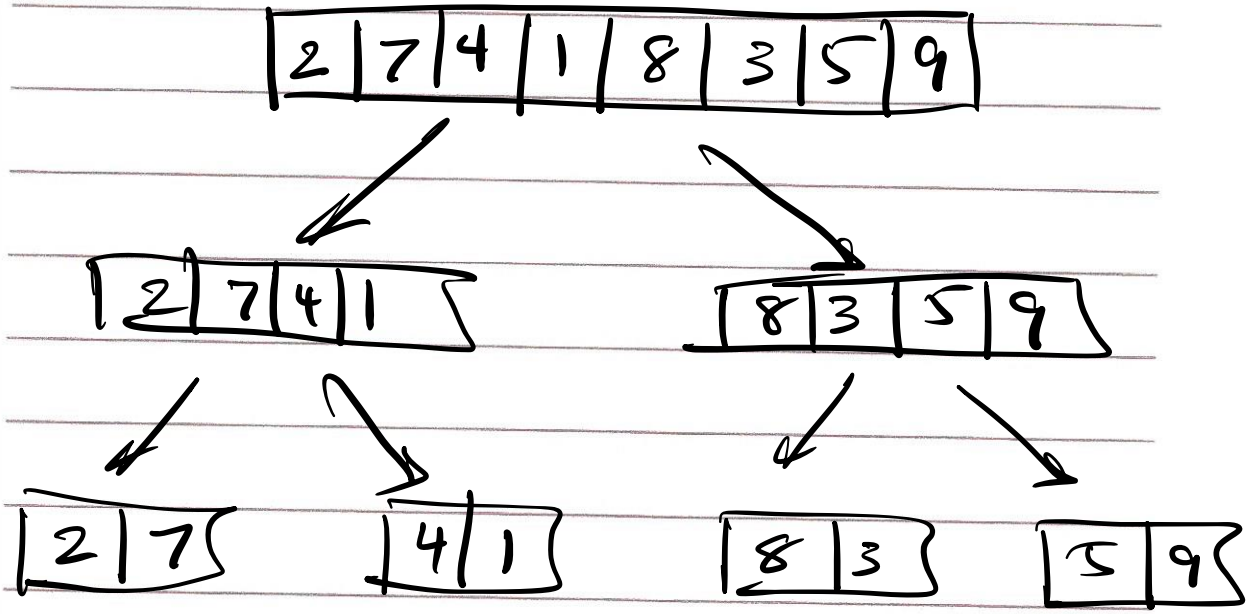
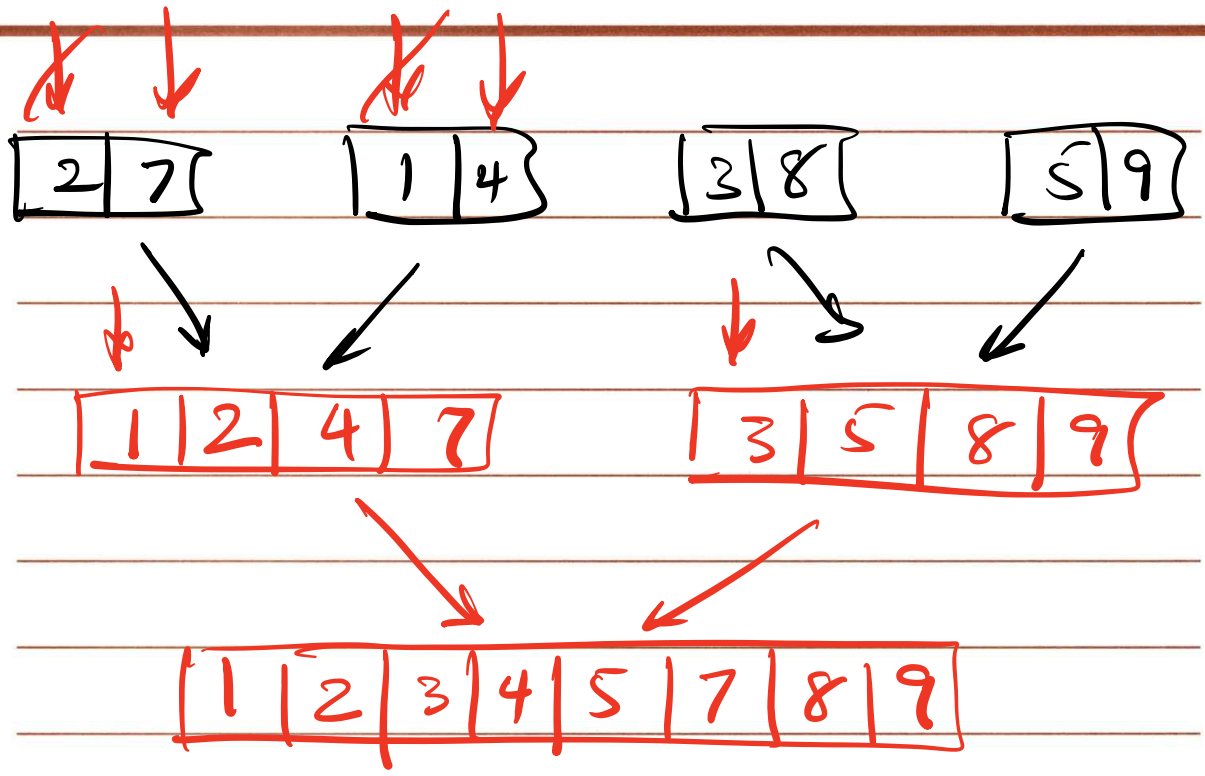
Divide & Conquer

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
1 2 3 4 5 7 8 9

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

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 of size n.

