Data Structures and Algorithms
Spring 2020
Assignment 2

IT University Innopolis

Contents

1 About this homework

1.1 Coding part

For practical (coding) part you have to provide you program as a single Java class file that contains all necessary definitions and a `main` method.

The program should read from standard input and write to standard output unless mentioned otherwise.

1.1.1 Preliminary tests

For some coding parts a CodeForces or CodeTest system will be used for preliminary tests. You will have to specify a submission number in those systems as a proof that your solution is correct. Only correct solutions will be assessed further by TAs.

1.1.2 Code style and documentation

The source code should contain adequate internal documentation in the form of comments. Internal documentation should explain how the code has been tested, e.g. the various scenarios addressed in the test cases.

Do not forget to include your name in the internal documentation.

All important exceptions should be handled properly.

Bonus points might be awarded for elegant solutions, great documentation and the coverage of test cases. However, these bonus points will only be able to cancel the effects of penalties.

1.2 Theoretical part

Solutions to theoretical exercises have to be elaborate and clear. Write all solutions in a single document and make it a PDF for submitting.

Do not forget to include your name in the document.

1.3 Submission

You are required to submit your solutions to all parts via Moodle.

For this assignment you will need to submit: - Java class file(s) for the coding parts; - submission number or a link to submission in CodeForces as a proof of correctness; - a PDF file with solutions to theoretical parts.

Submit files as a single archive with your name and group number. E.g. `BS18-00_Ivanov_Ivan.zip`.

1.4 Plagiarism

Plagiarism will not be tolerated and a plagiarised solutions will be heavily penalised for all parties involved. Remember that you learn nothing when you copy someone else's work which defeats the purpose of the exercise!

You are allowed to collaborate on general ideas with other students as well as consult books and Internet resources. However, be sure to credit all the sources you use to make it clear what part of your solution comes from elsewhere.

2 Segment intersection (70 points)

2.1 Segment intersection

Two line segments

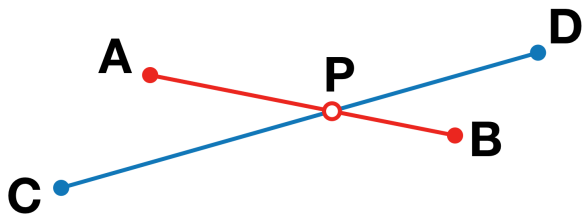

Figure 1: Two intersecting line segments.

Given two line segments on a 2D plane it is often natural to ask whether they intersect and, if so, what is the intersection point. Given the coordinates of the endpoints of the line segments AB and CD, if the intersection point P exists, it can be computed using the following formulae:

$$x_P = \frac{(x_A y_B - y_A x_B)(x_C - x_D) - (x_A - x_B)(x_C y_D - y_C x_D)}{(x_A - x_B)(y_C - y_D) - (y_A - y_B)(x_C - x_D)}$$

$$y_P = \frac{(x_A y_B - y_A x_B)(y_C - y_D) - (y_A - y_B)(x_C y_D - y_C x_D)}{(x_A - x_B)(y_C - y_D) - (y_A - y_B)(x_C - x_D)}$$

It is important to consider all special cases that might happen:

- the lines are parrellel or coincident; in this case the denominator is zero:

$$(x_A - x_B)(y_C - y_D) - (y_A - y_B)(x_C - x_D) = 0$$

- lines intersect but line segments do not; in this case you should check that intersection point belongs to both line segments (or use an alternative method, e.g. via first degree Bezier parameters);
- line segments share an endpoint.

You can refer to CLRS pp. 1017–1019 for a detailed description of the cases.

Importance of line segment intersection

Is the problem of finding an intersection of two line segments important? Find out about applications of line segment intersection and write a paragraph explaining **in your own words** why this problem is important and attach it as a code comment to the implemented algorithm.

Naive n-segment intersection

Given a collection of n line segments we can find all intersections by checking every pair of line segments:

```
for i from 1 to n
  for j from i+1 to n
    if segments[i] intersects with segments[j] then
      ...
```

It is easy to see that the time complexity of this algorithm is $O(n^2)$.

However, there exist a more asympotically efficient solution. Find out about the Sweep Line algorithm and summarize it **in your own words** in one or two paragraphs of code comments.

