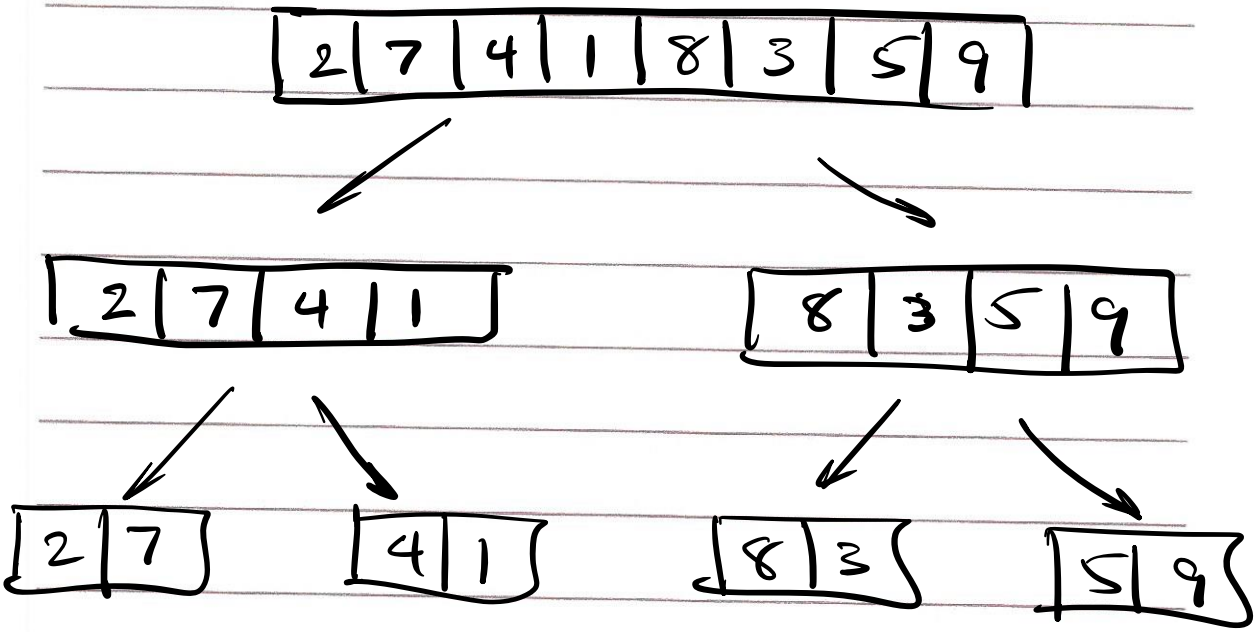
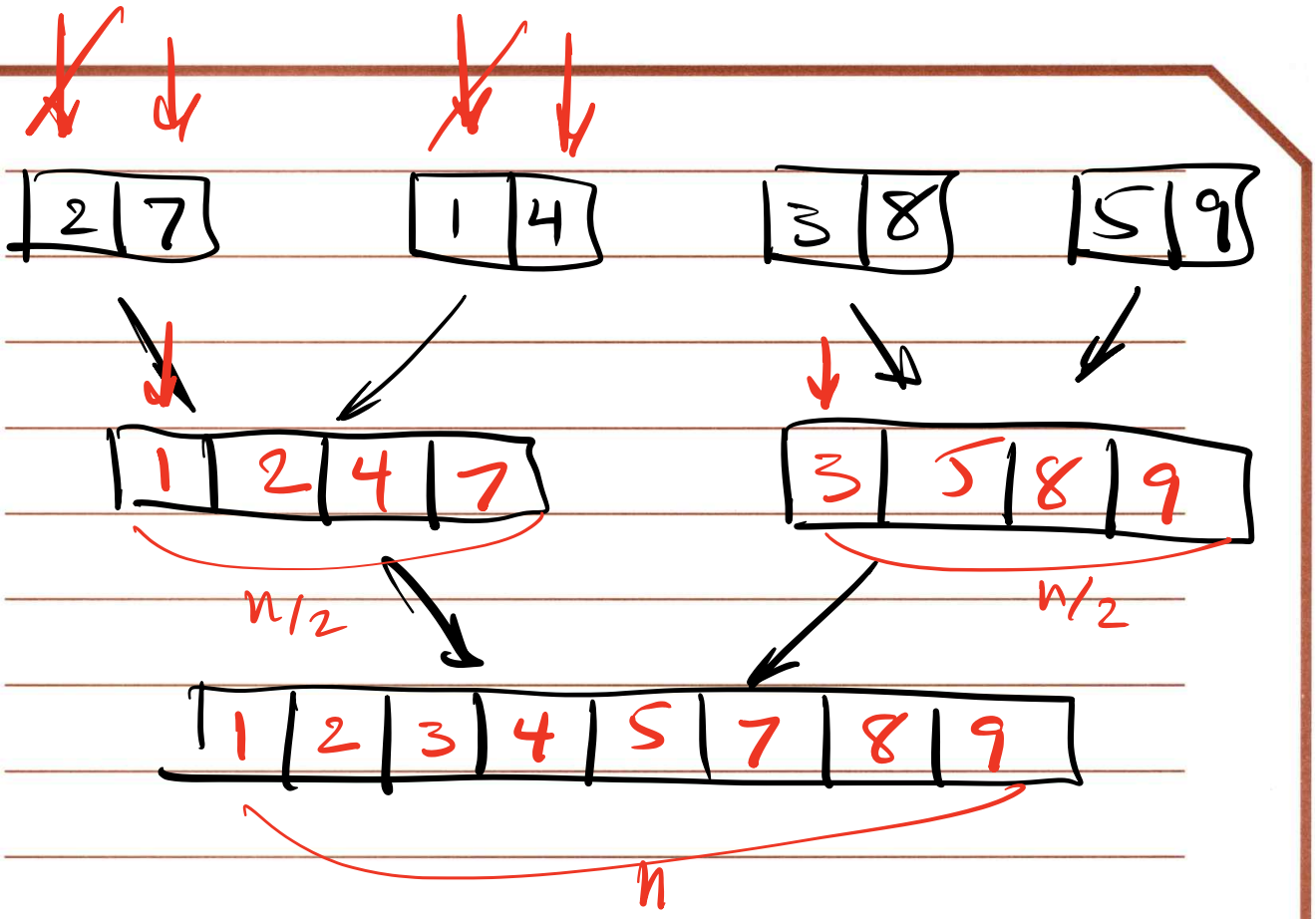
Divide & Conquer

