# *Closest Pair Report*

*Dennis, Thor, Maria, Daniel and Nicoline*

*September 18, 2017*

## *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 $\delta$.

| $n$ | $u$ | $v$ | $\delta$ |
|---|---|---|---|
| 2 | 0 | 1 | 1 |
| 254 | 29 | 16 | 1 |
| 30 | 9 | 11 | 1 |
| 4094 | 3953 | 2802 | 1 |
| 6 | 4 | 0 | 1 |
| 62 | 11 | 24 | 1 |
| 65534 | 35326 | 14974 | 1 |

## *Implementation details*

For the comparison of points close to $s$ in $S_y$, we inspect 15 points[1]:

[1] 5.10, Kleinberg and Tardos, *Algorithm Design*, Addison–Wesley 2008.

```
    for(int i = 0; i < Sy.size(); i++) {
        int interval = Math.min(i+15, Sy.size());

        for(int j = i+1; j < interval && Sy.get(j).getY()-Sy.get(i).getY() < dMin; j++)
        {
            double dist = distance(Sy.get(i),Sy.get(j));
            if(dist < dMin)
            {
                dMin = dist;
                closestPair = new Point[] {Sy.get(i), Sy.get(j)};
            }
        }
    }
```

Our running time is $O(n \log n)$ for $n$ points, since the optimized algorithm is used to avoid sorting the input on each recursive call.

Instead, the input is only sorted once and passed to the recursive closestPairRec() method. During every recursive call, the points (sorted by y-coordinate) are placed in either sublist (Qy , Ry), based on the comparison of their corresponding x-coordinate and the middle split point (x-coordinate).
Here is a snippet of the code:

```
void closestPair() {
     ...
     Collections.sort(pointsByY, Point.getByYComparator());
     ...
 }

 Point[] closestPairRec(List<Point> pointsX, List<Point> pointsY) {
  ...
  //Split on y
  List<Point> Qy = new ArrayList<Point>();
  List<Point> Ry = new ArrayList<Point>();

  // Place points based on comparison with middle split point
  for(int i = 0; i < pointsY.size(); i++) {
      Point point = pointsY.get(i);
      double x = pointsY.get(i).getX();
      if(x < pointsX.get(splitIndex).getX()) {
          Qy.add(point);
      } else {
          Ry.add(point);
      }
     }
    ...
 }
```