Recurrence Equation for merge-sort

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

Conquer

Combine

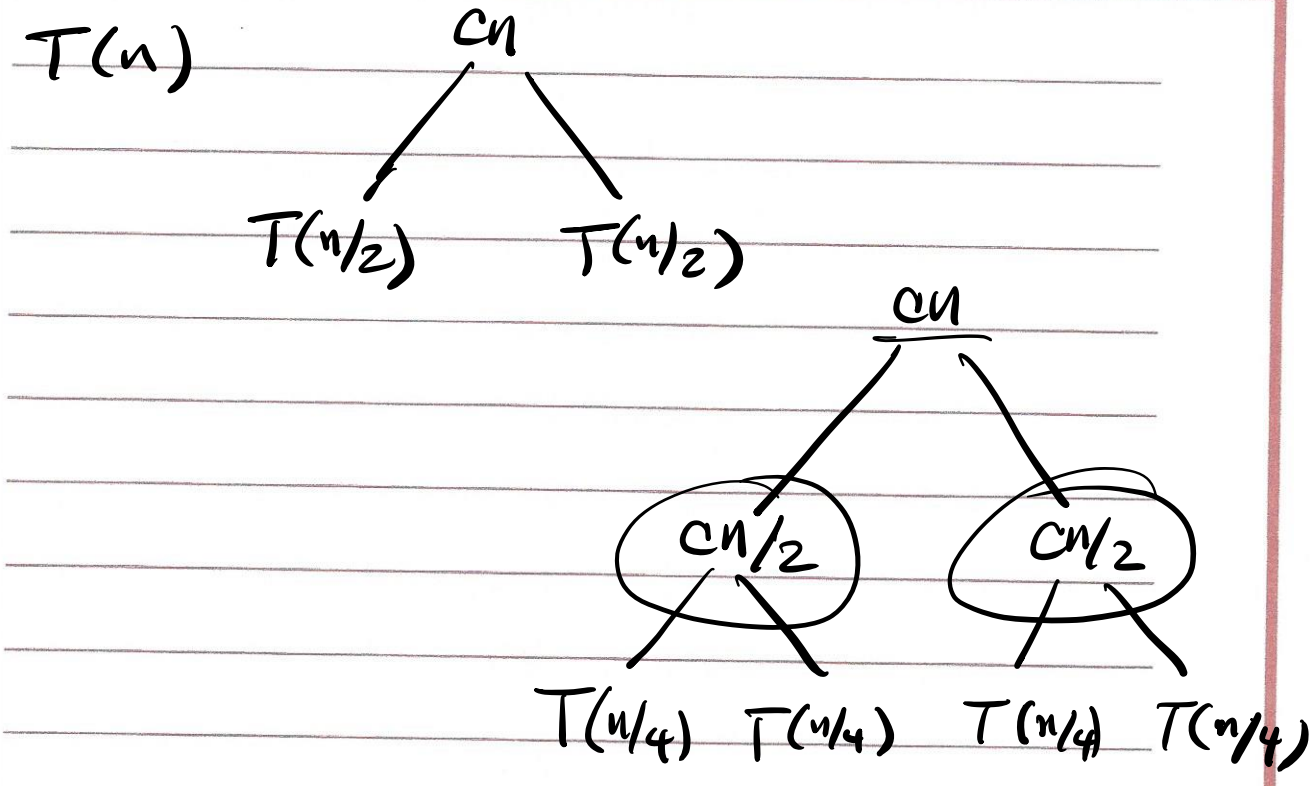
Divide

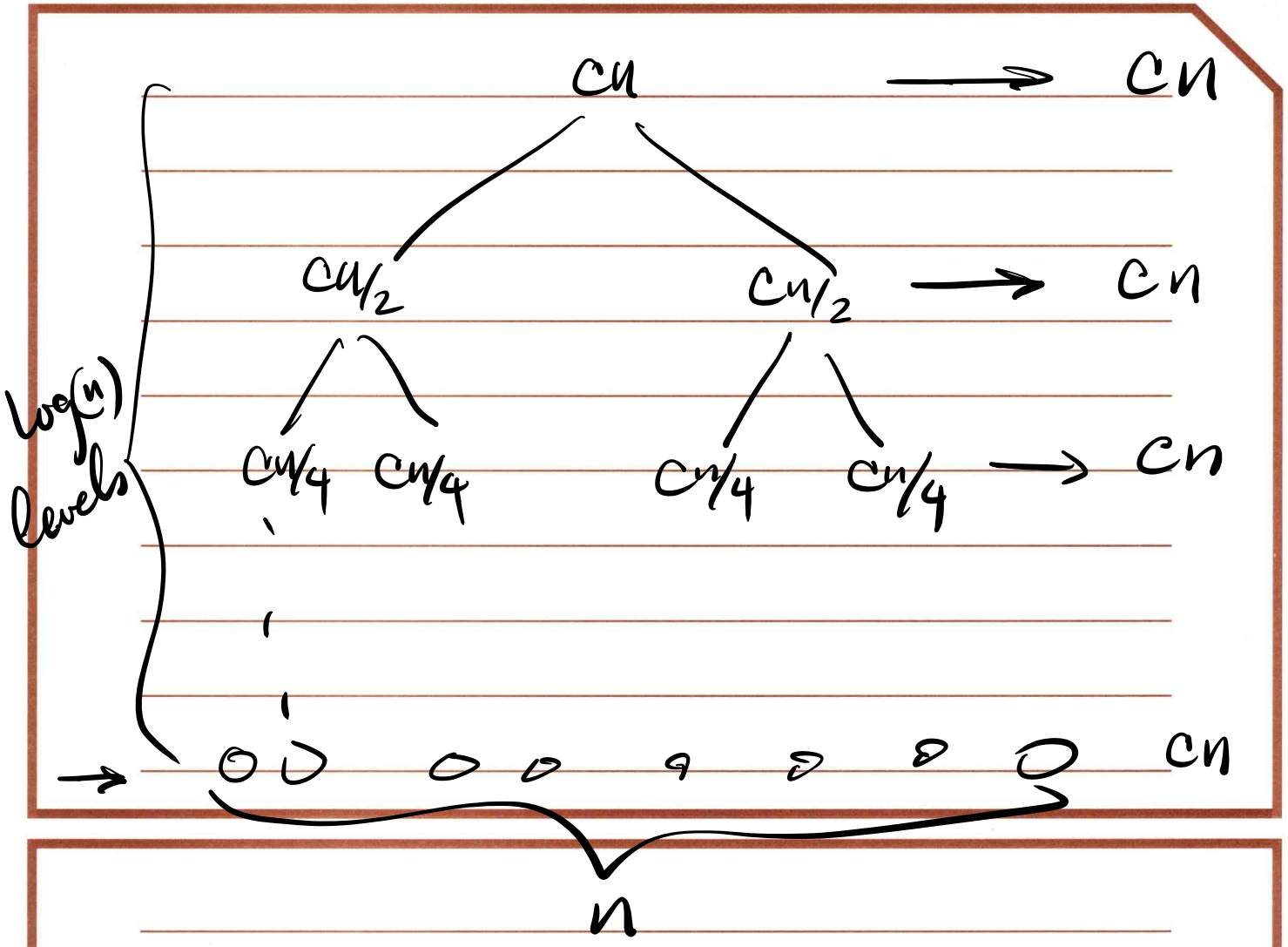
in general, our recurrence equation for a D&C solution will look like:

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

$f(n)$

- a: # of subproblems of each step
- n/b: Size of each subproblem
- $D(n)$: Divide time
- $C(n)$: Combine time





$$\text{overall cost} = \Theta(n \log n)$$

Using Master Method on Merge Sort

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

$$n^{\log_b a} = n^{\log_2 1} = n$$

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

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 $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n, then

