ALGORITHMS 2021-2022

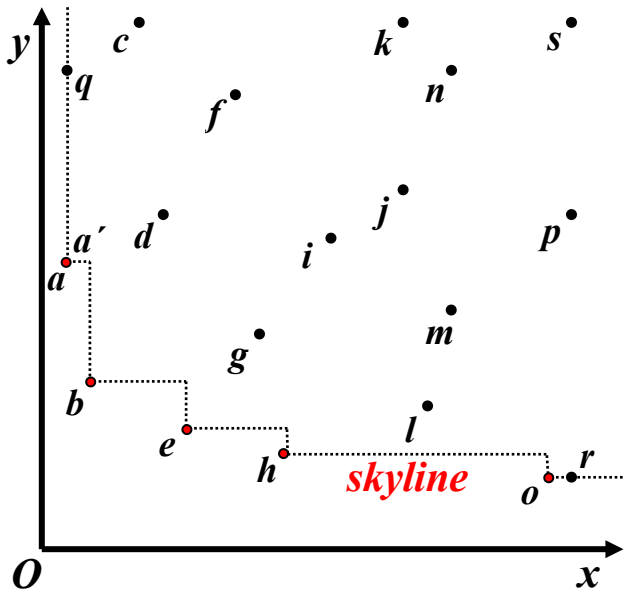
A PROGRAMMING EXERCISE (1 point)

Efficient Algorithm for skyline queries

Definition: when a dominance relation is given to a dataset, then **the ridge query returns all objects that are not dominated by any other objects in the set**. In case the dataset consists of multidimensional objects, one object will dominate another if it is the same or better in all dimensions and at least in one strictly best.

Formally, for two two-dimensional objects **A(x1,y1)** and **B(x2,y2)**, assuming that the best value is the minimum in each of the two dimensions, we will have that **A dominates B** if and only if: **x1≤x2 and y1≤y2 and (x1≠x2 or y1≠y2).**

Example: A simple example is when objects are points in the Euclidean plane, assuming that the best value in each of the two dimensions is the minimum. Consider the 20 points in the following figure: a(1,12), a'(1,12), b(2,7), c(4,22), d(5,14), e(6,5), f(8,19), g(9,9), h(10,4), i(12,13), j(15,15), k(15,22), l(16,6), m(17,10), n(17,20), o(21,3), p(22,14), q(1,20), r(22,3), s(22,22). Then the ridge query returns the set of points: {(1,12), (2,7), (6,5), (10,4), (21,3)}.



What needs to be implemented

In this task you have to implement an efficient algorithm for finding the skyline on two-dimensional point sets in the Euclidean plane.

As input, a file will be given in the following format:

```
20
1 12
1 12
2 7
4 22
5 14
6 5
8 19
9 9
10 4
12 13
15 15
15 22
16 6
17 10
17 20
21 3
22 14
1 20
22 3
22 22
```

That is, it will contain the data of the points where the first line will contain the number N of points and in each subsequent line the coordinates of the points will follow, which will be integers in the interval [0...1000] separated by a space.
The program should calculate and print on the screen the coordinates of the ridge points strictly in the following output format:

```
1 12
2 7
6 5
10 4
21 3
```

(i.e. in ascending order in terms of their abscissa).

Method of implementation:

1) Use only the **Java** language, following the instructions for submitting java exercises to the Eagle system (**Instructions-Eagle-2022.pdf**).
2) The **only input argument** will be the input file. This argument should be **args[0**] from the command line arguments, which is read by **main.**
3) For the solution, use an algorithm that is efficient [**O(NlogN)**], using methods taught in the course (such as divide&conquer, etc.) or from the literature, as N may be large and **there will be a time limit on the execution of the test cases**.
4) The source code should have **concise inline comments** and **comments** above each function, explaining the rationale for your implementation.

