Finding the closest pair of points in 1-dimension

Problem statement:[1]

Given a set S of n points on a line (unsorted), find two points whose distance is smallest.

Bruce-force:

- Pick two of n points and compute the distance
- Cost: $\Theta(n^2)$

[1] Section 33.4 of [CLRS, 3rd edition] provides an algorithm for finding the closest pair of points in 2-dimension, i.e., on a plane.

Finding the closest pair of points in 1-dimension

Algorithm 1

1. Sort the points
2. Perform a linear scan

Cost: $\Theta(n \lg n) + \Theta(n) = \Theta(n \lg n)$

Remark: Unfortunately, Algorithm 1 cannot be extended to the 2-dimension case.

Finding the closest pair of points in 1-dimension, 3

Algorithm 2 (Divide-and-Conquer):

1. Divide the set S by some point mid (say, median) into two sets S_1 and S_2, with the property:

$$p < q \text{ for all } p \in S_1 \text{ and } q \in S_2$$

2. Conquer: finds the closest pair *recursively* on S_1 and S_2, separately, gives us two pairs of points

$$\{p_1, p_2\} \quad \text{and} \quad \{q_1, q_2\},$$

   the closest pair in S_1 and S_2, respectively. .

3. Combine: the closest pair in the set S is

$$d = \min\{|p_1 - p_2|, |q_1 - q_2|\} \quad \text{or} \quad d' = |p_3 - q_3|,$$

   where $p_3 \in S_1$ and $q_3 \in S_2$.

Finding the closest pair of points in 1-dimension

1. both p_3 and q_3 must be within distance d of mid if $\{p_3, q_3\}$ is to have a distance smaller than d.
2. How many points of S_1 can lie in $(mid - d, mid]$?
   Answer: at most one
3. How many points of S_2 can lie in $[mid, mid + d)$?
   Answer: at most one
4. Therefore, the number of pairwise comparisons that must be made between points in different subsets is thus at most one.
5. We can certainly find the points in the intervals $(mid - d, mid]$ and $[mid, mid + d)$ in linear time $O(n)$.
6. Algorithm 2 complexity:

$$T(n) = 2T(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n).$$

Finding the closest pair of points in 1-dimension

Pseudocode[2]

```
ClosestPair(S)
if |S| = 2, then
   d = |S[2] - S[1]|
else
   if |S| = 1
      d = infty
   else
      m = median(S)
      construct S1 and S2
      d1 = ClosestPair(S1)
      d2 = ClosestPair(S2)
      p = max(S1)
      q = min(S2)
      d = min(d1, d2, q-p)
   end if
end if
return d
```

[2] The idea of Algorithm 2 can be extended to the 2-d case, see Section 33.4 of [CLRS, 3rd edition]