Nisha Masharani (nisham)

CS161 Homework 3 Problem 3-1

Algorithm

The algorithm remains the same as described in lecture 6 and on page 1040 of CLRS except for the combination step. To combine the results, let a_L, b_L, c_L be the closest three points on the left side, and let a_R, b_R, c_R be the closest three points on the right side. Let $d = \min_{i \in \{L,R\}} a_i b_i + b_i c_i + a_i c_i$. If there are three points that straddle L that are closer together, they must all be at most $\frac{d}{2}$ away from each other, meaning they must be within a band of $\frac{d}{2}$ around L. Looking only at points in this band, sort the points by y-coordinate, then place a grid with side length $\frac{d}{4}$ onto the band. For each point $p = (x, y)$, look at the 15 points $p_i = (x_i, y_i), 1 \leq i \leq 15$ above p (closest to p with $y_i \geq y$). For each pair of points p_i, p_j that are within $\frac{d}{2}$ of each other, repeat the process on the point in the pair with the smaller y value to find a third point p_k within $\frac{d}{2}$ of both p_i and p_j. Keep track of the minimum $d' = p_i p_j + p_i p_k + p_j p_k$. If, at the end, $d' < d$, the three closest points are the ones that yielded the perimeter d'.

Correctness

The proof of correctness is the same except for the proof that we only need to check 15 points above point p_i to find all of the points in the band within distance $\frac{d}{2}$ of p_i. Without loss of generality, assume the three closest points are $a, b \in L$ and $c \in R$. That means that the distance $d' = ab + bc + ca < d$. If we look at the geometry of a triangle, $\max\{ab, bc, ca\} < \frac{d'}{2} < \frac{d}{2}$. To see this, let $ab = \frac{d'}{2}$. In this case, since $d' = ab + bc + ca$, we get that $bc + ca = \frac{d'}{2}$, so c must be a point on the line from a to b. Therefore, for the three points to form a triangle of perimeter d', the maximum distance between the three points must be less than $\frac{d'}{2} < \frac{d}{2}$. Thus, we know that these three points must be within a $\frac{d}{2}$ band around L, and the maximum distance along the y-axis between the three points must also be $\frac{d}{2}$. Thus, we can say that the points are within a $\frac{d}{2} \times d$ rectangle around L.
Next, we must show that at most 16 points can reside within this rectangle. For a triangle, if two of the points (which we will call a and b without loss of generality) are closer together than $\frac{\sqrt{2}d}{4}$, which is the maximum distance that will fit into a square of side length $\frac{d}{4}$, then the third point c must be at least $\frac{(2-\sqrt{2})d}{4}$ away from one of the points and at least $\frac{d}{2}$ away from the other point(in the case that the triangle is a straight line) and at least $\frac{(4-sqrt2)d}{8}$ away from both of the points (in the case that the triangle is an isosceles triangle), which means that the altitude from c to ab is at least $\frac{\sqrt{4-2\sqrt{2}}d}{4}$. In all of these cases, the distance from c to ab is ¿ $\frac{d}{4}$, so there can be at most two points per square if two of the points are closer than $\frac{\sqrt{2}d}{4}$. If the minimum distance between two points in the triangle is greater than $\frac{\sqrt{2}d}{4}$, that means that two of the points cannot fit in the same square, so there must be at most two points per square.

In both of these cases, we get that there are at most two points per square with side length $\frac{d}{4}$. Thus, in a rectangle of side lengths $\frac{d}{2} \times d$, there are at most $2 \times 2 \times 4 = 16$ points. Having shown that at most 16 points can reside within the rectangle, since all three of the closest points must be within that rectangle, we must only check the 15 points above the lowest point to find all three points.

Running Time

Similarly to the original closest point algorithm, we can describe the running time of our algorithm with the recurrence $T(n) = 2T(n/2) + \Theta(n)$, after pre-sorting the points by both x and y coordinate. To see this, we note that our recursive step, like in the original closest point algorithm, divides the input in half, so the time to run the recursive step is $T(n/2)$ for each half. We also can see that the combine step runs in $\Theta(n)$ time by looking at what happens to an individual point. For each point in the $\frac{d}{2}$ band around L, which is at most n points, we do 15 comparisons. Then, for each of those 15 comparisons, we do at most 15 more comparisons to find the third point. Thus, we do at most $15^2 = 225$ comparisons for each point, meaning that our running time for combining the results is $225n = \Theta(n)$.
This recurrence is in the correct form for the master method, with $a = 2$, $b = 2$, and $f(n) = cn$. $n^{\log_b a} = n^{\log_2 2} = n$. Therefore, we are in the second case of the master method, where $f(n) = \Theta(n \log^0(n))$, so $T(n) = \Theta(n \log n)$. We also must pre-sort the points. If we use an $\Theta(n \log n)$ sorting algorithm such as merge sort or quick sort, since we are doing two sorts, the time to sort is $\Theta(n \log n)$. Thus, the total running time of the algorithm is $\Theta(n \log n) + \Theta(n \log n) = \Theta(n \log n)$.