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

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

$f(n) = n \lg^1 n$

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

$f(n) = n^{1000}$

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

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

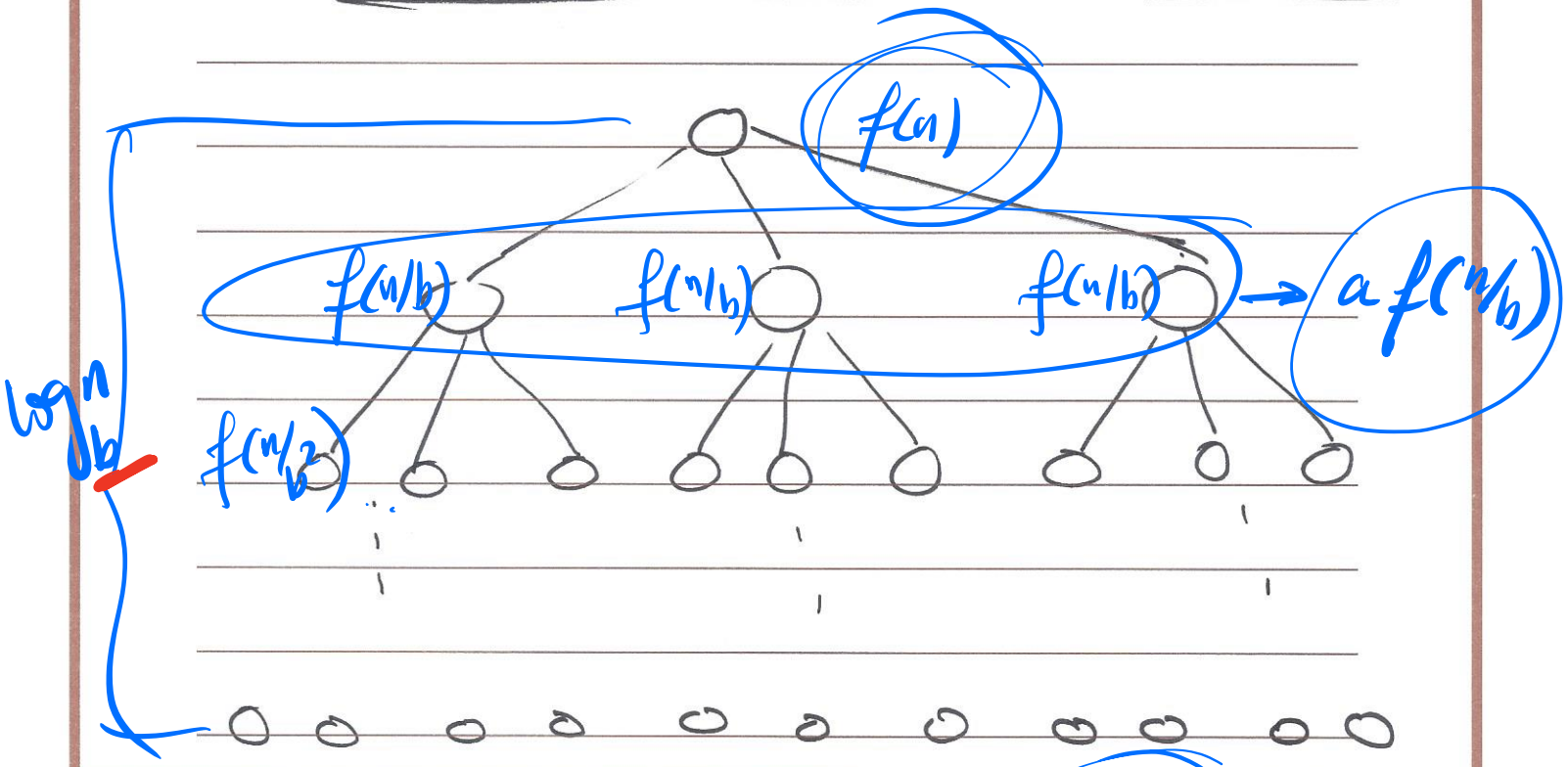
in general if $f(n) = \Theta(n^{\log_b a} \lg^k n)$

where $k \geq 0$

Then

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

Intuition Behind The Master Method



$1, a, a^2, \ldots a^{\log_b n} = n^{\log_b a}$

$n + \alpha n + \alpha^2 n + \alpha^3 n + \ldots$

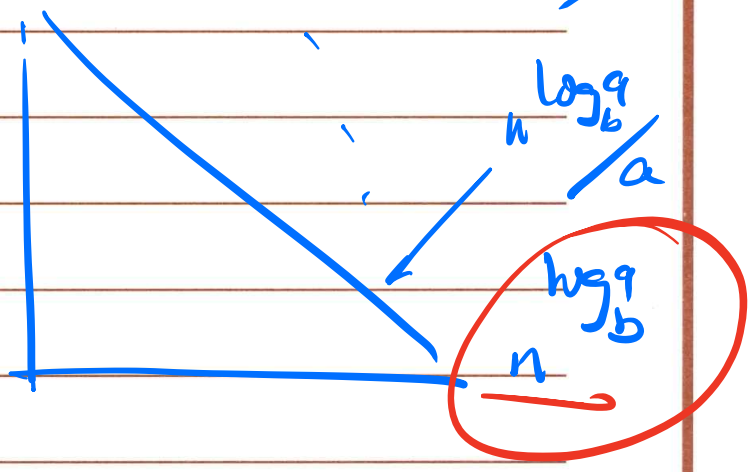
$\alpha < 1 \longrightarrow$ Sum will be $O(n)$

