Delaunay triangulation and Voronoi diagram

## Overview

### Problem

* The Delaunay triangulation and Voronoi diagram problems can be understand and solved easily using geometric theorems
* However, implementing a solution of the problem using these theorems is extremely costly on processing power.
* Due to the performance of the algorithm, solving complex cases of the problem such as when all points are collinear becomes extremely costly on processing power leaving these problems virtually unsolvable without extremely powerful machines.
* The solving of these problems can be greatly optimized with the implementation of data structures such as the “quad-edge” data structure.

### Goals

* Solving the Delaunay triangulation and Voronoi diagram problems the traditional way and analyzing the algorithm’s performance.
* Solving the problems topologically by implementing the “quad-edge” data structure, analyzing the algorithm, and comparing it with the traditional geometrical way.
* Showing how the implementation of these data structure also helps in finding the minimum spanning tree created by random points along a finite space.
* Showing how the dramatic increase in the performance in the algorithm allows for the creation of an application which can solve the problem dynamically.

### Out of scope

* Analyzing other uses of the data structure outside those defined as the goals.
* Comparing the solution with other advanced ways of solving the problems.
* Implementing scientific applications of the Delaunay triangulation and Voronoi diagram problems.

## Context

### Use cases

* Understanding graphically the Delaunay Triangulation and Voronoi problems via an application that lets us visualize randomly generated Voronoi diagrams.
* Showing the Delaunay Triangulation, Voronoi Diagram, and minimum spanning tree of a set of Voronoi cells in a dynamically generated Voronoi Diagram in which the user can move the cells to change the diagram.

### Assumptions

* Will test only cases of the Delaunay Triangulation and Voronoi Diagram problem:
  + When the cells are defined by random points within the space in a 1000 by 1000 canvas
  + When de canvas is increased in size to 4000 by 4000 canvas
  + When all the points are collinear
* The best algorithm will be the topological algorithm when it comes to the processing power, however due to the complexity of the code it may occupy more memory however this increase in memory demands will be inconsequential compared to the gain in performance.
* The algorithmic time of the traditional algorithm will be
* The algorithmic time of the “Quad-edge” implementation will be

## Proposal

This is a java application which will allow the user to create and visualize a Voronoi diagram of the desired size, visualize its Delaunay triangulation, and the minimum spanning tree created by connecting all the Voronoi points.

### Interface

The project will have a simple graphical interface in which the user will be able to specify the size of the Voronoi diagram desired, and after it has been generated 2 buttons that the user can toggle to see the Delaunay triangulation and the minimum spanning tree. The user will also be able to click and drag on any of the points in the Diagram to move them generating a new diagram, the results of this movement will be shown dynamically.

### Data structures

The object of interest in this project is analyzing how implementing the “quad-edge” data structure improves the algorithm performance to solve the problems. Other additional data structures that need to be implemented to create this application include:

* Arrays
* Circular linked lists
* Minimum spanning trees

#### The Quad-Edge data structure

The quad-edge data structure is a natural computer implementation of the corresponding edge algebra. An edge e is represented in the data structure by one edge record e, divided into four parts e[0], e[1], e[2], e[3]. Part e[r] corresponds to the edg , where is an arbitrary canonical representation with specified direction and orientation of the edge.

Each part e[r] of an edge record contains two fields, Data and Next. The Data field is used to hold geometrical and other non-topological information about the edge . This field neither affects nor is affected by topological operations, therefore its contents and format are entirely dependent on the application. The Next field of e[r] contains pointers to the edge e[r]Onext; its four variants correspond to eOnext, eRot Onext, eRot2 Onext and eRot3 Onext.

### Metrics and reports

The main metrics in which we are interested in this project is the processing power needed to run the program in each of the two versions. Additionally, Stack calls and memory usage will be tracked but this will not be of major importance to our analysis of the algorithms.

## Timeline

Delivery 1: spec

Delivery 2: implementation of the geometric way of solving the problem with the traditional algorithm and performance analysis

Delivery 3: implementation of the “quad-tree” data structure to solve the problem, visualization of the Delaunay triangulation and the minimum spanning tree and its performance analysis

Delivery 4: completed java program with graphical user interface and dynamically generated diagrams.