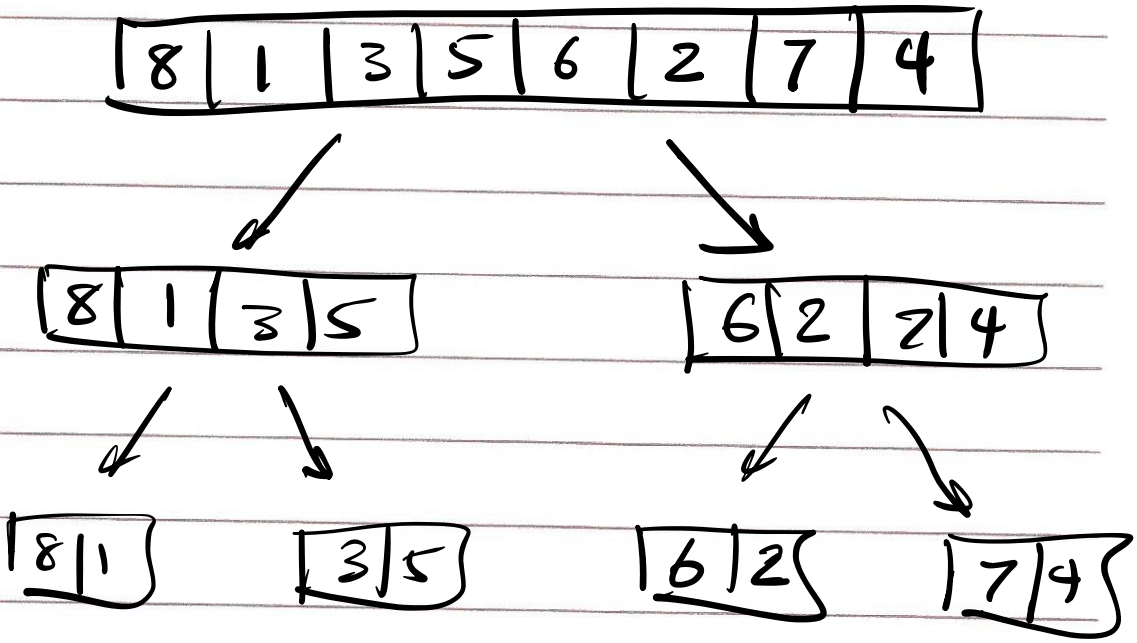
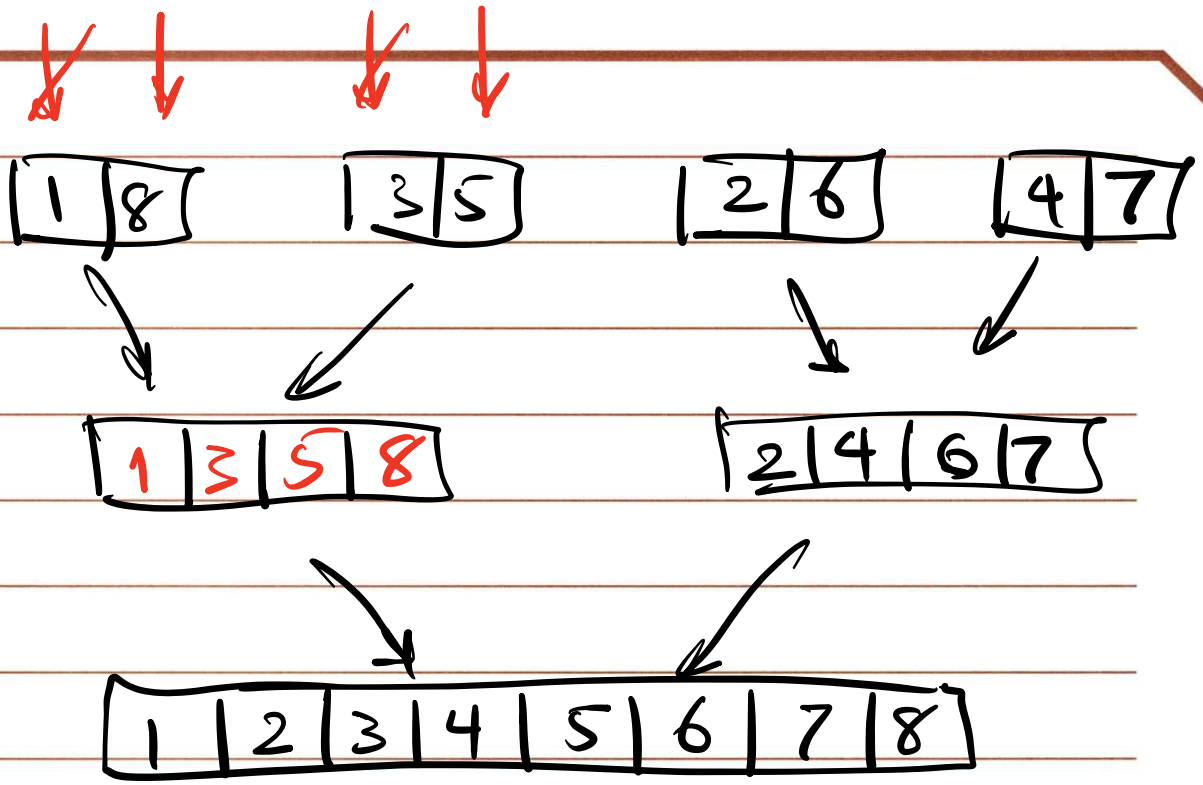
Divide & Conquer

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




1 8
3 5
2 6
4 7
1 3 5 8
2 4 6 7
1 2 3 4 5 6 7 8

```
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 size of n.