1- Divide problem into n subproblems

2- Conquer: i.e. solve the subproblems recursively, or if trivial solve the problem itself

3- Combine the solution to the subproblems




2 7
1 4
3 8
5 9
1 2 4 7
3 5 8 9
n/2
n/2
1 2 3 4 5 7 8 9
n

```
1  MERGE-SORT(A, p, r)
2      if p < r then
3          q = ⌊(p+r)/2⌋
4          MERGE-SORT(A, p, q)
5          MERGE-SORT(A, q+1, r)
6          MERGE(A, p, q, r)
7      end if
```

Divide

Conquer

Combine

Analyzing Merge-sort

Divide – Takes $O(1)$

Conquer – If the original problem takes $T(n)$ time, the two subproblems take $2 \cdot T(n/2)$

Combine – Takes $O(n)$ on array size of n.

Recurrence equation for merge sort

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2 \cdot T(n/2) + O(n) + O(1) & \end{cases}$$

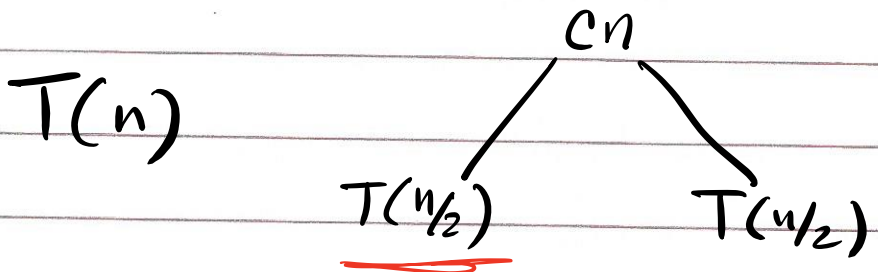
- $2 \cdot T(n/2)$: Conquer time
- $O(n)$: Time to combine
- $O(1)$: time to Divide

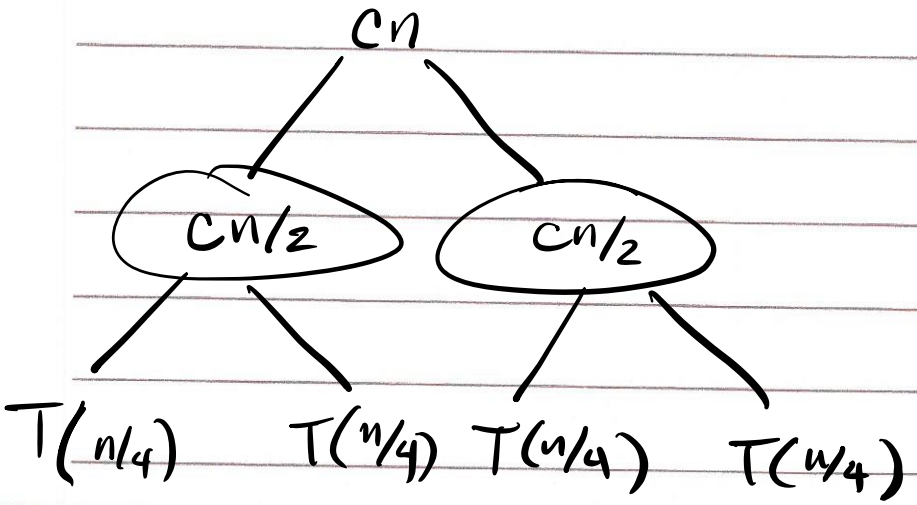
in general, our recurrence equation for a D&C solution will look like:

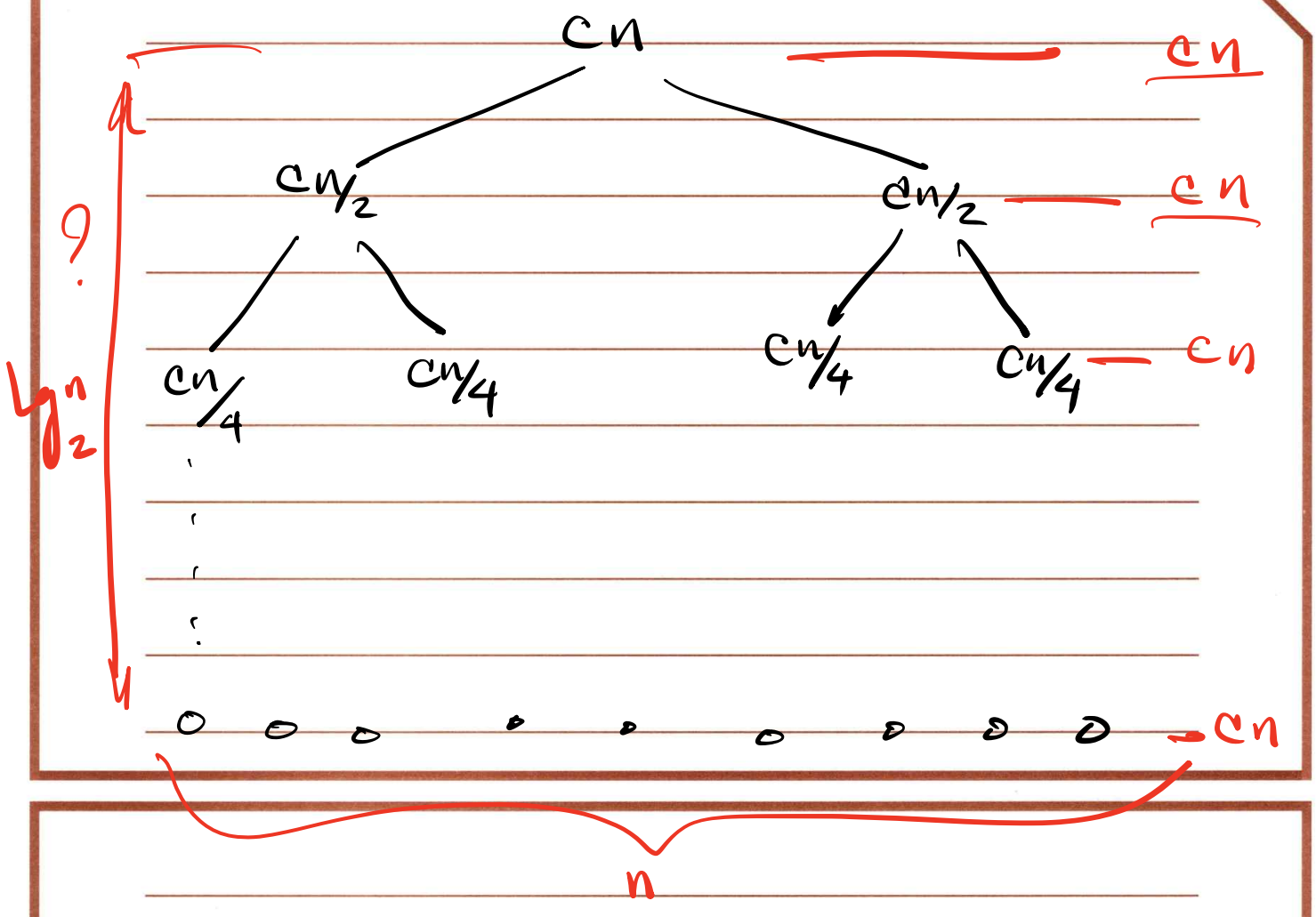
$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT(n/b) + D(n) + C(n) & \end{cases}$$

