CS381 Homework 1 Problem 1

Connor Couetil

January 29, 2026

1 Exercise 3.3

I've split up the approach into two functions, the first function calculates the minimum distance across all the points, you have to do that before you calculate the set of points with a factor of that distance, and then it case the recursive function that returns the list of points. The key is making `within-factor-2` the same runtime as `min-distance`, so you can call both of them sequentially and still achieve the same worst-case runtime.

`within-factor-2(`P`)`: Given P, a set of n points in $\mathbb{R}^2$, it returns a list of all pairs of points with a distance a factor of 2 greater than the minimum distance between all points.

```
1. $d =$ min-distance($P$)
2. Return within-distance($P$, $2 * d$)
```

`within-distance(`P`, `d`)`: Given P, a set of n points in $\mathbb{R}^2$, and d, a measure of distance, it returns a list of all pairs of points in P within the distance d.

```
1. If $|P| \leq 1$, return $\{\}$
2. Let $X$ be an array that contains all $x$ of $(x, y) \in P$
3. $a =$ median($X$) // O(n)
4. $P_1 = \{(x, y) \in P : x < a\}$ // O(n)
5. $P_2 = \{(x, y) \in P : x \geq a\}$ // O(n)
6. $d_1 =$ within-distance($P_1$, $d$)
7. $d_2 =$ within-distance($P_2$, $d$)
8. Return $\{p \in P : |p_i - p_j| < d\}$ over all $i, j$ with $i < j < i + 22$ // O(1)
```

The recurrence relation for the algorithm `within-distance` is

$$T(n) = O(n) + O(n) + 2T(n/2) + O(1)$$

The calculation of the median is in $O(n)$, as stated in the textbook. Separation of the points by median into two sets can occur in a single pass, $O(n)$. The recursion occurs on sets half the size of the starting set, by the defintion of splitting points along some median.

Now, the final calculation of distance between pairs is bounded by a constant value 22, thus the work done is also constant, or $O(1)$. The argument is as follows.

The last step of the algorithm checks the set of points next to the median within the distance d that is the factor 2 of the of the minimum distance, henceforth 2δ. So for a given point p, we only need to check within 2δ on its own side of the median line and 2δ on the other side. We split that region in two, and consider one half. Each point in the half is still at least δ from any other. Drawing a circle with that point at the center, the maximum points you can fit in the square will lie within a padded square whose dimensions are 3δ by 3δ, its width is the 2δ search space distance plus twice the $\delta/2$ radius of the packed circles. So, the maximum number of circles that can fit in the padded square are the maximum number within our search distance, thus the maximum number we need to check. That number is the ratio of the area of the padded square and the point circle

$$\frac{(2\delta + 2(\frac{\delta}{2}))^2}{\pi(\frac{\delta}{2})^2} = \frac{(3\delta)^2}{\pi(\frac{\delta^2}{4})} = \frac{36}{\pi} = 11.45... \approx 11$$

There are no fractional points, so there are 11 points to check in the square, there are two squares so 22 points total to check, the value of the constant in the recursion.

To take this one step further, consider the factor of 2 as some factor F. The side of the padded square is then $F\delta + \delta$ and the final ratio becomes

$$\frac{F^2 + 2F + 1}{\pi}$$

and we realize that as you increase the factor of the minimum distance to search the number of points to check increases quadratically.