Your mission is to design a fence layout using fence posts connected by straight-line segments, forming a convex polygon. **The polygon must enclose all the trees completely, and your goal is to achieve this while minimizing the perimeter of the fence.**

Problem Statement

Given the coordinates of the trees in the 2D field, your task is to determine the positions for the fence posts. These posts will be used to construct a convex polygonal fence that surrounds all the trees and achieves the smallest possible perimeter.

You have to solve this problem using Divide and Conquer Approach.

I/O

Take input from a file named *in.txt*. You will generate output on a file named *out.txt*. **If you don't take input from a file and output to a file, you will receive a penalty of 5 marks in this task.**

Input

The input consists of a single integer N, representing the number of trees ($3 \leq N \leq 10^5$). The following N lines will contain two integers each, representing the x and y coordinates of the trees. **The coordinates of trees are guaranteed to be unique.**

Output

Print the coordinates of the fence, one coordinate per line. Each coordinate should be represented as two space-separated integers: x-coordinate and y-coordinate.

You don't need to output any images.

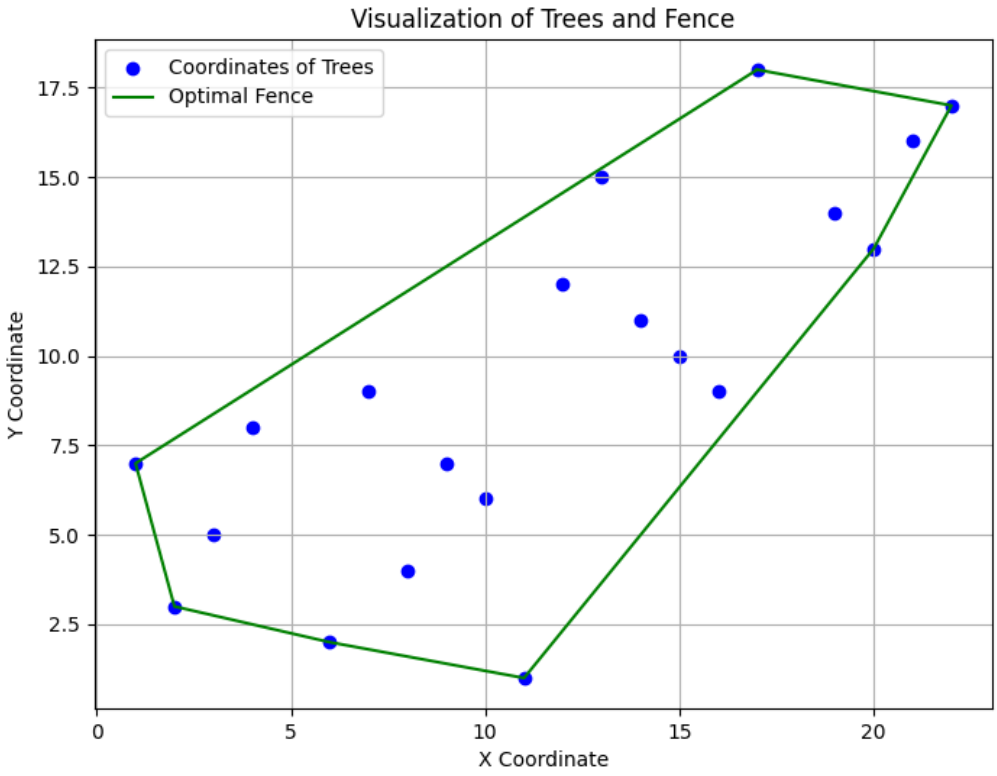


Figure 1: An Example of Fence Building

Task 2: Minimum Number of Platforms Required for Train Schedule

Imagine you're in charge of a busy train station. Your job is to make sure that trains arrive and leave on time without any delays. You need to figure out how many platforms you need to prevent any train from waiting.

Each train has a specific schedule for when it arrives and when it departs. Trains can only arrive and leave at the exact times listed in their schedules. If a train arrives exactly when another one leaves, they can share the same platform without any issues.

Your task is to create a program that looks at the train schedules you have and tells you the smallest number of platforms you need. This way, you can ensure that all trains run smoothly, passengers are happy, and everything stays on track.

Write a program that takes the train schedule as input and outputs the minimum number of platforms required.

Take input from a file named *input.txt*.

Input

The input consists of an integer N ($1 \leq N \leq 10^5$), representing the number of trains in the schedule. The next N lines contain two integers each, representing the arrival and departure times of each train. **The times are given in 24-hour format without leading zeros.**

Output

Print a single integer, representing the minimum number of platforms needed.

Note

1. Trains can arrive and depart at the same time.
2. The departure time of a train will always be greater than or equal to its arrival time.

Input File

```
5
900 940
950 1000
1100 1120
1130 1145
1000 1200
```

Output

```
2
```

Explanation:

(a) Train 1 arrives at 9:00 and departs at 9:40.

(b) Train 2 arrives at 9:50 and departs at 10:00.

(c) Train 3 arrives at 11:00 and departs at 11:20.

(d) Train 4 arrives at 11:30 and departs at 11:45.

(e) Train 5 arrives at 10:00 and departs at 12:00.

To avoid any delay, we need at least 2 platforms: one for trains 1, 2, and 5, and another for trains 3 and 4.

Submission Guideline

1. Create a directory with your 7-digit student id as its name
2. Put the source files only into the directory created in Step 1
3. Zip the directory (compress in .zip format; .rar, .7z, or any other format is not acceptable)
4. Upload the .zip file on Moodle.

For example, if your student id is 2105xxx, create a directory named 2105xxx. Put only your source files (.c, .cpp, .h, etc.) into 2105xxx. Compress 2105xxx into 2105xxx.zip and upload the 2105xxx.zip on Moodle.

For Queries

If you have any questions related to the assignment please first check the queries thread in Moodle. You should post your confusion in the thread. If your query goes unanswered for 24 hours, please mail **me**.

Mark Distribution

The tentative mark distribution are given below:

	Task Detail	Marks
1	Proper Implementation of Task-1	50
2	Proper Implementation of Task-2	40
3	Proper Formatting	5
4	Proper Submission	5

Special Instructions

When writing code, it is essential to ensure readability, reusability, and good structure. This involves using appropriate functions to implement algorithms, giving variables meaningful names, adding suitable comments when necessary, and maintaining proper indentation. You will need to use your offline implementation to solve the online. There will be a viva too. So, please understand the concepts before you proceed to code.

Please note that you must rely on your own implementation to solve the assigned tasks. It is strictly prohibited to copy code from any source, including friends, seniors, or the internet. **Any form of plagiarism, regardless of its origin or destination, will result in a deduction of 100% marks for the offline assessment**. Moreover, repeated instances of plagiarism may lead to stricter consequences in accordance with departmental policies.