Closest Pair Report

Marcus Rodan and Julius Barendt

April 28, 2015

Results

Our implementation produces the expected results on all input–output file pairs. The following table shows the closest pairs in the input files `wc-instance-*.txt`. Here n denotes the number of points in the input, and (u, v) denotes a closest pair of points at distance δ.

n	u	v	δ
2	0	1	1
6	2	3	1
14	6	7	1
30	0	1	1
62	30	31	1
126	62	63	1
254	126	127	1
1022	510	511	1
4094	2046	2047	1
16382	8190	8191	1
65534	32766	32767	1

Implementation details

First we load all the points from the given inputfile. After the points is loaded we sort them by X-coordinate which is a $O(n \log n)$ operation. After the points is sorted we copy them to a new vector called yPoints. The vector yPoints will at the termination of the algorithm contain the points sorted by y-coordinate. We then make a recursive call with the window $[0, \ldots, n-1]$.

The recursive call is as follow(in some pseudo-Java)

```
compute(xPoints, yPoints, xlb, xrb)
if [xlb,...,xrb] is of size 1
return distance between two points
and sort them in vector yPoints.

Calculate middle(M) in [xlb,...xrb]
```

$\alpha \;=\; \textit{Make recursive call with window } [xlb, ..., M]$

$\beta \;=\; \textit{Make recursive call with window } [M+1, ..., xrb]$

`Let` $\delta \;=\; min(\alpha,\;\beta)$

```
Merge the separate sorted vectors yPoints[xlb,..,M]
and yPoints[M+1,...,xrb] into yPoints[xlb,...,xrb].
```

`Choose the points(`$\aleph$`)` *which lies in yPoints[xlb, ..., xrb]*
`and is whoose x-coordinate at maximum differs from`
`the middle lines x-coordinate with` δ.

$\gamma \;=\;$ *Make a careful bruteforce on points* $\aleph$.
`return min(`δ, γ`)`.

We claim that our running time is $O(n \log n)$ for n points. Since we are splitting the vector in halves each iteration we are calling the compute $O(\log n)$ times. The merge step in the recursion is a $O(n)$ operation. Likewise both the choosing of points $\aleph$ and the careful brute force is $O(n)$ operations. The careful brute force is $O(n)$ operation because $\forall \rho \in \aleph$ we only need to check the distance to its 16 next neighbours. We therefore have $O(\log n \times (C + n + n + n)) = O(n \log n)$