$n + 1/2\, n + 1/4\, n + 1/8\, n + \ldots$

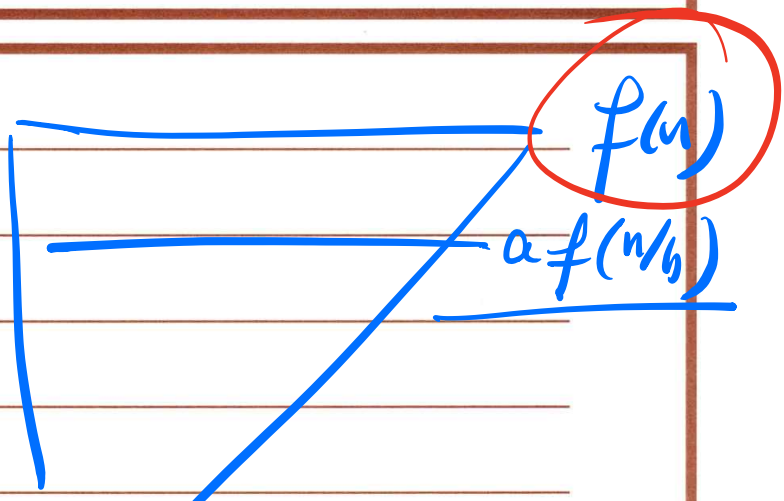
$2n = O(n)$

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

$\text{Sum} = O(n^{\log_b^a})$



Case 3



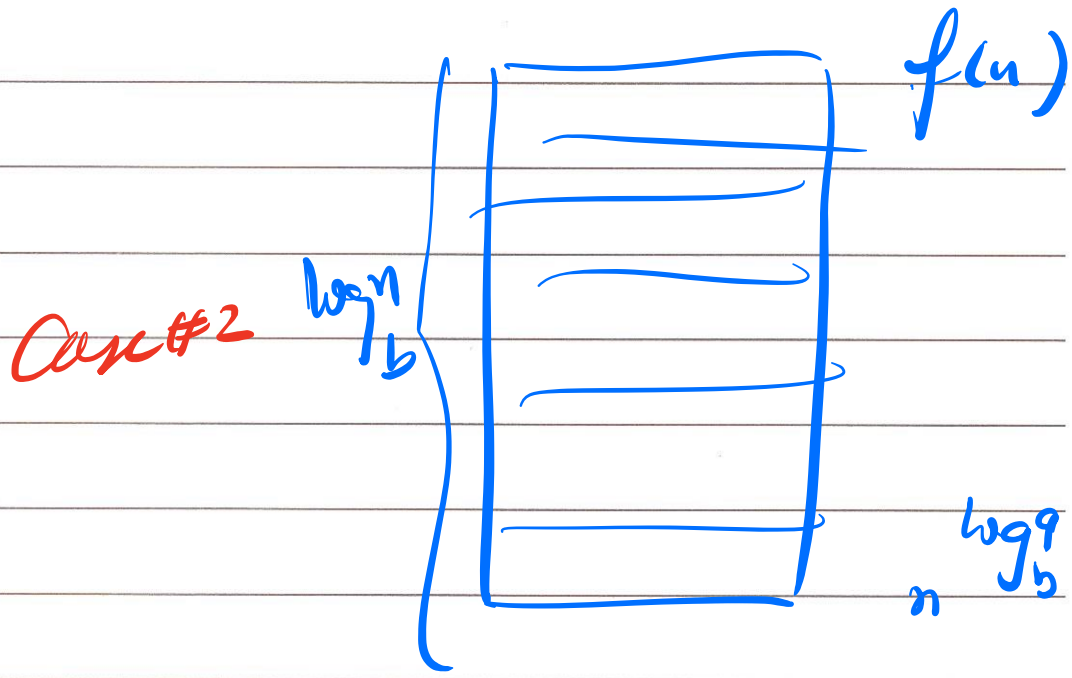
$f(n) + af(n) + a^2 f(n) + \cdots$

$O(f(n))$

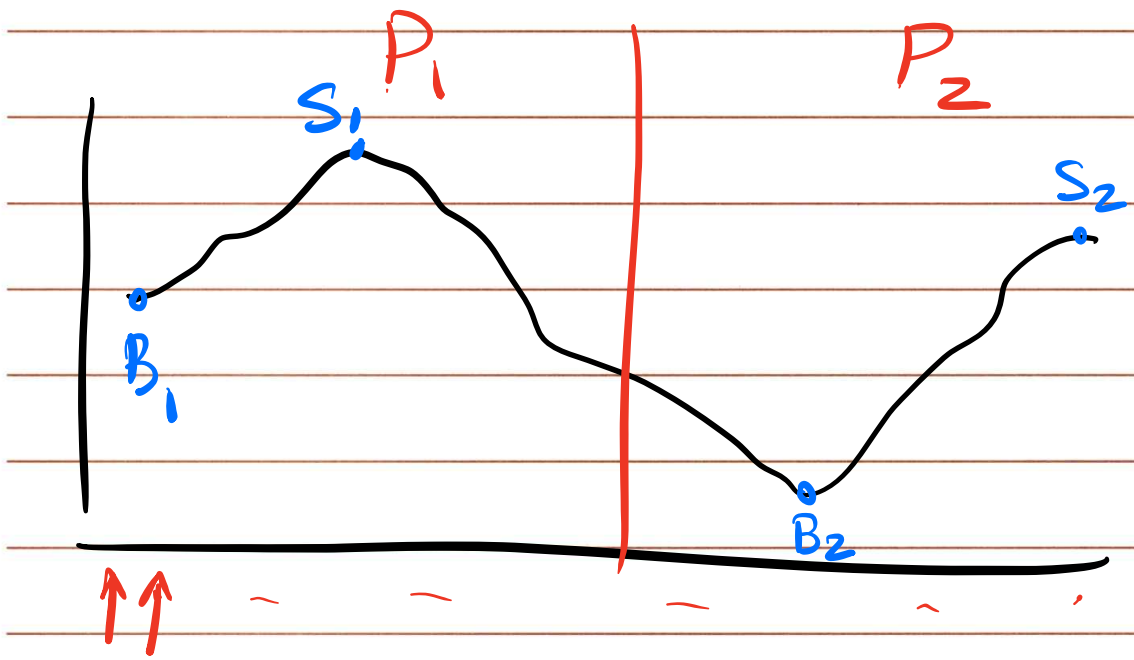
Case 1: Complexity driven by the no. of leaf nodes in the recursion tree.

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



$(n-1)+(n-2)+(n-3)+\cdots+1$

$= O(n^2)$

Case 1: buy & sell in P_1

$B = B_1$

$S = S_1$

$M = \min(M_1, M_2)$

$X = \max(X_1, X_2)$

Case 2: buy & sell in P_2

$B = B_2$

$S = S_2$

$M = \min(M_1, M_2)$

$X = \max(X_1, X_2)$

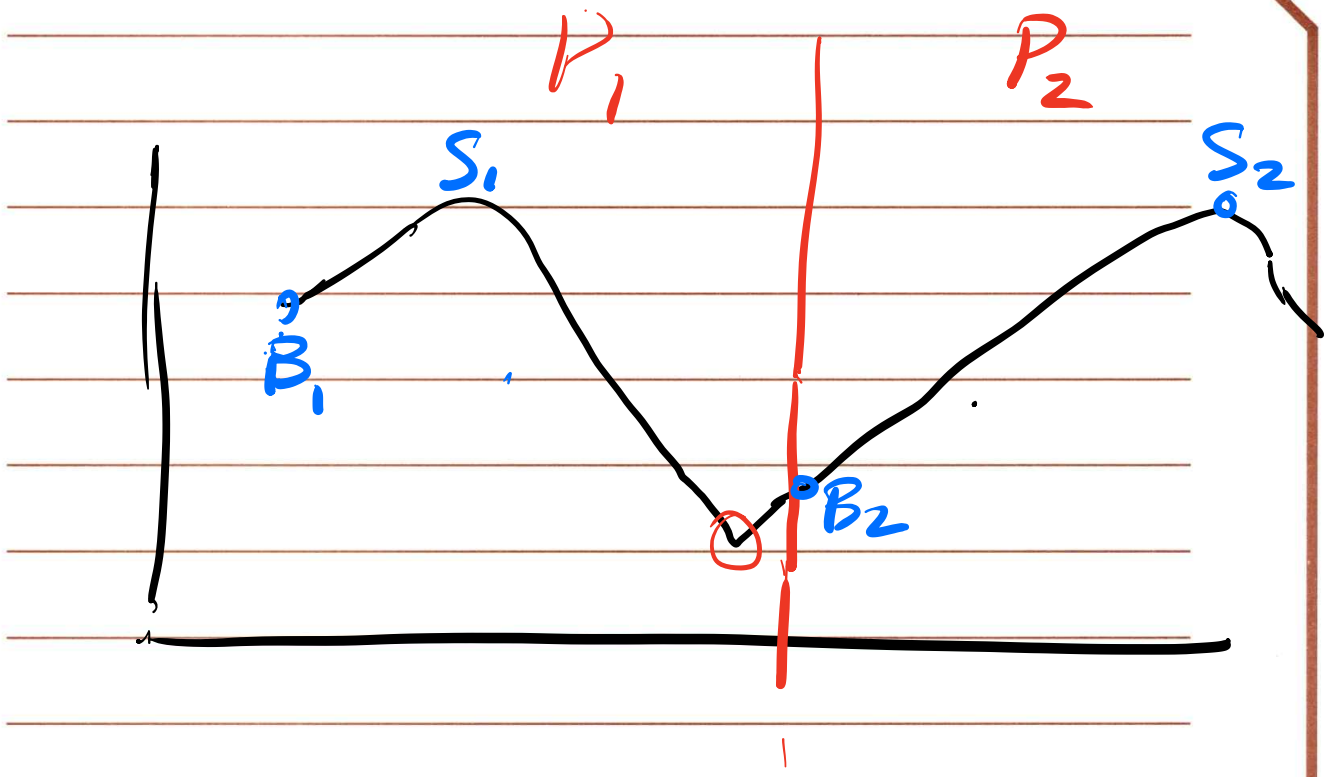
Case 3: buy in P_1 & sell in P_2

$B = B_1$ M_1

$S = S_2$ X_2

$M = \min(M_1, M_2)$

$X = \max(X_1, X_2)$



$D(n) = \Theta(1)$

$C(n) = \Theta(1)$

$f(n) = \Theta(1)$

$n^{\log_b a} = n^{\log_2 2} = n^1$

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

Dense Matrix Multiplication

$$\underbrace{\begin{bmatrix} A \end{bmatrix}}_{n} \underbrace{\begin{bmatrix} B \end{bmatrix}}_{n} = \underbrace{\begin{bmatrix} C \end{bmatrix}}_{n} \Big\} n$$

