Closest Pair Report

Bjarke Brodin (bjal), Bjørnar Haugstad Jåtten (bjja), Helle Friis (hefr) and Simon Boye Jørgensen (sboj)

September 16, 2022

Problem

Given a list of points (x_i, y_i) in two-dimensional euclidean space, find a pair with the minimal distance between the two points of any pair.

Algorithm

We solve the problem using the divide-and-conquer algorithm exactly as described by Kleinberg and Tardos in [1, ch. 5.4], a short outline of the algorithm as they describe it is given below.

We first sort all the points by their x coordinate once, then we apply a mergesort-like recursion, described as follows.

Given a list of points P, if $|P|4$ then find the minimal distance by brute force, this is the bottom of our recursion. if $|P| \geq 4$ however, then we partition the list in halves using the median x coordinate, m_x and recursively obtain the minimum (pair) distance of each sublist min_{left}, min_{right}. Having obtained the local solutions for each list, we then apply a merge step.

In order to merge we need to consider the pairs thant have a point in each partition, which we can do by examining just the sublist $(x_i, y_i) \in P$ where $m_x + \delta \leq x_i \leq m_x + \delta$ with $\delta = min(min_{left}, min_{right})$. Since this list has points $O(n)$, we sort it by y coordinates, and then consider only points in proximity in this order. As shown in [1, ch. 5.4] (statement 5.10), we need only consider pairs of points within 15 positions of eachother within this list.

Analysis

We give a brief asymptotic analysis of our implemented algorithm.

Initially we apply a sort, this is done using java standard library `Arrays.sort`, which is just Timsort, and thus has a worst case bound of $O(n \log n)$.

Then for each merge step in the recursion, we again apply a sort of $O(n \log n)$, then a linear scan to compute the distance, this is $O(n)$.

In order to compute the split, we perform two binary searches of $O(\log n)$. Additionally, we copy the sublists needed for the recursion $O(n)$ at each step (this can be optimized away, but we had no need).

This leaves us with $O(n \log n + \log n(n \log n + n + \log n)) = O(n \log n \log n) = O(n \log^2 n)$ since it can easily be shown that mergesort-like recursion does $\log n$ recursive steps, each amounting to n work ($n \log n$ in our case, since we apply a sort of $O(n)$ elements to obtain the y ordering).

Results

We have created a script, run-tsp-files.sh, that generates a diffOutput.txt file that shows the difference between our output (which can be found in the file output.txt), and the given output file from the assignment. As we can see in the diffOutput.txt file, there are some differences, however, these are only related to decimal rounding errors.

References

[1]John Kleinberg and Eva Tardos, Algorithm Design, 1st ed. ,Boston: Addison Wesley, 2006.