Closest Pair Report

Alexander Berg, Thomas Hoffmann Kilbak, Kristoffer Bruun Højelse, Emil Jäpelt, Adam Negaard

September 19, 2022

Results

Our implementation produces the expected results on all test cases.

Implementation details

We begin, before any recursion takes place, by creating two lists, each list containing all points in the input. The first list, P_x, is sorted on the x-coordinates of the points, the second list, P_y, is sorted on the y-coordinates. In each recursive call the lists: Q_x, Q_y, R_x, and R_y are constructed by splitting P_x and P_y like described in section 5.4 of Kleinberg and Tardos, *Algorithm Design*, Pearson 2014. However, the algorithm also maintains an array $inLeft$ of booleans which, for each index i, is *true* if a point $P_x[i]$ has been added to Q_x or R_x. This is necessary in order to ensure that Q_y contain the same points as Q_x (and similarly for R_x and R_y). Otherwise, if P_y is simply split into Q_y and R_y based on the points relative position to s then there is a chance that a point stored in Q_x does not exist in Q_y (and similarly for R_x and R_y). This could potentially cause incorrect results in some cases. It is, however, interesting to note that, for the given test cases, the algorithm creates the correct results even if $inLeft$ is not used. Eventhough the book does not mention this, we assume that the algorithm described uses a similar technique to ensure the same consistency.
For the comparison of points close to s in S_y we inspect 15 points. Because the input is sorted only once, before any recursive call, the running time is $O(n \log n)$ for n points. The program can be run using .NET 6.0 by navigating to the folder containing "Program.cs" and executing the command:

```
dotnet run
```