Brute Force mult. $\rightarrow \Theta(n^3)$

$$\begin{bmatrix} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{bmatrix}$$

combine Step

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

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

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

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

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

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

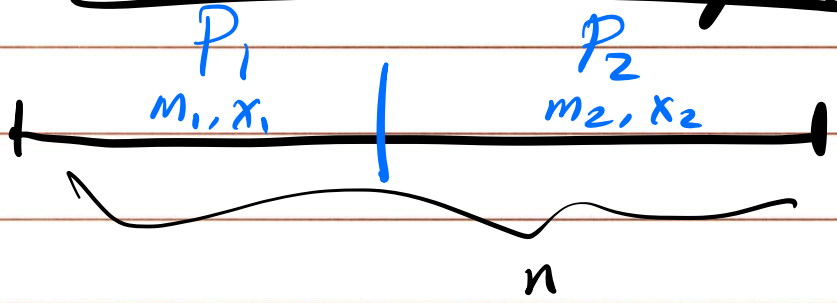
$$C_{11} = P + S - T + V$$
$$C_{12} = R + T$$
$$C_{21} = Q + S$$
$$C_{22} = P + R - Q + U$$

Strassen's Method

$a = 7$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n^{\log_2 7} = n^{2.81}$

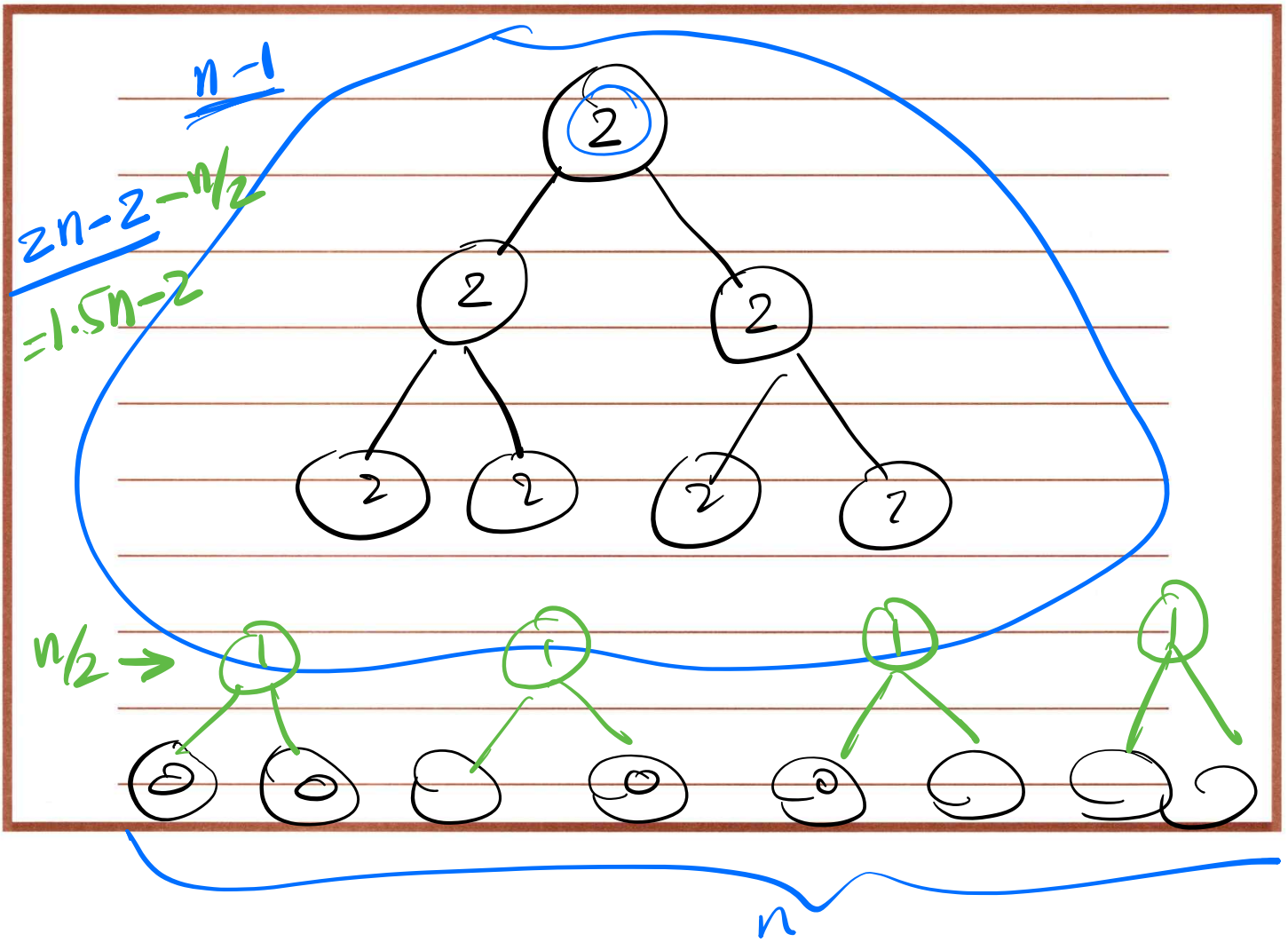
$f(n) = \Theta(n^2)$ $\longleftarrow$ case #1 $\rightarrow T(n) = \Theta(n^{2.81})$