$f(n)$

- a: # of subproblems
- n/b: size of the subproblem
- $D(n)$: Divide time
- $C(n)$: Combine time







total cost $= \Theta(n \lg n)$

Using Master Method:

$f(n) = \Theta(n)$ ← $T(n) = \Theta(n \lg n)$

$n^{\log_b a} = n^{\log_2 2} = \Theta(n)$

Master Method

It is a cookbook method for solving recurrences of the form

$$T(n) = aT(n/b) + f(n)$$

- where $a \geq 1$, $b \geq 1$ are constants
- $f(n)$ is an asymptotically positive function.

Master Theorem

- Given the above definition of the recurrence relation, $T(n)$ can be bounded asymptotically as follows:

1- If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$,

then $T(n) = \Theta(n^{\log_b a})$

2- If $f(n) = \Theta(n^{\log_b a})$ then

$$T(n) = \Theta(n^{\log_b a} \lg n)$$

3- If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then

$$T(n) = \Theta(f(n))$$

$f(n) \; ? \; n^{\log_b a}$

$n^{\log_b a} = n^1$

$f(n) = n \lg n$

$n^{\log_b 1} = n$

$f(n) = n^{1.00001}$

$T(n) = n \lg^2 n$

not case 3

case #3

Case #2 generalized

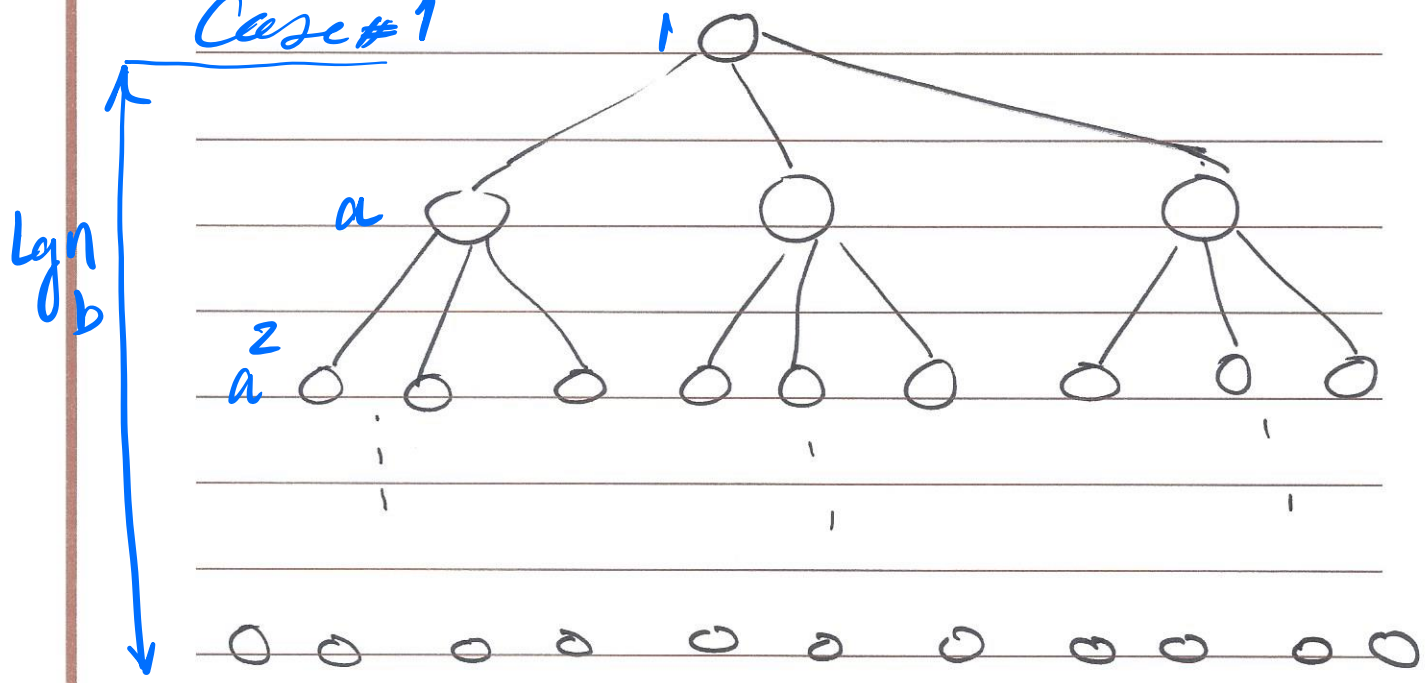
in general if $f(n) = \Theta(n^{\log_b a} \lg^k n)$ where $k \geq 0$

