Lec11-closest-points

KT 5.4

Tuesday, September 17, 2024 12:08 PM

In some of the shortest path algorithms, we were able to cast a hard problem in terms of easier subproblems. Here, we start formalizing these ideas.

Divide & Conquer

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

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

Recall: merge sort

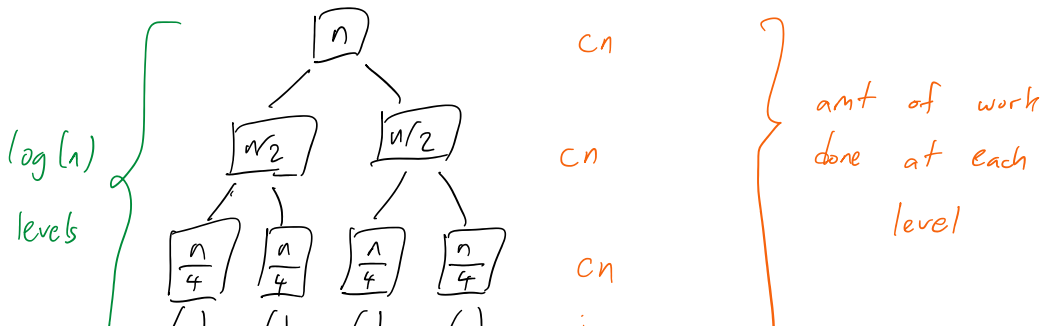
```
MergeSort (L):                                  (simplified for list of size 2^k)
    if |L| = 2:                                 (base case)
        return [min(L), max(L)]
    else:
        L1 = MergeSort(L[0, |L|/2])             } smaller size problems
        L2 = MergeSort(L[|L|/2, |L|-1]) 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

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

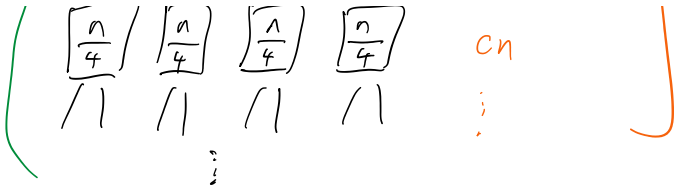
Total time: $T(n) \leq 2T\left(\frac{n}{2}\right) + cn$

Solving recurrences:

Method 1: Unrolling (into tree)



levels



$\Rightarrow O(n \log n)$

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 $T(n) \le f(n)$

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

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

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

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

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

$= cn[\log n] + cn$

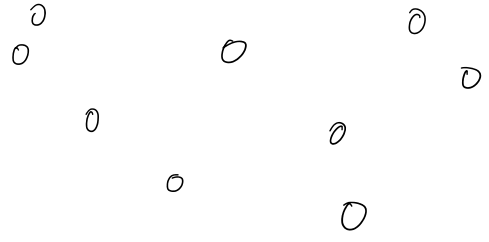
$\le cn[\log n + 1]$

$= cn[\log n + \log 2]$

$= cn[\log 2n] \le 2cn \log 2n$ ✓

Finding closest set of points

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



Brute force: $O(n^2)$ - check every pair of points

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

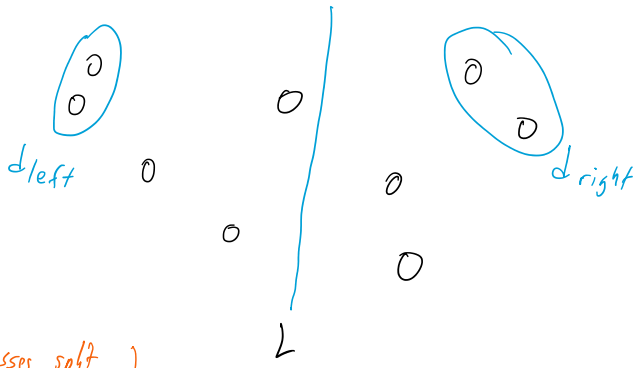
Divide & Conquer

point $(h, \ldots)$

Divide & Conquer

- Split left/right sides (balanced) point
- Recursively find closest pair on each side
  d_{left}, d_{right}
- Merge together

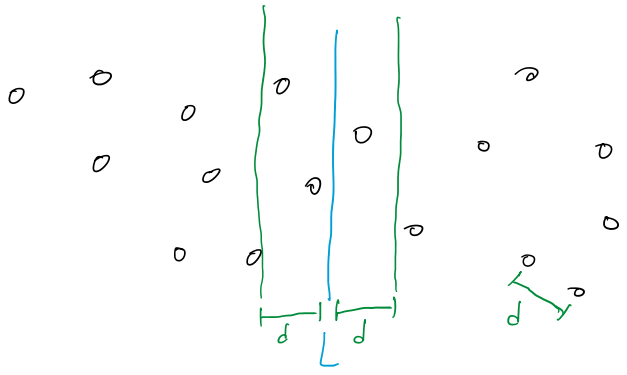
(hard case because what if closest pair crosses split)



Initial Guess $d = \min\{d_{left}, d_{right}\}$ ↗ does not take into account

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 $dist(p_i, L) < d$ & $dist(p_j, L) < d$

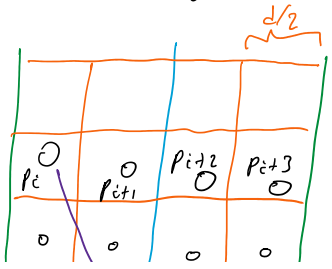


Let S_y be an array of points within distance d of L, sorted by descending y-coordinate

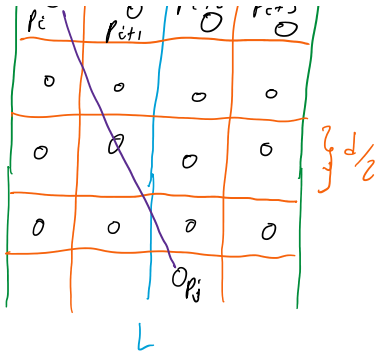
How big is S_y? Could be all points, so can't check all pairs in S_y

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 points are close in $\mathbb{R}^2$, then must be close in S_y array.

proof. Divide region into squares of side-length $d/2$



only 1 pt per box allowed
since pts within box closer than d,
and each box entirely on



Since pts within box closer than d, and each box entirely on one side of L.

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

They must be separated by at least 3 rows

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

Thus, scanning S_y takes linear time since 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 L) 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 MergeSort $= O(n \log n)$ time.