Finding Min & Max in an unsorted array

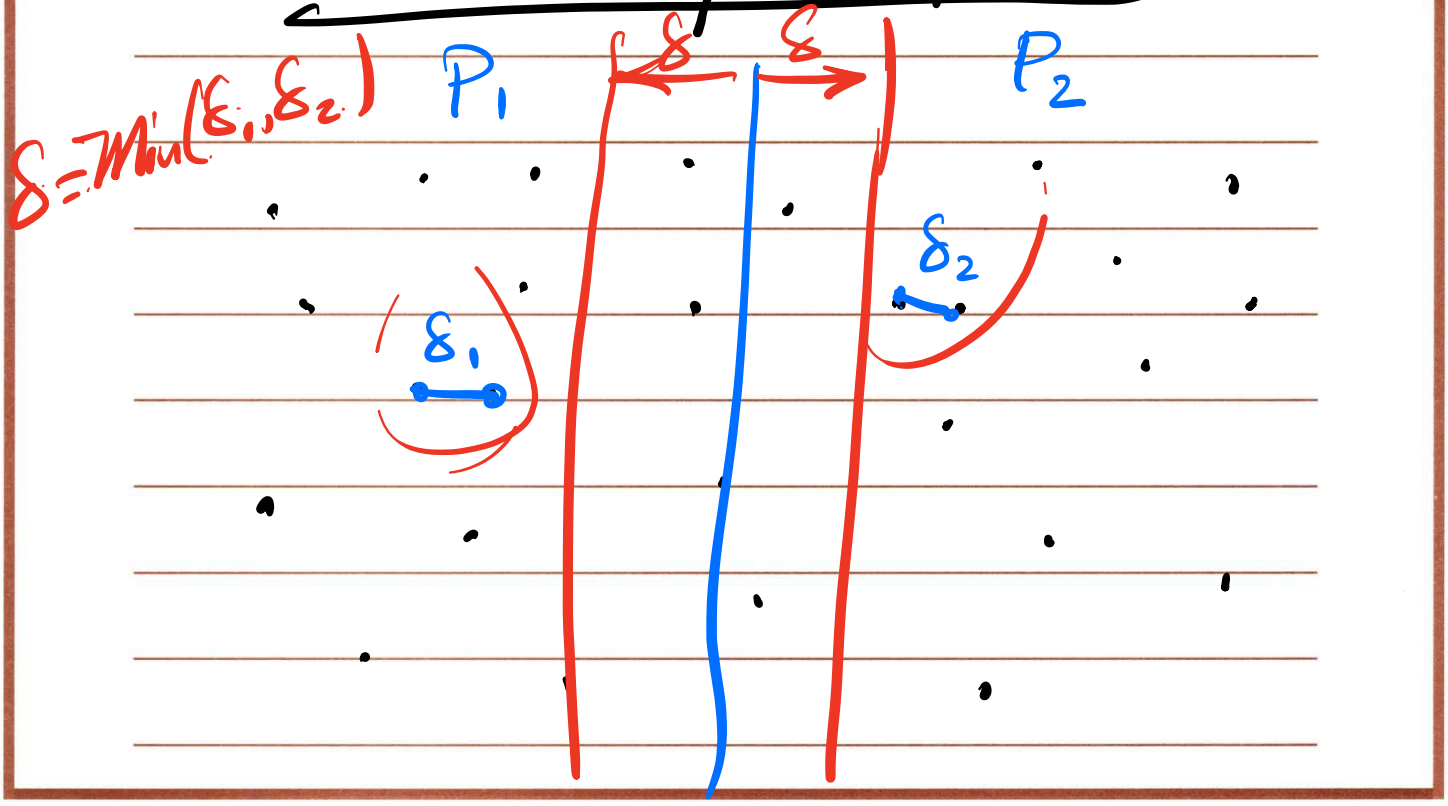


Brute force $(n-1)$ comps to find Min
$(n-1)$ " " " Max

$2n-2 = \Theta(n)$



Closest pair of points problem (2D)

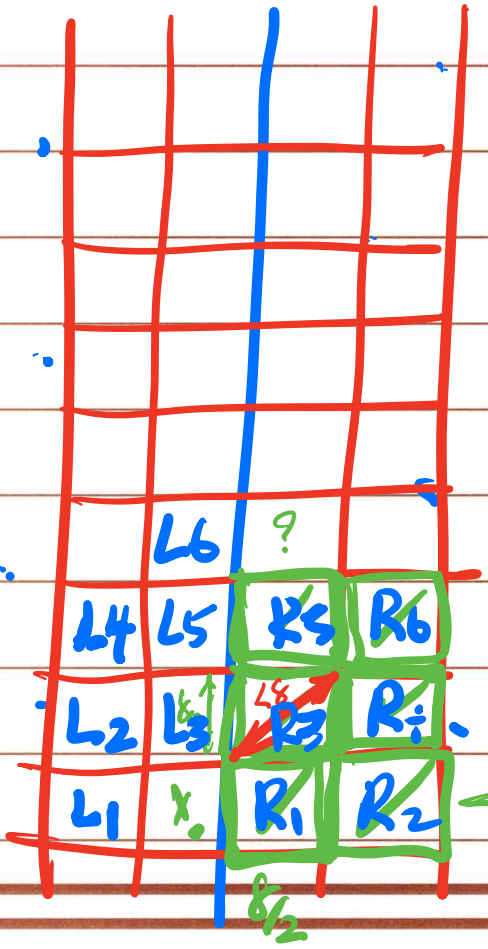


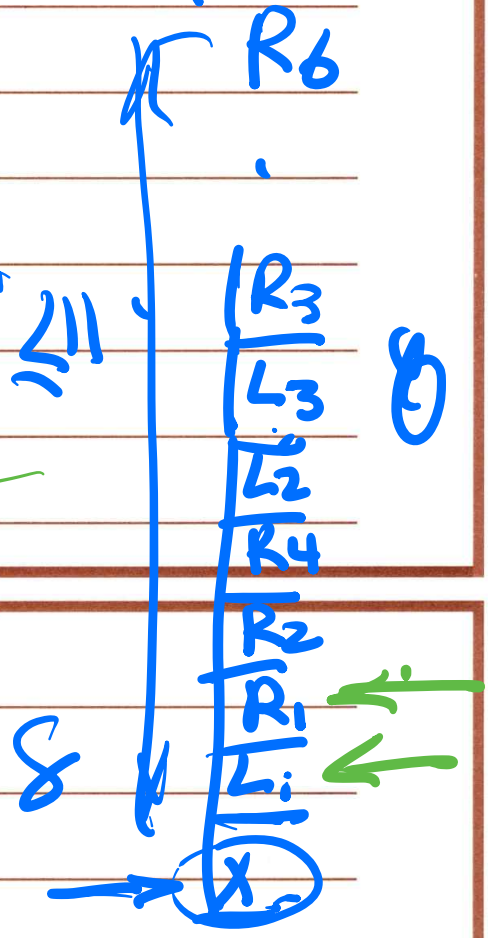
Brute force takes $\Theta(n^2)$ time.

Case #1 : Both pts are in P_1 ✓

Case #2 : " " " P_2 ✓

Case #3 : one pt. is in P_1 the other in P_2





$\delta/2$

$\delta/2$ □

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

Implementation

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:

$\Theta(1)$ Construct Q_x ... left half of P_x

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

$\Theta(1)$ Construct R_x ... right half of P_x

$\Theta(n)$ " 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)

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

S = set of points in P within distance of δ from L.