Then

$$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$$

Intuition Behind The Master Method

Case #1



$$a^{\log_b n} = n^{\log_b a}$$

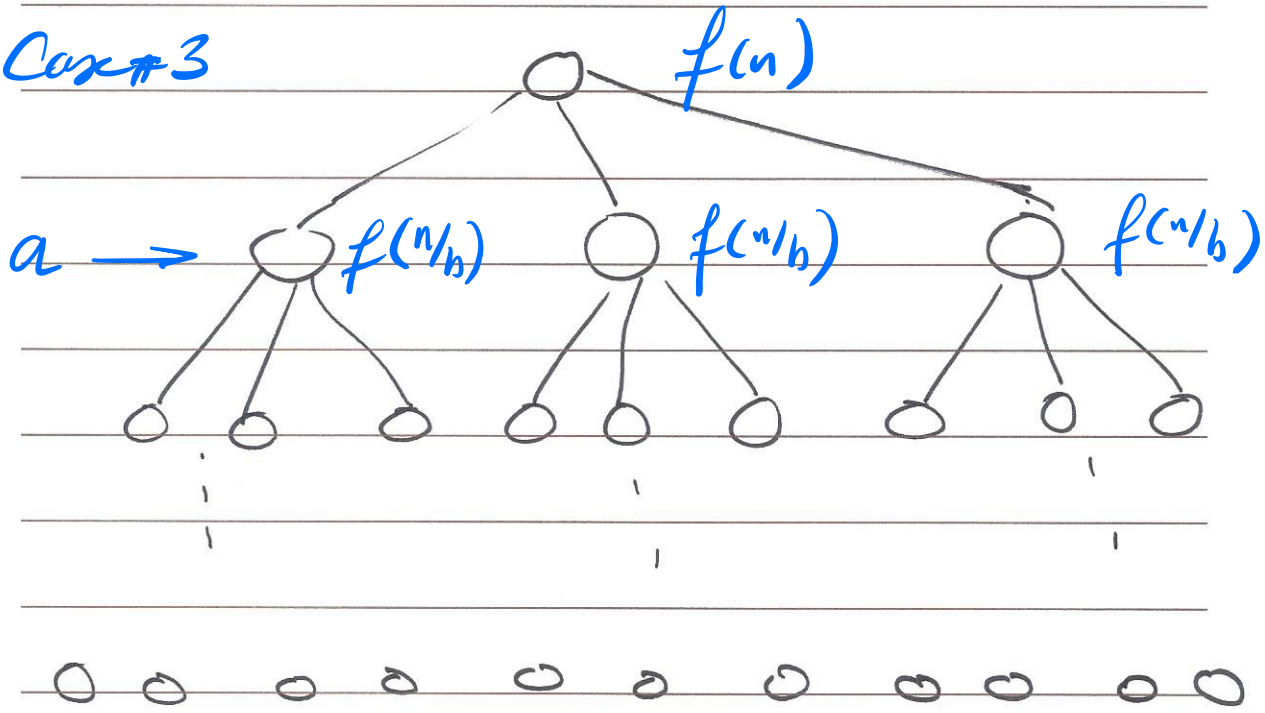
total no. of subproblems =

$$n^{\log_b a} + \frac{1}{a} n^{\log_b a} + \frac{1}{a^2} n^{\log_b a} + \cdots$$

