Algorithm Design and Analysis (CS3383)-Tutorial **(Jan/30/2019)**

Rasoul Shahsavarifar

Find the closest pair in a set of planar (2D) points:
We apply the divide and conquer method in order to compute the closest pair of points in a given planar (2D) dataset S. The divide and conquer algorithm to solve this problem can be achieved by following the given steps bellow:

1. Presort the elements based on their $X-$coordinates.
2. Divide the sorted set into two equal sized subsets using a line ℓ.
3. Recursively compute the closest pairs in each subset. Suppose that d_1 and d_2 are the distances corresponding to these two closest pairs.
4. Define $d = \min\{d_1, d_2\}$, and store the pair corresponding to d.
5. Consider a strip around ℓ with the width of $2d$ and the boundaries paralleled with ℓ.
6. Ignore all points that are not within the strip defined in step 5.
7. Sort the remaining points in step 6 based on their $Y-$coordinates.
8. Scan the points from step 7 in the Y order.
9. For each scanned point in step 8, compute its distances from its neighbors within (or, on the boundary of) a box $(2d \times d)$ (see Figure 1). Note that at most 5 points are qualified to be inside (or, on the boundaries) of this box. Otherwise, being minimum for d will be contradicted.
10. If the distance between any pair of points within the strip P is less than d, update the stored pair from step 4.

Analysis: The merging cost is affected by the sorting in steps 7. This means that the recurrence relation would be as follows:

$$T(n) = 2T(n/2) + O(n \log n).$$

Solving this recurrence using the recursion tree method implies that $T(n) = O(n \log^2 n)$. Do you know how?

Improvement: The merging cost can be improved by slightly changing the step 3 as follows:
Modified Step 3: Recursively compute the closest pairs in each subset, and return the points in each set in sorted order based on their $Y-$coordinate.

New Analysis: The merging cost in the modified algorithm is just $O(n)$ because in this version, we need to merge two sorted array. So the recurrence would be as follows:

$$T(n) = 2T(n/2) + O(n) \tag{1}$$

By applying the Master theorem, $T(n) = O(n \log n)$.

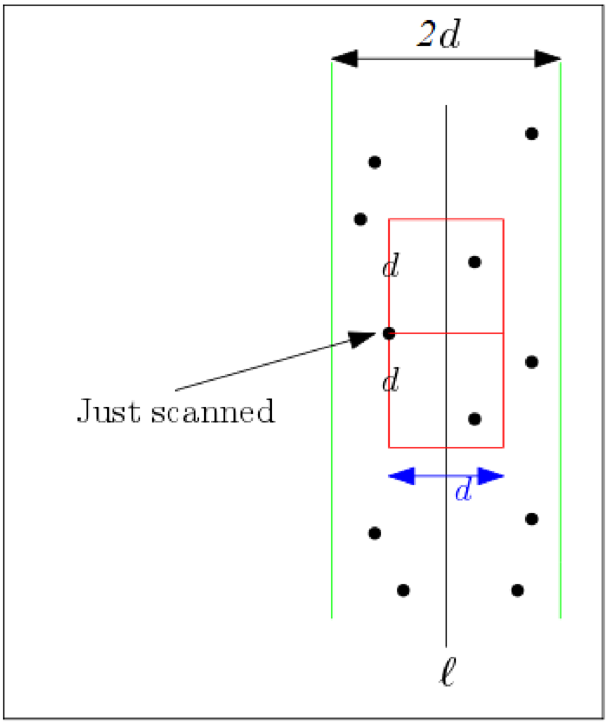

Figure 1: Strip P with the width of $2d$ around ℓ