$O(n)$ 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

if $d(s, s') < \delta$ then

Return (s, s')

$O(1)$ else if $d(q_0, q_1) < d(r_0, r_1)$ then

Return (q_0, q_1)

else

Return (r_0, r_1)

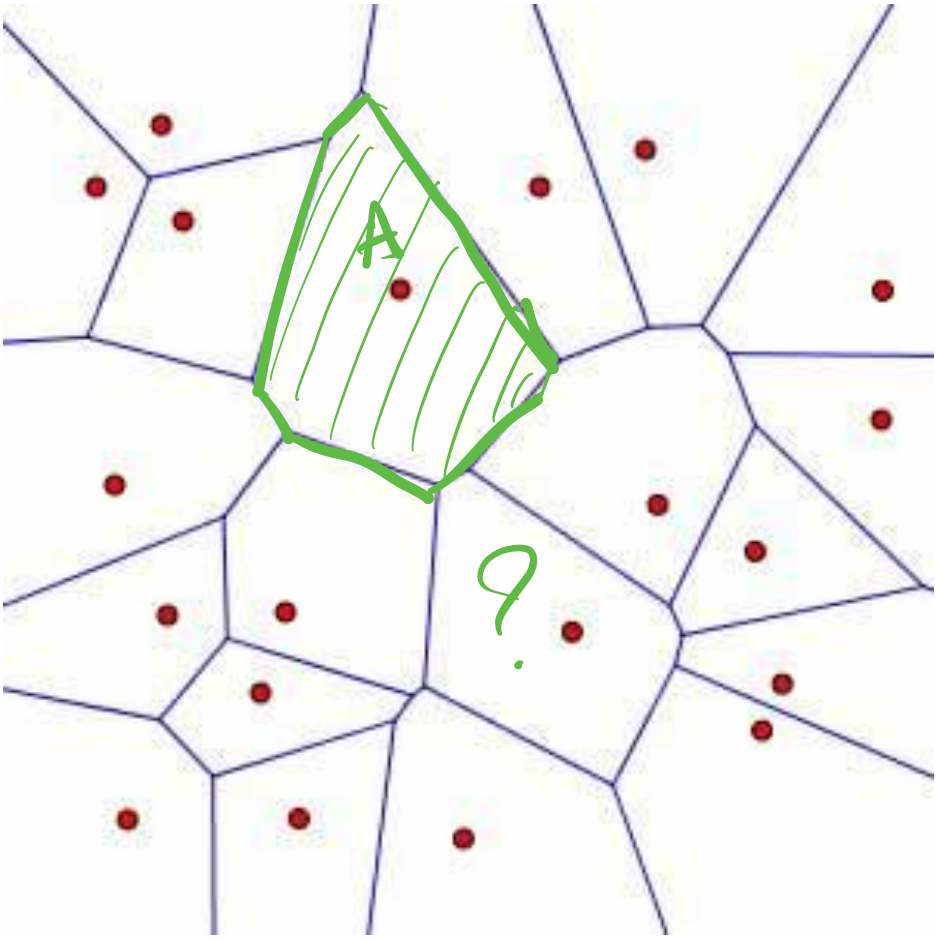
endif

$f(n) = O(n)$

$\log_b^a = n^{\log_2^1} = n$

Case #2 $\rightarrow$ $T(n) = O(n \lg n)$

All Nearest Neighbors Problem



Voronoi Diagram

Discussion 5

1. Suppose we have two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, along with T_1 which is a MST of G_1 and T_2 which is a MST of G_2. Now consider a new graph $G = (V, E)$ such that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup E_3$ where E_3 is a new set of edges that all cross the cut (V_1, V_2).

Consider the following algorithm, which is intended to find a MST of G.

```
Maybe-MST(T1, T2, E3)
    emin = a minimum weight edge in E3
    T = T1 U T2 U { emin}
    return T
```

Does this algorithm correctly find a MST of G? Either prove it does or prove it does not.

2. Solve the following recurrences using the Master Method:

a. $A(n) = 3\,A(n/3) + 15$
b. $B(n) = 4\,B(n/2) + n^3$
c. $C(n) = 4\,C(n/2) + n^2$
d. $D(n) = 4\,D(n/2) + n$

3. There are 2 sorted arrays A and B of size n each. Design a D&C algorithm to find the median of the array obtained after merging the above 2 arrays (i.e. array of length 2n). Discuss its runtime complexity.

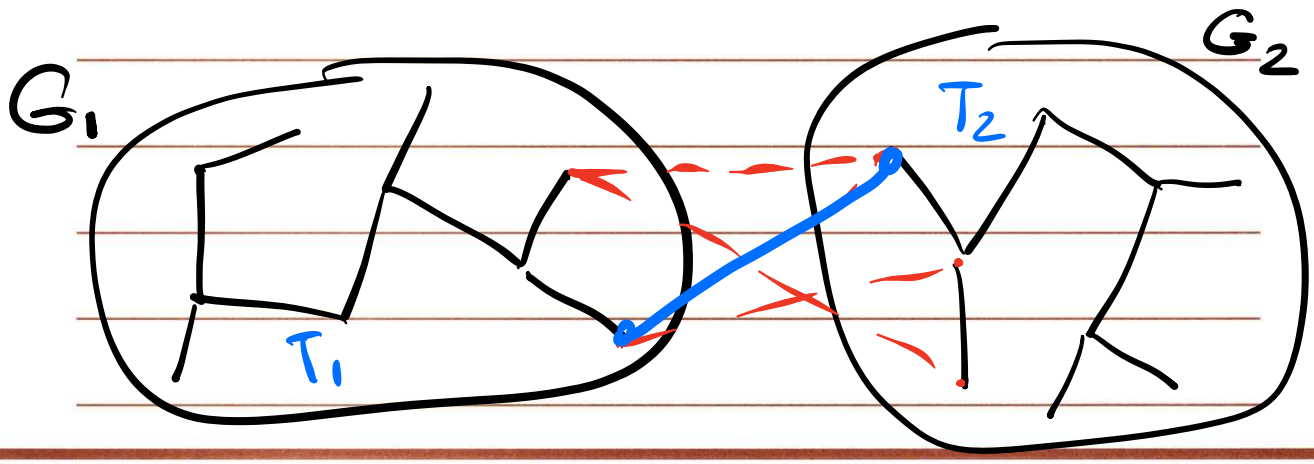
4. A tromino is a figure composed of three 1x1 squares in the shape of an L.
Given a $2^n x 2^n$ checkerboard with 1 missing square, tile it with trominoes.
Design a D&C algorithm and discuss its runtime complexity.

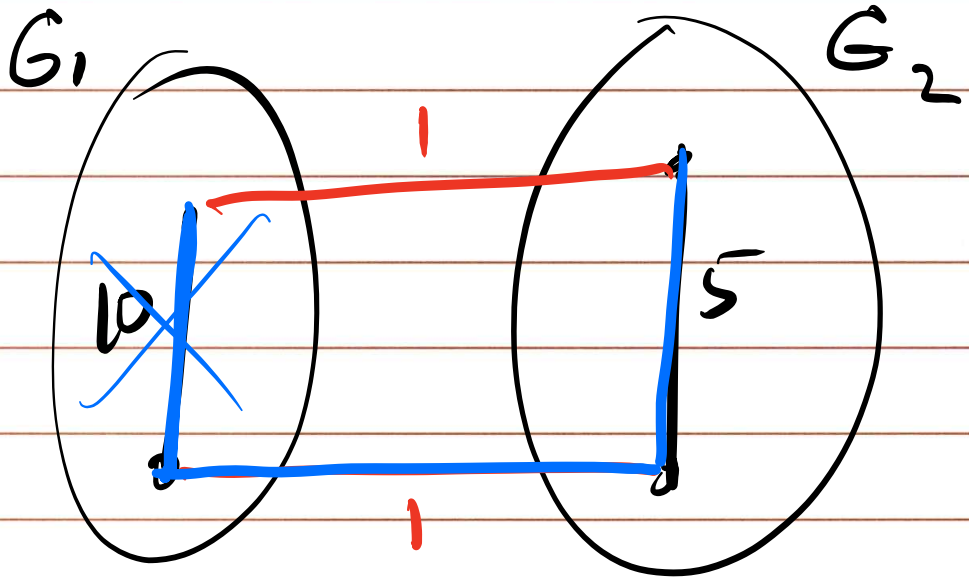
1. Suppose we have two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, along with T_1 which is a MST of G_1 and T_2 which is a MST of G_2. Now consider a new graph $G = (V, E)$ such that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup E_3$ where E_3 is a new set of edges that all cross the cut (V_1, V_2).