$$= \Theta(n^{\log_b a})$$

Intuition Behind The Master Method

Case #3



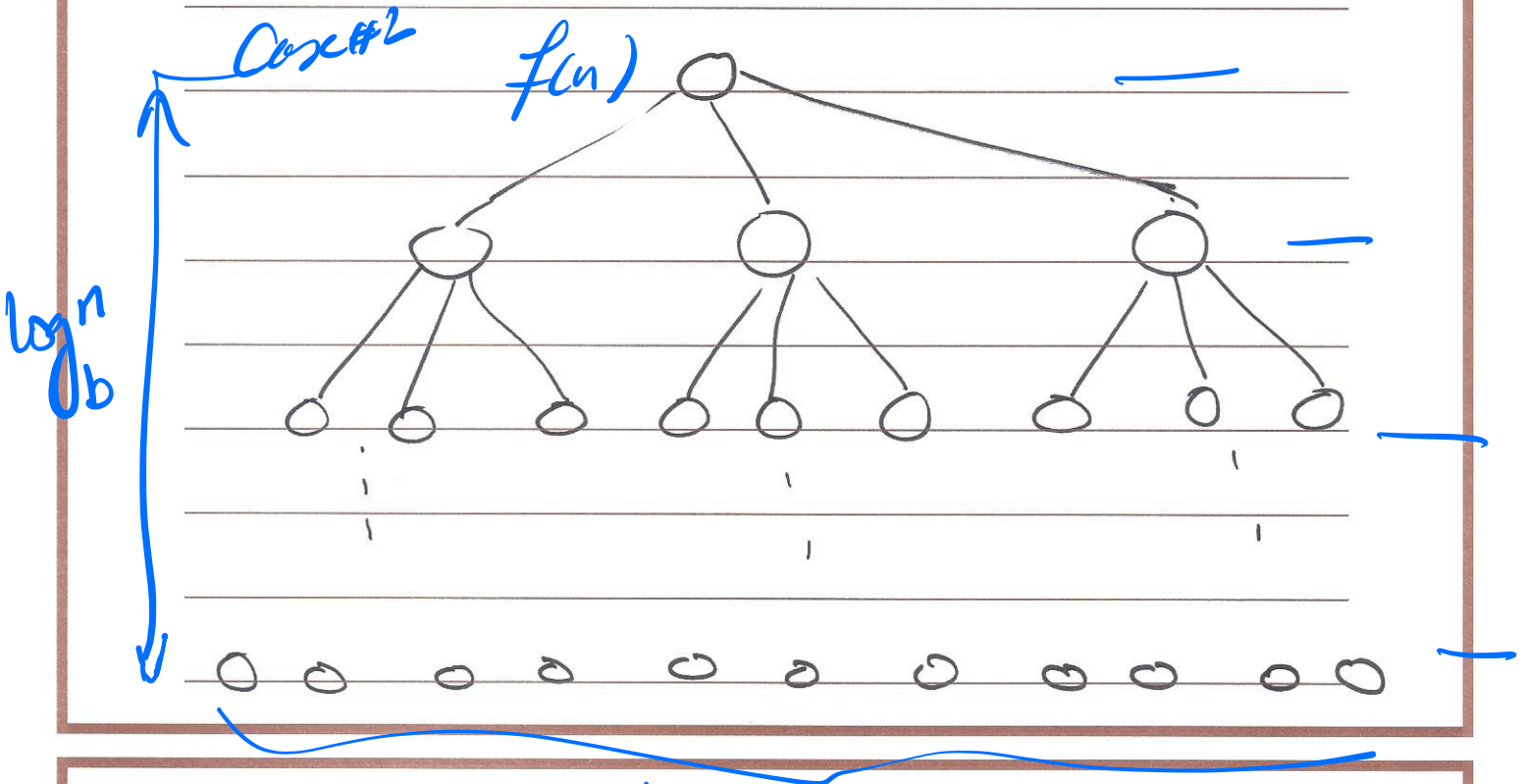
$$a f(n/b) \leq c f(n)$$

for $c < 1$

$$\underbrace{f(n) + c f(n) + c^2 f(n) + \cdots}$$

$$\Theta(f(n))$$

Intuition Behind The Master Method



$n^{\log_b a}$

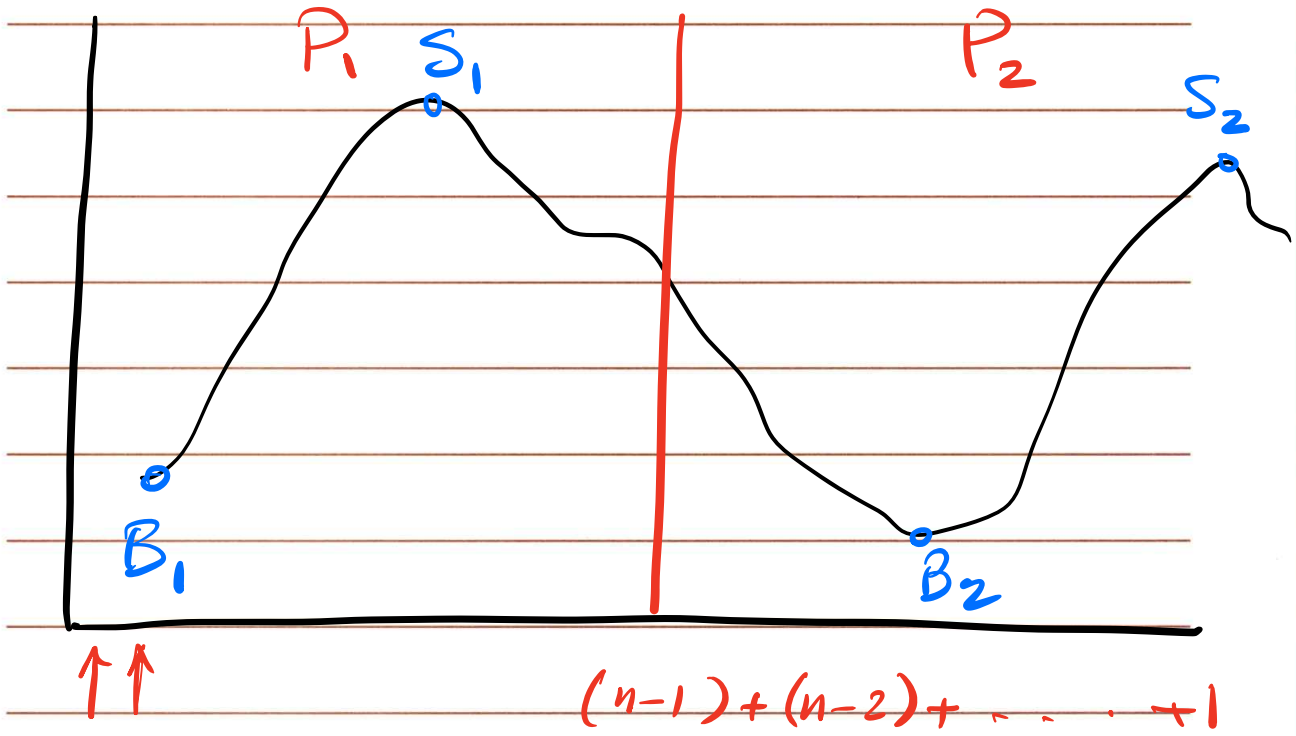
total time $= \Theta\left(n^{\log_b a} \lg n\right)$

Case 1: Complexity driven by the no. of leaf nodes in the recursion tree.

Case 3: Complexity driven by the cost of the root node in the recursion tree

Case 2: Cost of operations are the same at every level of the recursion tree.

