Closest Pair Report

Mark Dear, Stelios Papaoikonomou, Radek Niemczyk and Rasmus Løbner Christensen

September 16, 2015

Results

Our implementation produces the expected results on all input files. Comparing with the given output file, we get the same values for all closest pair of points, on each input file.

Implementation details

The pseudo-code states the following (p. 136 of Kleinberg and Tardos, *Algorithm Design*, Addison–Wesley 2008.):

a) Construct Px and Py in O(n log n) time
b) Construct Qx, Qy, Rx, Ry in O(n) time
c) Construct Sy in O(n) time
d) For each point in Sy compute distance from 'delta'

While comparing our implementation with the complexity of the actual pseudo algorithm, we will focus on the exact same points (though we have looked through our methods, e.g. to check for unnecessary loops etc.)
Therefore, we will repeat the points from above, and hereafter argue for the complexity of our own implementation.

a) Construct Px and Py in O(n log n) time
The class, Parser, does this. It takes the output from the FileReader class (which reads a file via the Java File library), and iterate through each line in the given file. The iteration is done via a for loop. For each line we use the trim() method, and then checks whether the trimmed string begins with either EOF or a digit. If the line begins with EOF, we stop looking at the lines. Otherwise, if the trimmed string contains a digit, we split this line on widespace (one or more widespaces). Hereafter parsing the values as doubles, and add these to Px and Py. Lastly we sort Px and Py by x/y-coordinate, by using java's Comparator class. We assume all the above (except the sorting), to run in constant time, hence we get the complexity of: O(k + (n log n)) => O(n log n) due to Comparator's complexity of O(n log n).

b) Construct Qx, Qy, Rx, Ry in O(n) time

Our implementation uses Java's built in List.Sublist method. Looking at the documentation this is guaranteed to run in O(1) for ArrayLists. The method doing the splitting of lists, is called 'constructQR'. But since we assume that the worst case require us to split n lists, we get the run-time of O(n).

c) Construct Sy in O(n) time

The current implementation runs through Px (which is of size n), and checks whether each Point lies within 'delta' of L. During this it creates a list, and lastly this list gets sorted by y-coordinate. The running time of this operation should be O(n + n log n). This is above the mentioned O(n) time.

d) For each point in Sy compute distance from 'delta'

In order to do this, our implementation goes through each element in Sy (which worst-case is of size n), and then checks for the 15 (if they exists) next points in Sy, whether these constitute a smaller pair with the current element. The complexity is therefore O(n + k), since we assume the for loop over 15 elements to be a constant, which is the same as O(n).

Total runtime is therefore: O(n log n + n + n + n log n + n) => O(2(n log n)) (???)