Programming Project 2025-26

Version 1 - November 14, 2025

This project requires you to implement and test an algorithm to compute the Euclidean Minimum Spanning Tree (EMST) of a set of points in the plane.

1 Project description

Let $\mathcal{D} = \{p_1, p_2, \ldots, p_n\}$ be a set of points $p_i = (x_i, y_i) \in \mathbb{R}^2$ in the plane. The Euclidean distance $\text{dist}(p_i, p_j)$ between a pair of points p_i and p_j is defined as

$$\text{dist}(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

For convenience, we assume that the points have **integer coordinates in** $[0, n-1]$, and that **there are no duplicate points** (i.e., two points with the same coordinates). A *spanning tree* for $\mathcal{D}$ is a tree connecting the n points of $\mathcal{D}$, and it can be represented as a set E of $n-1$ *edges*, where each edge is a pair of points (p_i, p_j)[1]. For each edge $e = (p_i, p_j) \in E$, its *weight* $w(e)$ is defined as the Euclidean distance between p_i and p_j, namely $w(e) = \text{dist}(p_i, p_j)$. The weight $w(E)$ of the spanning tree E is defined as the sum of the weights of all its edges:

$$w(E) = \sum_{e \in E} w(e).$$

The project requires you to design an efficient algorithm to compute a spanning tree E for $\mathcal{D}$ with *minimum weight*, subject to the constraint that each edge e must have a weight $w(e) \leq \alpha$, where $\alpha > 1$ is a real constant. We refer to such a spanning tree as an *α-Euclidean Minimum Spanning Tree* (or α-EMST for short). An example of α-EMST is depicted in Figure 1.

Motivating application. Suppose that $\mathcal{D}$ represents the locations of the houses in a city. The goal is to install a new fiber cable infrastructure to connect all houses, while minimizing the total cost (proportional to the total amount of cable used). Moreover, it is required that each link is not too long to avoid signal degradation. The EMST problem can be employed to determine the topology of the infrastructure.

There are several algorithms that solve the EMST problem without the α-constraint (see [1, Sec.14.7]). The following algorithm solves the α-EMST problem and is a straightforward adaptation of the popular Prim's MST strategy.

[1] If the tree is rooted, an edge represents a parent-child relation.

Algorithm based on Prim's strategy:

Input: Set $\mathcal{D}$ of n points in $\mathbb{R}^2$ and parameter $\alpha > 1$

Output: α-EMST E for $\mathcal{D}$, or FAIL is no such tree exist

Let p be an arbitrary point of $\mathcal{D}$;
$V \leftarrow \{p\}$; $E \leftarrow \emptyset$;
for $n-1$ times **do**
    Find the pair $e = (p_1, p_2)$ with $p_1 \in V$ and $p_2 \in \mathcal{D} \setminus V$, with minimum weight;
    if $w(e) \leq \alpha$ **then**
        Add e to E;
        Add p_2 to V;
    else return FAIL;

You must write, test, and submit a Java program that solves the α-EMST problem, **implementing the above strategy**. Note that a naïve implementation might require $\Theta(n^3)$ time, which can be improved to $\Theta(n^2)$ time by maintaining, at each iteration, the distance of each of the points $\mathcal{D} \setminus V$ from its closest point in V. A more clever implementation that uses a priority queue for the points of $\mathcal{D} \setminus V$ can be devised, attaining $O(n \log n)$ time when α is constant.

2 Input-output formats

Your program must comply with the following input-ouput formatting instructions.

- (INPUT) It receives in input the following command-line arguments, in this order:
  - the path to the file storing the input points;
  - the parameter α (a float).

  In the file the points are stored, one per row, as pairs of integers separated by comma (','').
- (OUTPUT) If no EMST with edges of weight at most α was found, the program must print FAIL. Otherwise it prints:
  - The weight $w(E)$ of the computed EMST E;
  - (only if $n \leq 10$) the list of edges of E, one edge per line. An edge $e = (p_i, p_j)$, with $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$, must be printed as follows: $(x_i, y_i)(x_j, y_j)$.

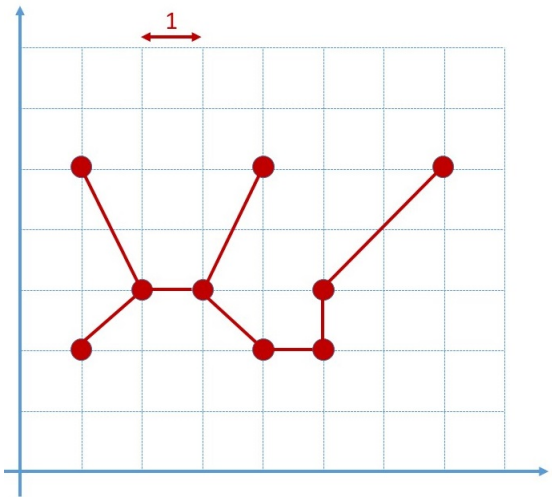

Figure 1: Example of α-EMST with $n = 9$ points and $\alpha = 3$

3 Deliverables

You must submit **only one file** `EMST.java` containing your program, using the dedicated link in the course Moodle. Your code cannot call library functions other than those in `java.lang`, `java.io`, and `java.util` packages.

In Moodle we will later upload some input files, each paired with a corresponding output file that shows the expected output for that input. You can use these files to test the correctness of your code and to make sure that it complies with the specified output format.

IMPORTANT: do not change the name of the Java file, and make sure that your program compiles and runs (with the output formatted exactly as in the given examples) using the `javac`, `java` commands, otherwise, we may not be able to evaluate your submission.

4 Grading

Your project will receive **1 point** if the program correctly solves the problem. We will test your solutions on a set of small instances to check the correctness. Your project will receive **2 points** if, besides being correct, the program is also efficient for constant α (i.e., can solve a large instance in a reasonable time).

References

[1] M.T. Goodrich, R. Tamassia, M.H. Goldwasser. *Data Structures and Algorithms in Java*, Sixth Edition. John Wiley and Sons, 2014.