Stock Market Problem



Combine steps

Case 1 buy & sell in P_1

$B = B_1$ M = absolute Min

$S = S_1$ X = " Max

Case 2 buy & sell in P_2

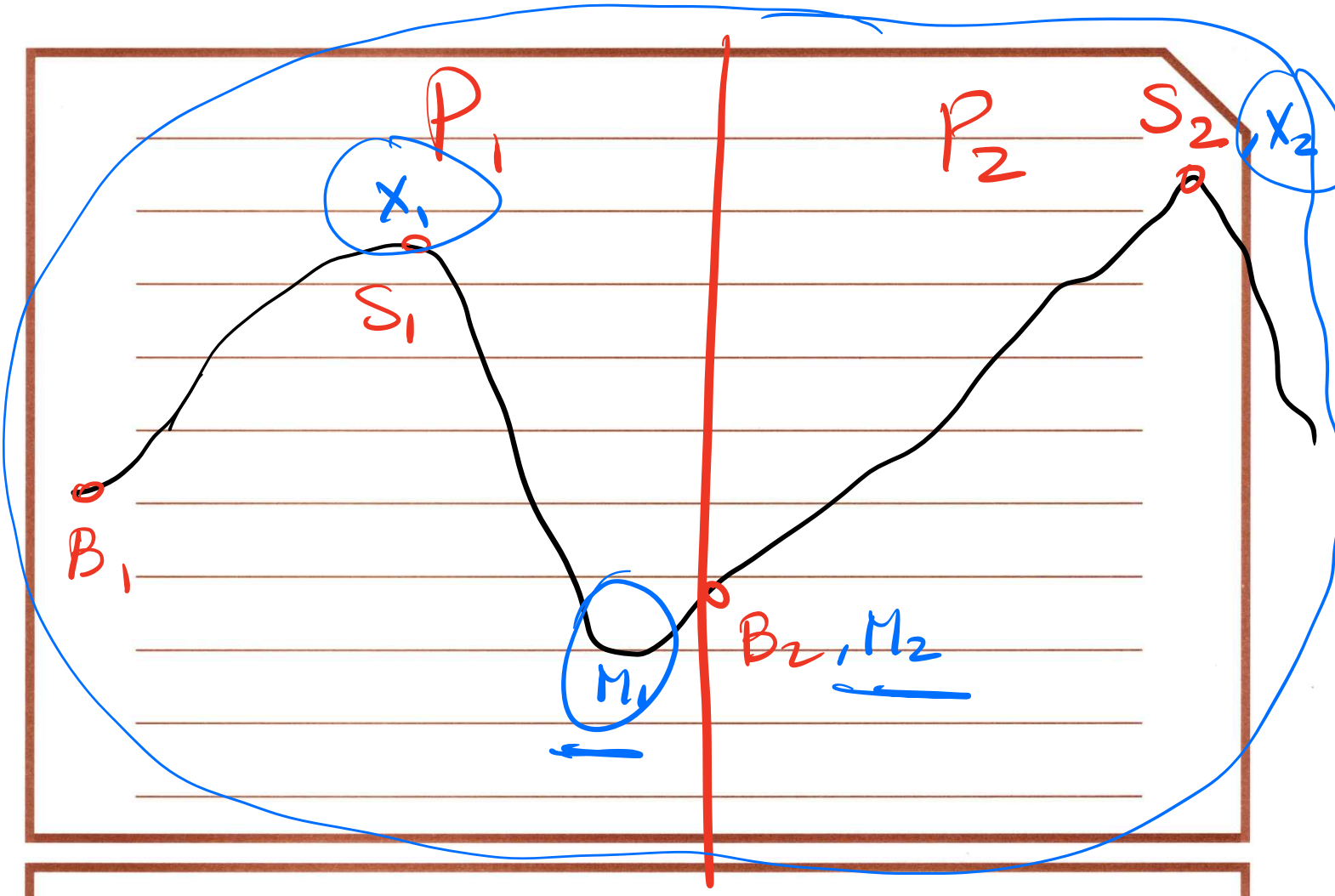
$B = B_2$ M = absolute Min

$S = S_2$ X = " Max

Case 3 buy in P_1 & sell in P_2

$B = B_1$ M = absolute Min

$S = S_2$ X = " Max



Analysis of our solution:

$C(n) = O(1)$ ← Combine

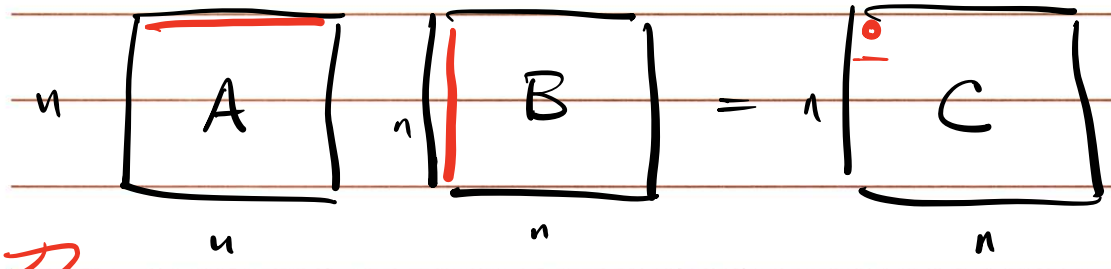
$D(n) = O(1)$ ← Divide

$f(n) = O(1)$

$n^{\log_b q} = n^{\log_2 1} = n^0 = 1$

Case #1 $T(n) = \Theta(n)$

Dense Matrix Multiplication



Brute force takes $\Theta(n^3)$

A_{11}	A_{12}
A_{21}	A_{22}

$\times$

B_{11}	B_{12}
B_{21}	B_{22}

$=$

C_{11}	C_{12}
C_{21}	C_{22}

combine step

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

$f(n) = D(n) + C(n) = \Theta(1) + \Theta(n^2) = \Theta(n^2)$

$n^{\log_b q} = n^{\log_2 8} = n^3$

Case #1 $\rightarrow T(n) = \Theta(n^3)$

Compute 7 $n/2 \times n/2$ intermediate matrices

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$Q = (A_{21} + A_{22})\, B_{11}$$
$$R = A_{11}(B_{12} - B_{22})$$
$$S = A_{22}(B_{21} - B_{11})$$
$$T = (A_{11} + A_{12})\, B_{22}$$
$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$V = (A_{12} - A_{22})(B_{21} + B_{22})$$

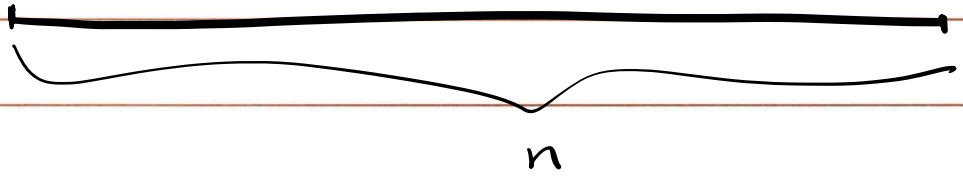
$$C_{11} = P + S - T + V$$
$$C_{12} = R + T$$
$$C_{21} = Q + S$$
$$C_{22} = P + R - Q + U$$

