Lec12-closest-points

KT 5.4

Sunday, September 24, 2023 4:28 PM

We just saw an example of dynamic programming, where you can break it to smaller subproblems. Now we're going to consider another simpler problem using divide & conquer

Divide & Conquer

Related to induction.
- base case
- inductive hypo
- induction step

algorithm
- easy small case (maybe brute force)
- alg works on problems of size $< k$
- combine small answers to get big full answer (recursive)

Recall: merge sort

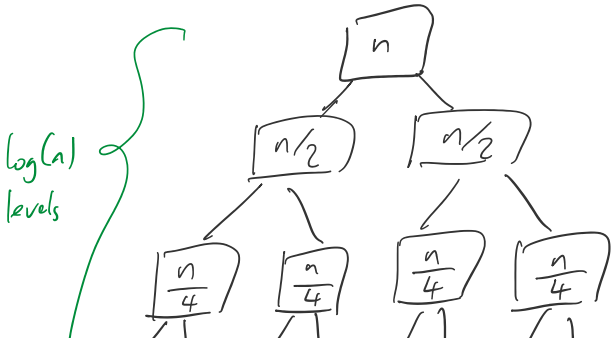
```
MergeSort(L):
    if |L| = 2:                                   (base case)
        return [min(L), max(L)]
    else:
        L1 = MergeSort(L[0, |L|/2])               } smaller size problems
        L2: MergeSort(L[|L|/2, |L|-1])
        return Combine(L1, L2)                    } combine answers
```

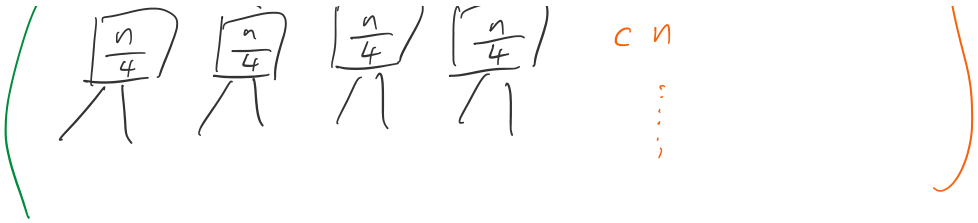
Combine: walk down both lists, taking smaller number each time. $O(n)$ time.

Total time: $T(n) \leq 2T(\frac{n}{2}) + cn$

Solving recurrences

Method 1: Unrolling (into tree)





$\Rightarrow O(n \log n)$.

$T(n) \le 2T(\frac{n}{2}) + cn$

Method 2: Substitution method (by induction)

1. Show $T(k) \le f(k)$ for small k
2. Assume $T(k) \le f(k)$ for all $k < n$.
3. Show $T(n) \le f(n)$.

WTS: $T(k) \le 2ck \log 2k$

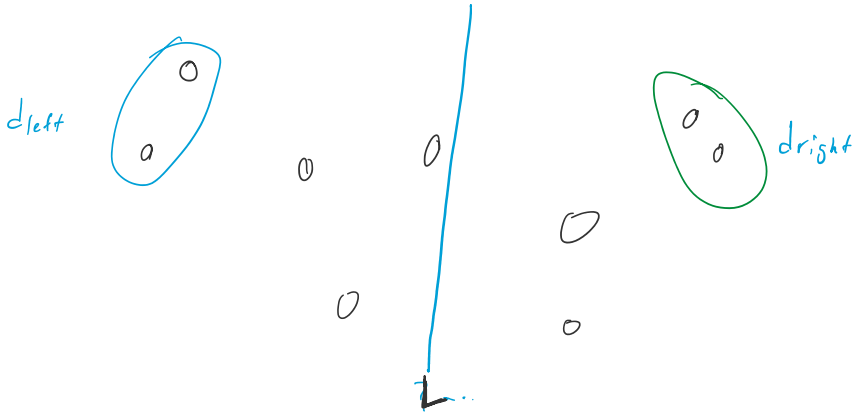
Base case: $2c \log 2 = 2c \ge T(2)$

Induction:

$$\begin{aligned} T(n) &\le 2T(\tfrac{n}{2}) + cn \\ &\le 2c(\tfrac{n}{2})\log(\tfrac{n}{2}) + cn \\ &= cn[\log n - \log 2] + cn \\ &= cn[\log n - 1] + cn \\ &= cn \log n. \end{aligned}$$ ✓

Finding closest set of points:

Problem: Given points $\{p_1, \dots, p_n\} \subseteq \mathbb{R}^2$, find pair (p_i, p_j) that are closest.