To summarize, in this section of the assignment you need to

- implement the algorithm determining if there exists an intersection point of two line segments in Java;
- write the paragraph about importance of this algorithm in a form of code comment;
- find out about the Sweep Line algorithm and summarize it **in your own words** in one or two paragraphs of code comments.

2.2 Sorting

Implement any of the following sorting algorithms:

- any version of `quicksort`;
- any version of `mergesort`.

Make sure that the sorting algorithms are generic and can be used with any comparable type.

Document the properties of implemented algorithm:

- What is the time complexity of the algorithm?
- Is it in-place or out-of-place?
- Is it stable?

2.3 Storage

For intermediate storage you need to implement a self-balancing binary search tree:

- either an AVL tree;
- or a red-black tree.

Make sure that these data structures are generic and can be used with any comparable type.

Annotate every operation on the data structure with asymptotic analysis of its time complexity.

2.4 Sweep line algorithm

Implement Sweep line algorithm using the sorting algorithm and self-balanced binary search tree you implemented in other parts of this homework.

Write a program that takes a list of line segments as input and checks if any two lines segments intersect.

3 Theoretical part (30 points)

Least squares

It is often useful to approximate real data with some simple (in some sense) function.

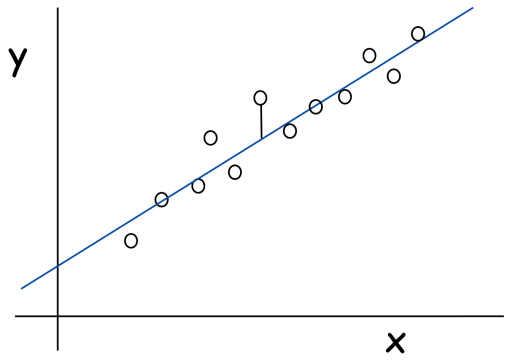

Figure 2: Approximating a set of points using least squares.

Given n points on a 2D plane $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ we can try to approximate that data using a straight line $y = ax + b$, where a and b are constants. One of the methods to estimate how well a given line approximates data is to compute the *sum of squared errors* (SSE), which can be summarized in the following formula

$$SSE = \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

Least squares is a method of finding the line that minimizes SSE for a given data set. Minimum SSE is achieved exactly when

$$a = \frac{n \sum_i (x_i y_i) - (\sum_i x_i)(\sum_i y_i))}{n \sum_i x_i^2 - (\sum x_i)^2} \qquad b = \frac{\sum_i y_i - a \sum_i x_i}{n}$$

Connected lines least squares

In this question you need to approximate a data set not with a single line, but with several connected lines.

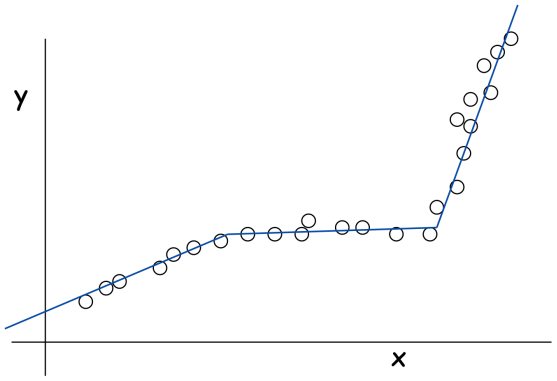

Figure 3: Approximating a set of points using connected lines least squares.

Here, we define cost as the sum of the sums of squared errors for each segment and the number of lines. For example, creating $n-1$ lines connecting each pair of adjacent points will result in zero total sum of squared errors, but with a high number of lines. On the other hand, creating a single line connecting the first to the last point will minimize the number of lines but also result in a high SSE.

You are asked to use dynamic programming to find the minimum cost (defined above) for given n points in the plane $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with $x_1 < x_2 < \ldots < x_n$.

Use the following notation:

- MIN(j) = minimum cost for points$(x_1, y_1), (x_2, y_2), \ldots, (x_j, y_j)$
- $e(i, j)$ = minimum sum of squared errors for points$(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_j, y_j)$

NOTE: not using the given notation will cost you points.

A complete solution should explicitly

1. Define what are the subproblems **(3 points)**
2. Recognize and solve the base cases **(3 points)**
3. Write down the recurrence that relates subproblems **(8 points)**
4. Write the pseudocode (always using DP) that returns MIN(n) **(10 points)**
5. State and justify the complexity of your pseudocode **(6 points)**