Strassen's Method

$n^{\log_b a} = n^{\log_2 7} = n^{2.81}$

$f(n) = \Theta(n^2) \Rightarrow T(n) = \Theta(n^{2.81})$

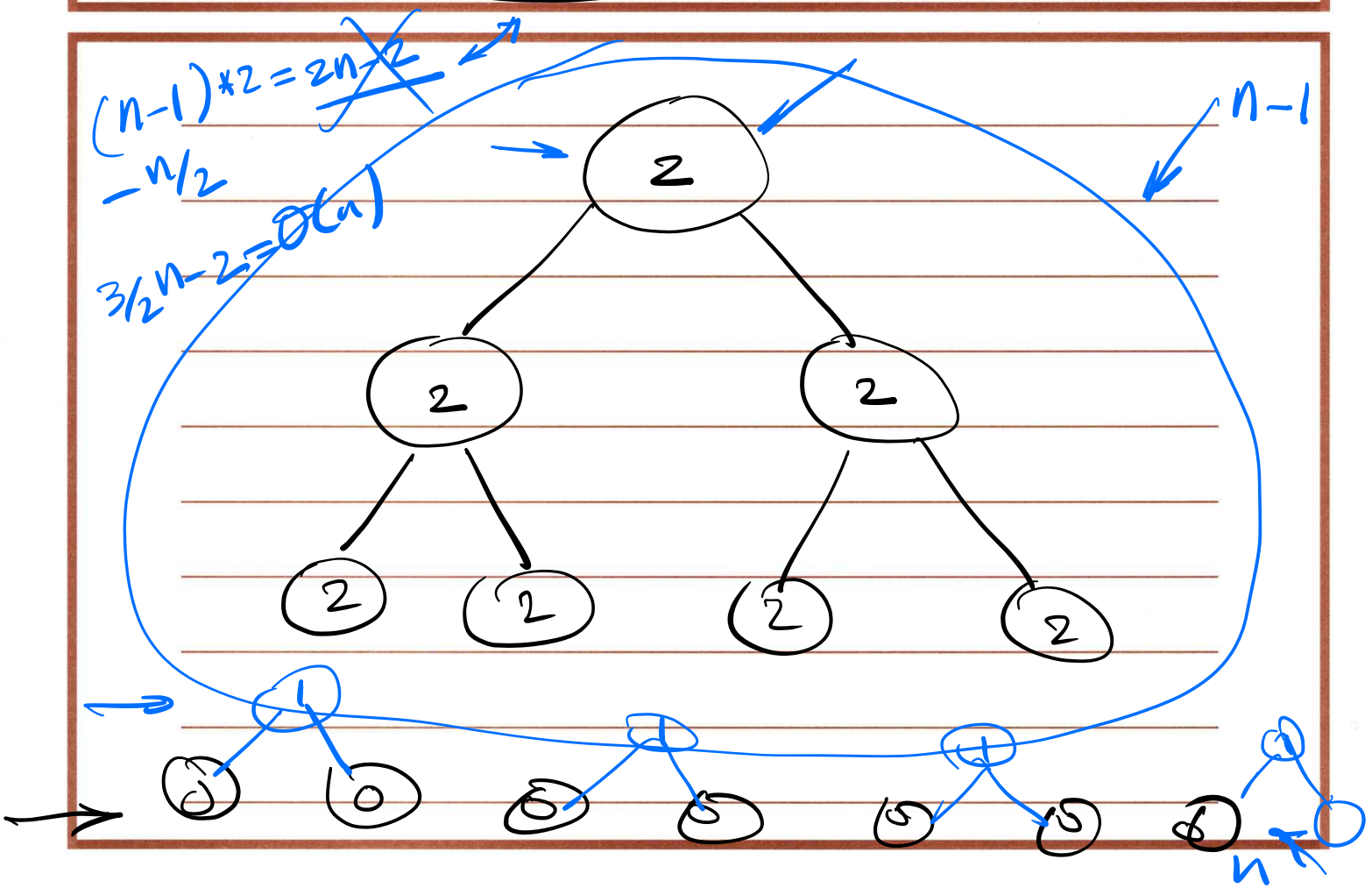
Finding Min & Max in an unsorted array



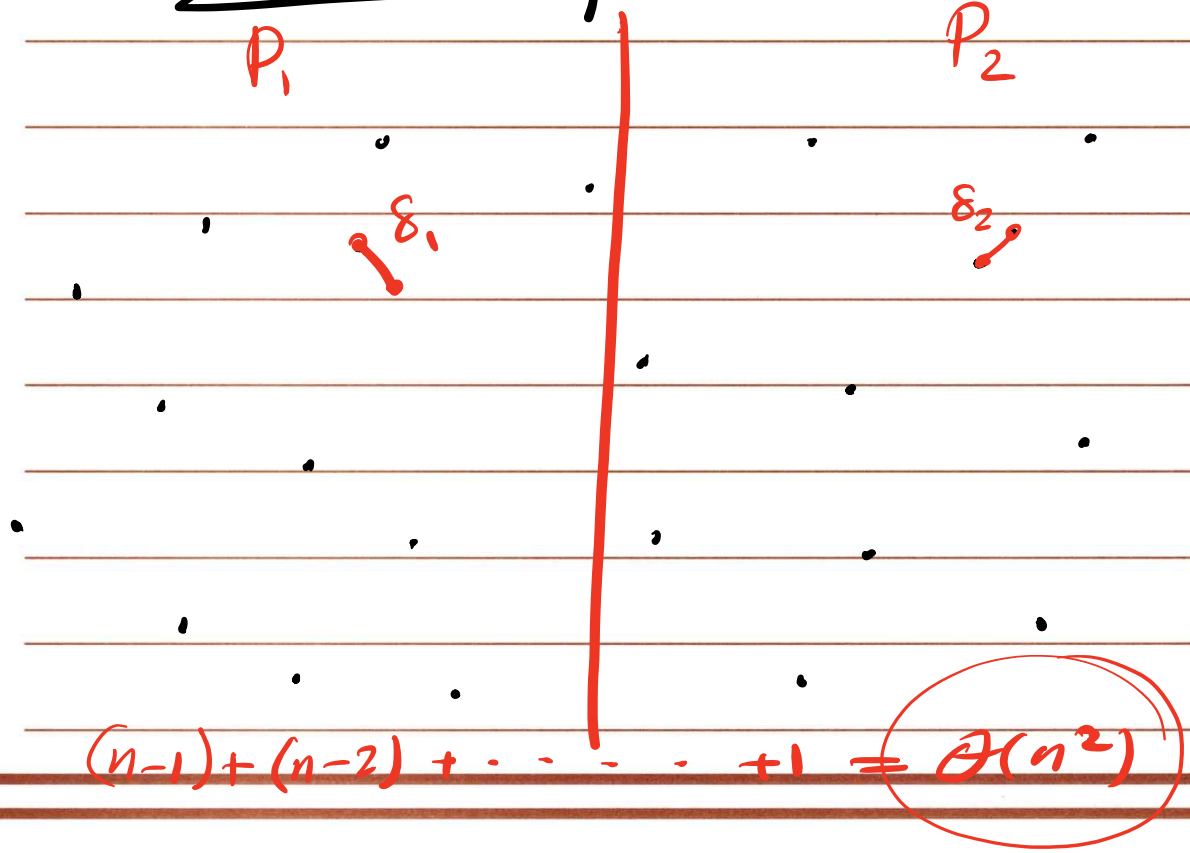
to find Min takes n-1 op's

" " Max " n-1 "

2n-2 $\Theta(n)$



Closest pair of points problem (2D)



Combine Step.

