

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:

$L1 = \text{MergeSort}(L[0, \frac{|L|}{2}])$

$L2 = \text{MergeSort}(L[\frac{|L|}{2}, |L|-1])$

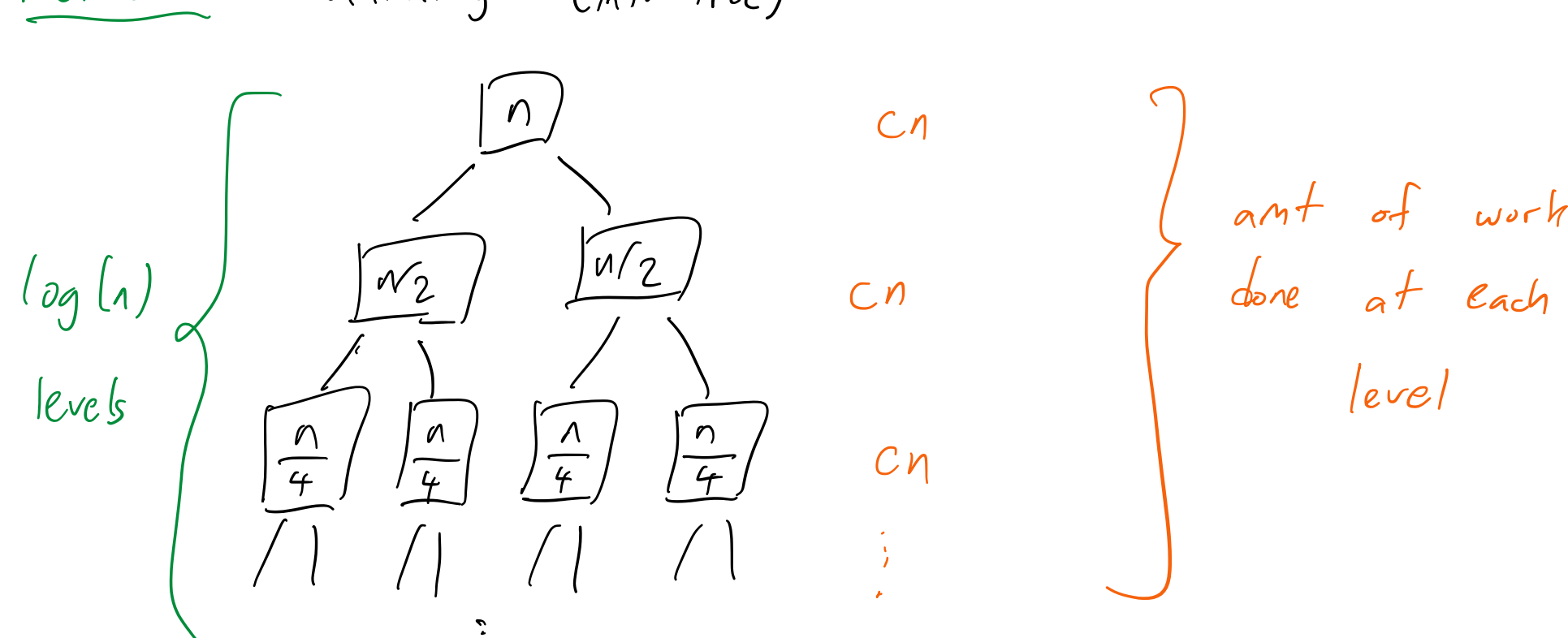
return Combine ($L1, L2$)

$\left. \begin{array}{l} \text{smaller size} \\ \text{problems} \end{array} \right\}$
 $\left. \begin{array}{l} \text{combine answers} \\ \text{walk down both lists, taking smaller number each time} \\ O(n) \text{ time} \end{array} \right\}$

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

Solving recurrences:

Method 1: Unrolling (into tree)



$\Rightarrow O(n \log n)$

Method 2: Substitution method (by induction)

