MCS 481 Project Two : polygon partitioning, range trees, and Voronoi diagrams
due Wednesday 12 March 2025 at 2pm

The goal of this project is to use open source software to explore properties of algorithms we have seen to partition polygons, to work with range trees, and to compute Voronoi diagrams.

0. Software

CGAL is well documented and it is recommended to start working from the examples that come with the installation of the library. For this project you may either use the C++ version, or the Python bindings generated via swig, known as `cgal-swig-bindings`. For the Voronoi portion of the project, the `spatial` package of `scipy` is also relevant.

1. Polygon Partitioning

In chapter 3, we considered partitioning polygons. The directory `/examples/Partition_2` from the CGAL source code distribution contains four sample programs of different partition algorithms. The programs show how to call the algorithms and do validity checks on the output.

We have seen a sweep method to produce a y-monotone partition of a polygon. This method is implemented by the CGAL algorithm `y_monotone_partition_2`. The purpose of the first assignment is to verify whether the cost of $O(n \log(n))$ holds in practice for random inputs.

CGAL provides an algorithm for an optimal convex partition of a polygon requiring time $O(n^4)$ and space $O(n^3)$. Compare the performance of `optimal_convex_partition_2` with the other routines, record the number of pieces in the computed partitions along with the running times.

Can you find configurations for which the differences in performance are large?

2. Range Trees

In the directory `/examples/RangeSegmentTrees` of the source code distribution of CGAL, we find examples of 1-, 2-, 3-, and 4-dimensional range trees. In class we introduced range trees to reduce the query time of kd-trees.

For a one dimensional range tree, investigate with random data of size n the $O(n \log(n))$ construction and $O(k + \log(n)$ query time, where k is the size of the output. Make sure the values of n and k are large enough to observe significant differences.

For a two dimensional range tree, examine whether the distribution of the data matters. Compare the performance on random points drawn from a mix of discrete and continuous data, e.g.: from $[a, b] \times \{0, 1, \ldots, r\}$ and $\{0, 1, \ldots, r\} \times [a, b]$, where a and b are real numbers and r is some small natural number. Compare the performance of the queries when x comes from a discrete range (from the set $\{0, 1, \ldots, r\}$) and y comes from the interval $[a, b]$ with the opposite case.

3. Voronoi Diagrams

In the directory `/examples/Voronoi_diagram_2` from the CGAL source code distribution is the sample program `vd2_point_location.cpp`. Consider the following questions.

1. For n sites, we have shown that it takes $O(n \log(n))$ operations to construct a Voronoi diagram. Verify whether the implementation corresponds to this complexity, for sites in general position. For reliable timings, use many different and sufficiently large values for n.

2. For sites in general position, the degree of each vertex of a Voronoi diagram equals three. Examine the behavior of the implementation when all sites occur on the same circle. Is there a difference when the sites are almost on a circle?

4. the Deadline is Wednesday 12 March 2025 at 2pm

Upload your answers to gradescope before 2pm on Wednesday 12 March 2025. Your solution consists of

1. The code for the programs you used in the experiments.
2. Tables with properly formatting of the essential characteristics for each experiment: the dimension of input and output, and the running time.
3. Your interpretation of the results and conclusions.

You may work in pairs for this project.

If you work in a pair, then one member of the pair must email the name of the partner to `janv@uic.edu` before 5pm on Wednesday 5 March.

If you have questions or difficulties with the project, feel free to ask questions during office hour.