Case 1 :

Both pts are in P_1 ✓

Case 2:

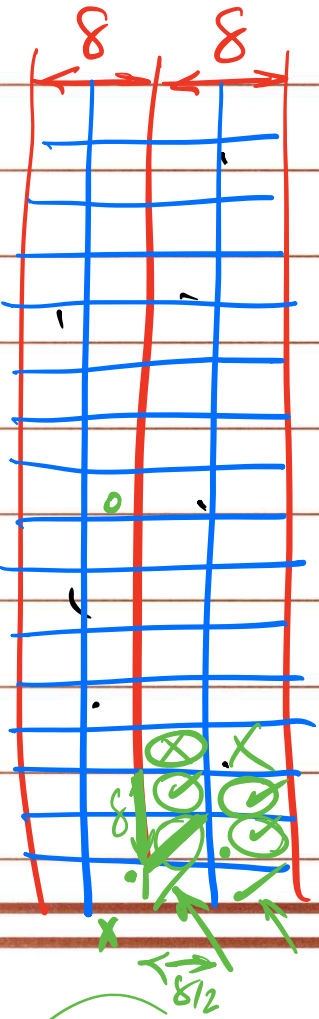
Both pts are in P_2 ✓

Case 3:

One pt. is in P_1 & the other in P_2

$\delta = ?$

$\delta = Min(\delta_1, \delta_2)$



$\delta/2$

$\delta/2$ □

$\frac{\delta\sqrt{2}}{2} < \delta$

Implementation

$O(n \log n)$

closest-pair (P)

Construct P_x : list of points sorted by x-coord.

" P_y : list of points sorted by y-coord.

$(p_0, p_1) =$ closest-pair-Rec(P_x, P_y)

---

closest-pair-Rec (P_x, P_y)

if $|P| \leq 3$ then

solve it directly

else

Divide

$O(1)$ Construct Q_x ... left half of P_x

$O(n)$ → " Q_y ... list of points in Q_x sorted by y-coord.

$O(1)$ Construct R_x ... right half of P_x

$O(n)$ → " R_y ... list of points in R_x sorted by y-coord.

Conquer

(q_0, q_1) = closest_pair_Rec (Q_x, Q_y)

(r_0, r_1) = closest_pair_Rec (R_x, R_y)

$O(1)$ → $\delta = \min(d(q_0, q_1), d(r_0, r_1))$

$O(n)$ → S = set of points in P within distance of δ from L.

Construct S_y = set of points in S sorted by y-coord.

Combine

$O(n)$ → for each point $s \in S_y$, compute distance from s to each of next 11 points in S_y. Let (s, s') be pair with min. distance

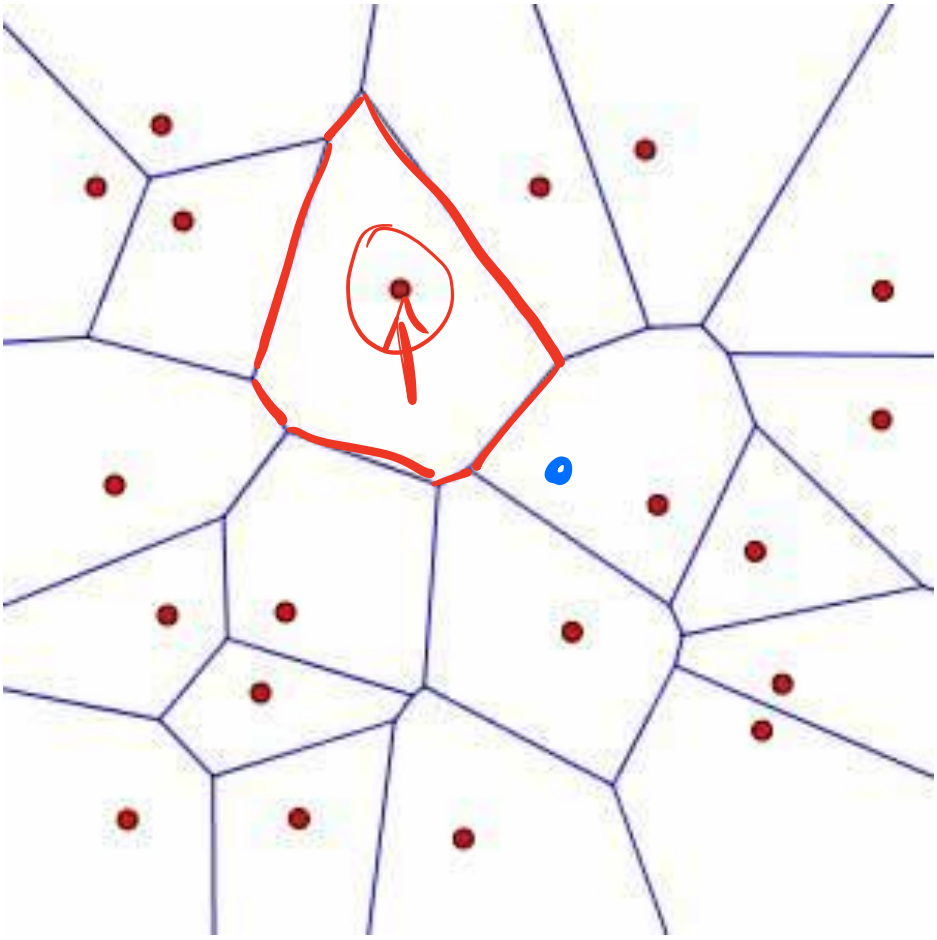
$O(1)$ →

```
if d(s,s') < δ then
    Return (s,s')
else if d(q0,q1) < d(r0,r1) then
    Return (q0,q1)
else
    Return (r0,r1)
endif
```

$f(n) = \Theta(n)$, $\log_2^2 = n$, $\log_b^a = n$, n^1

$T(n) = \Theta(n \log n)$

All Nearest Neighbors Problem



Voronoi Diagram