Consider the following algorithm, which is intended to find a MST of G.

Maybe-MST(T_1, T_2, E_3)
    e_{min} = a minimum weight edge in E_3
    $T = T_1 \cup T_2 \cup \{ e_{min} \}$
    return T

Does this algorithm correctly find a MST of G? Either prove it does or prove it does not.





2. Solve the following recurrences using the Master Method:

a. $A(n) = 3\,A(n/3) + 15$
b. $B(n) = 4\,B(n/2) + n^3$
c. $C(n) = 4\,C(n/2) + n^2$
d. $D(n) = 4\,D(n/2) + n$ X

a. $f(n) = \Theta(1)$, $n^{\log_b a} = n^{\log_3 3} = n^1$

Case # 1 $\longrightarrow$ $A(n) = \Theta(n)$

b. $f(n) = n^3$, $n^{\log_b a} = n^{\log_2 4} = n^2$

Case 3 ? $af(n/b) \leq cf(n)$ $c < 1$

$4(n/2)^3 \leq cn^3$

$n^3/2 \leq cn^3$ $.5 < c < 1$ ✓

Case 3 $\longrightarrow$ $B(n) = \Theta(n^3)$

C- $f(n) = n^2$

$n^{\log_b a} = n^{\log_2 4} = n^2$

Case 2 $\Rightarrow C(n) = \Theta(n^2 \lg n)$

$C'(n) = 4C'(n/2) + n^2 \lg^2 n$

$n^{\log_b a} = n^2 \qquad f(n) = n^2 \lg^2 n$

$C'(n) = \Theta(n^2 \lg^3 n)$

3. There are 2 sorted arrays A and B of size n each. Design a D&C algorithm to find the median of the array obtained after merging the above 2 arrays (i.e. array of length 2n). Discuss its runtime complexity.

$A = (5, 12, 15, 19, 22, 25, 30)$

$B = (8, 9, 18, 27, 30, 41, 50)$

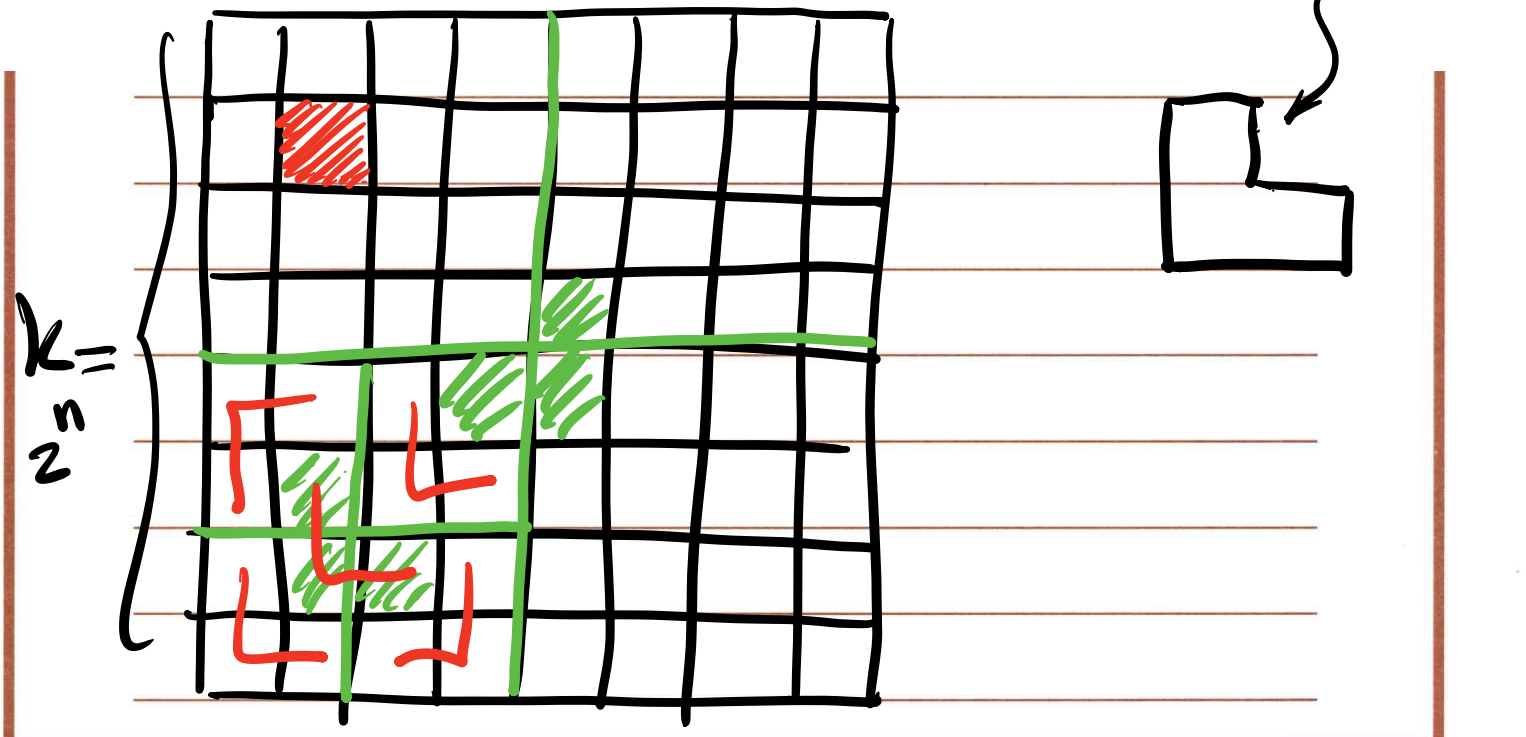
$f(n) = \Theta(1)$

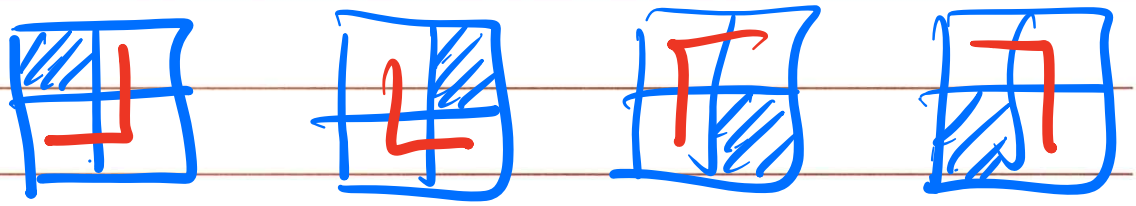
$n^{\log_b a} = n^{\log_2 1} = n^0 = \Theta(1)$

Case #2 →

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

4. A tromino is a figure composed of three 1x1 squares in the shape of an L. Given a 2^nx2^n checkerboard with 1 missing square, tile it with trominoes. Design a D&C algorithm and discuss its runtime complexity.





$f(n) = \Theta(1)$

$k^{\log_b a} = k^{\log_2 4} = k^2$

Case #1 $\rightarrow T(k) = k^2$