Closest Pair Report

Sigurt Dinesen, Gustav Røder & Lars Yndal Sørensen

September 7, 2015

Results

Our implementation produces the correct result, for all of the provided input/output examples (`closest-pair-out.txt`). The calculated results, and given example outputs, are compared when the program is run without parameters.

Implementation

The implementation follows the pseudo code from page 230 in *Kleinberg and Tardos, Algorithm Design Addison–Wesley 2005* closely.

For input sizes $n \leq 3$, we run a naive, pairwise $O(n^2)$ comparison algorithm. For larger inputs, we recursively divide the plane into two parts, each with $n/2 \pm 1$ points and find the closest point-pair in the left half, the right half and the pairs who's connecting edges cross the partition – returning the closest of the three pairs. The "crossing" pair is found using the trick from the book, taking into account the minimum distance of the "left" and "right" pairs, allowing it to be done in linear time.

The recursive function has a cutoff to the $O(n^2)$ solution, as mentioned above. It should be noted that $O(n^2)$, with the restriction that $n <= 3$, is considered to be constant time.

Performance

The running time is $n \log(n)$. This is illustrated in the following, by use of the `master theorem`.

The running time is an instance of the equation

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Which means we can analyse it using the `master theorem`: As we recursively divide the input into two parts of $O(n/2)$ size, we have $a = b = 2$ in the above equation, which gives us

$$k = \log_b(a) = \log_2(2) = 1$$

At each level of recursion:

- The four lists (named `Q_x, Q_y, R_x, and R_y` in both the book and our implementation) containing the points of each partition, in increasing order by the x and y coordinate respectively, are constructed at $O(n)$ cost.

- The set of points within δ distance of the partitioning line is found at $O(n)$ cost.
- Each point in that set is compared to the 15 following points, at $O(n * 15) = O(n)$ cost.

In summary: $f(n) \in O(n)$

Returning to the `master theorem`, we observe that as $n = n^1 = n^k$:

$$f(n) \in O\left(n^k \log^0(n)\right)$$

which, according to the `master theorem`, means that the recursive algorithm runs in

$$O(n^k \log^{0+1}(n)) = O(n \log(n))$$

Before the recursion, the input is ordered by both x and y coordinates, giving $O(n \log(n))$ preprocessing cost, which leaves the total running time as

$$O\left(n \log(n)\right)$$