Brute force: $O(n^2)$ — check every pair

Can we do faster? Yes. $O(n \log n)$ time.

Divide & Conquer

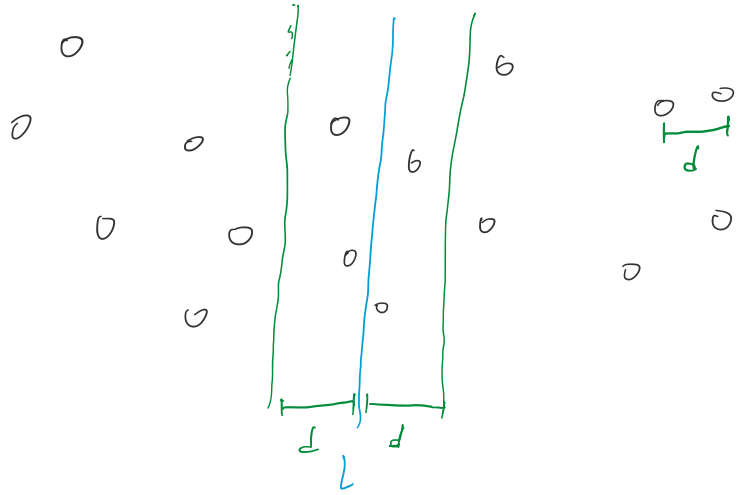
- Split into left & right side

- Recursively find closest pair on each side, d_{left}, d_{right}
- Merge together ← hard case because what if closest pair crosses split?

$d = \min\{d_{left}, d_{right}\}$ ← does not take into account pairs crossing split

Merging left & right

If $\exists$ pair (p_i, p_j) with $dist(p_i, p_j) < d$ that is split by centre line L, then $dist(p_i, L) < d$ & $dist(p_j, L) < d$



Let S_y be an array of points within distance d of L, sorted by decreasing y-coordinate.

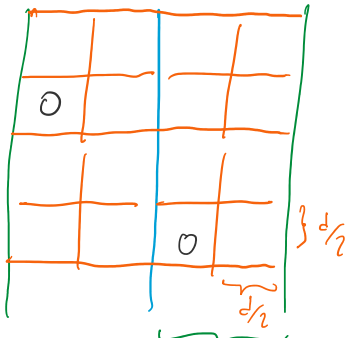
How big is S_y? Could be all points, so can't check all pairs in S_y.

Thm Suppose $S_y = [p_1, \ldots, p_m]$. If $dist(p_i, p_j) < d$, then $j - i \leq 15$.

i.e. if two pts are close in plane, they must be close in S_y array.

proof.

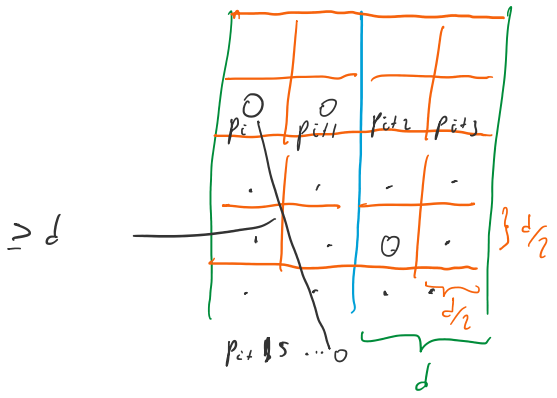
Divide region into squares with side length $d/2$



only 1 pt per box allowed since pts within box closer than d, and each box is on one side.

Suppose $\exists\ p_i, p_j$ with $dist(p_i, p_j) < d$ and $j - i \geq 15$.

Then must be separated by at least 3 rows



Contradiction, because then $\geq d$ dist apart.

Thus, scanning S_y takes linear time because we only need to check $\leq 15n$ pairs.

Also, initially sorting all points by y-coordinate takes $O(n \log n)$ time, but after, filtering to S_y (i.e. dist d from centre) takes n time

Total running time:

$O(\log n)$ depth recursion since dividing in half.

Merge takes $O(n)$ time.

Recurrence $T(n) \leq 2T(n/2) + cn$

$\Rightarrow$ same as Merge Sort $= O(n \log n)$ time.