Deliverable:

- **The** whole program (**source code**) should be implemented in a **single java file named Skyline.java** (Skyline class).
- This file should be uploaded and **submitted to** the system **Eagle** (eagle.csd.auth.gr) with your account.
- The Eagle system will **automatically execute** your code in various test cases (both overt and covert) and **score it automatically** (maximum score of **100**).
- You can submit to the system up to 20 times. Therefore you should have first implemented it in another environment of your choice (e.g. Netbeans, IntelliJ, etc.) and tested it to **run correctly** and **produce correct results** before submitting.
- **Start of submissions: Monday 2/5/2022 10:00am.**
- **End of submissions: Thursday 12/5/2022 11:59pm.**

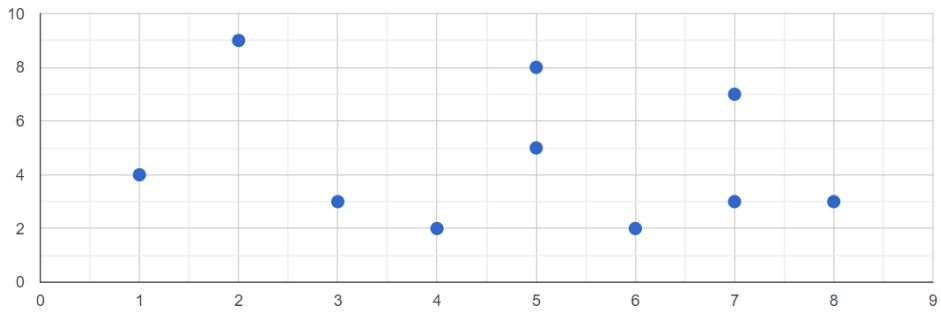
Clarifications:

- It is allowed to use ready-made code freely available (ready-made implementations with appropriate customization by you) provided that the source is clearly indicated in the comments. However, the programs will be checked by an automatic copy detection system. If copying between students is detected, these students will be zeroed out.
- At the top of the java file with the source code, be sure to include in comments your full name, AEM and academic email.
- *Students should be prepared to give oral explanations of their implementation if asked.*
- ***The grades of the exercises will also be valid for the September or degree examinations.***

A few more test cases:

Uniformly distributed:

```
10
1 4
5 8
6 2
7 3
2 9
3 3
4 2
5 5
8 3
7 7
```

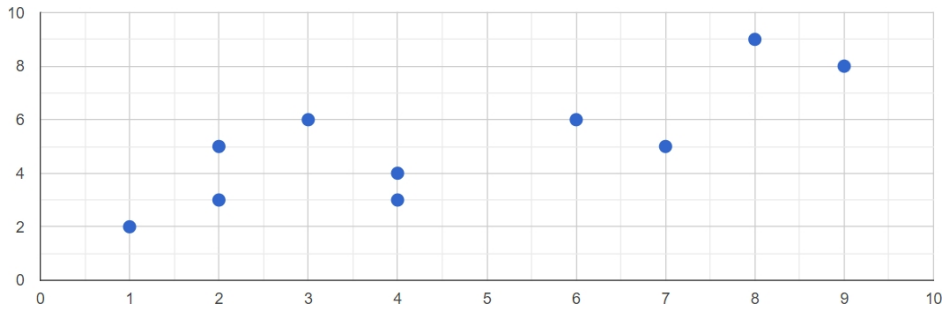


Skyline:

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

Correlated:

```
10
2 5
4 3
3 6
7 5
4 4
1 2
2 3
6 6
8 9
9 8
```

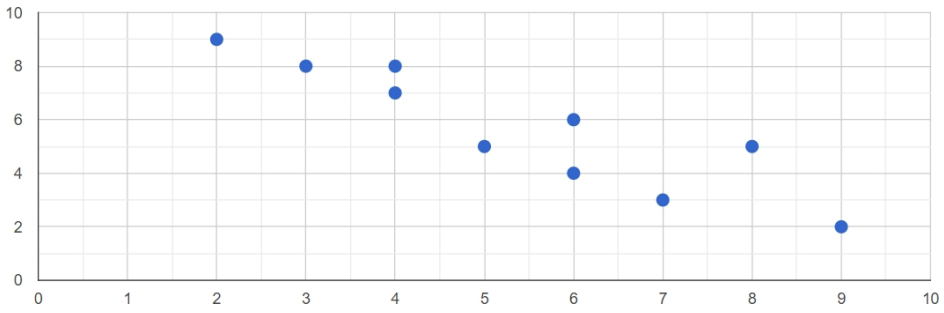


Skyline:

```
1 2
```

Anti-Correlated:

```
10
9 2
8 5
6 4
7 3
5 5
6 6
4 7
4 8
3 8
2 9
```



Skyline:

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