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

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

$$\text{WTS: } T(k) \leq 2ck \log 2k$$

$$\text{Base case: } 2c \log 2 = 2c \geq T(2)$$

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

$$\leq 2c \left(\frac{n}{2}\right) \log 2\left(\frac{n}{2}\right) + cn$$

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

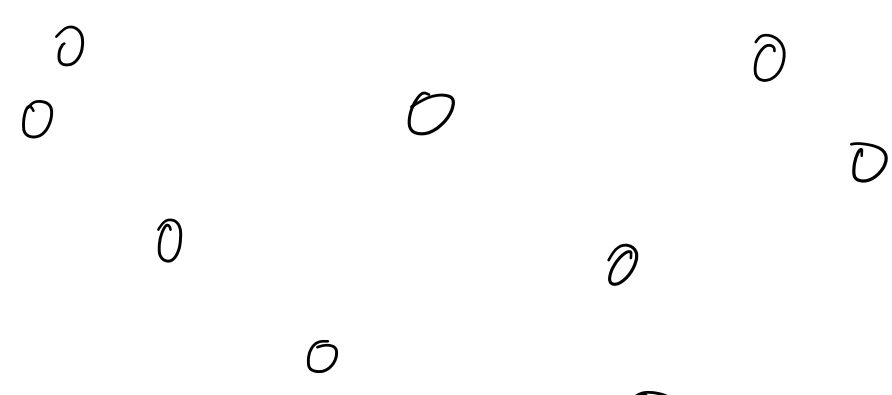
$$\leq cn [\log n + 1]$$

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

$$= cn [\log 2n] \leq 2cn \log 2n \quad \checkmark$$

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 of points

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

Divide & Conquer

- Split left/right sides (balanced)
- Recursively find closest pair on each side

$d_{\text{left}}, d_{\text{right}}$

- Merge together

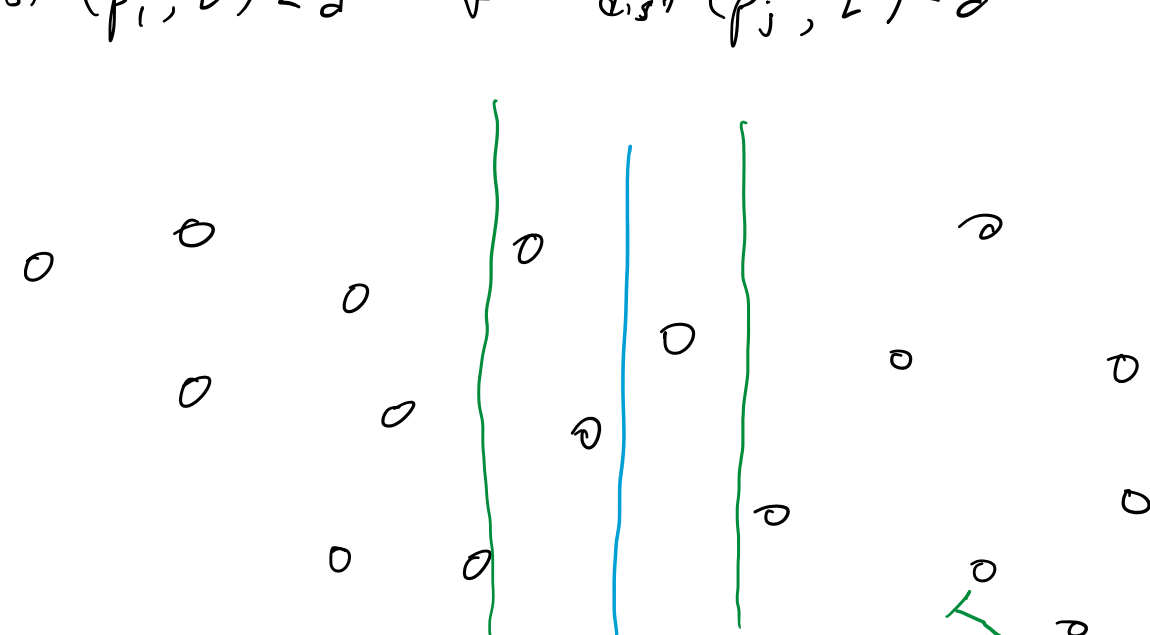
(hard case because what if closest pair crosses split)

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

Merging left & right

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

then $\text{dist}(p_i, L) < d$ & $\text{dist}(p_j, L) < d$



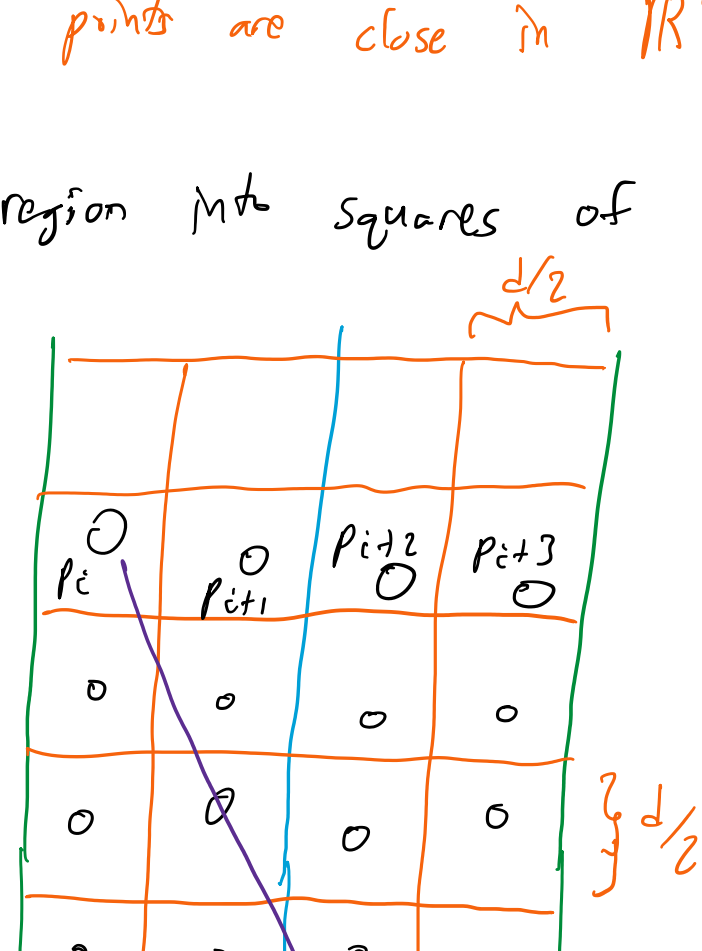
Let S_y be an array of points within distance d of L , sorted by descending y -coordinate

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

Thm: Suppose $S_y = [p_1, \dots, p_n]$. If $\text{dist}(p_i, p_j) < d$, 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
one side of L .

Suppose $\exists p_i, p_j$ with $\text{dist}(p_i, p_j) < d$ 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 $T(n) \leq 2T\left(\frac{n}{2}\right) + cn$

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