Recurrence equation for Merge Sort

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

Conquer

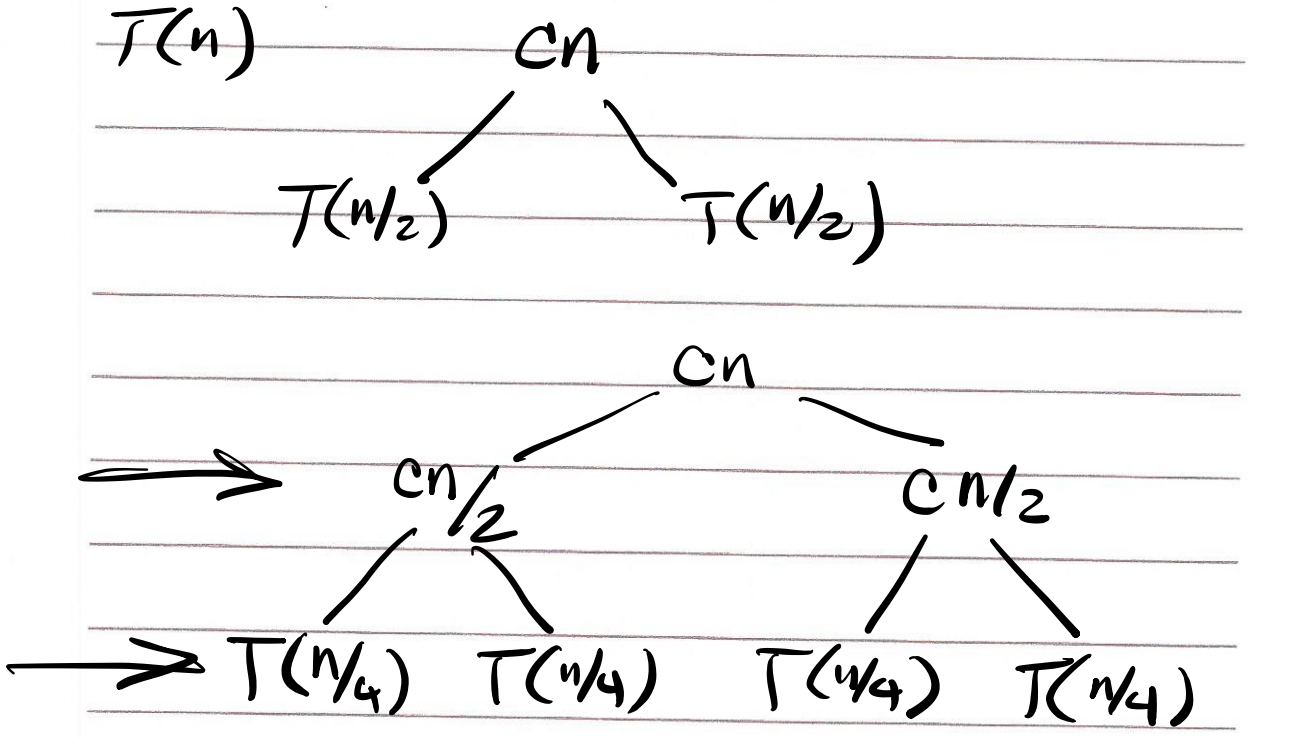
Combine

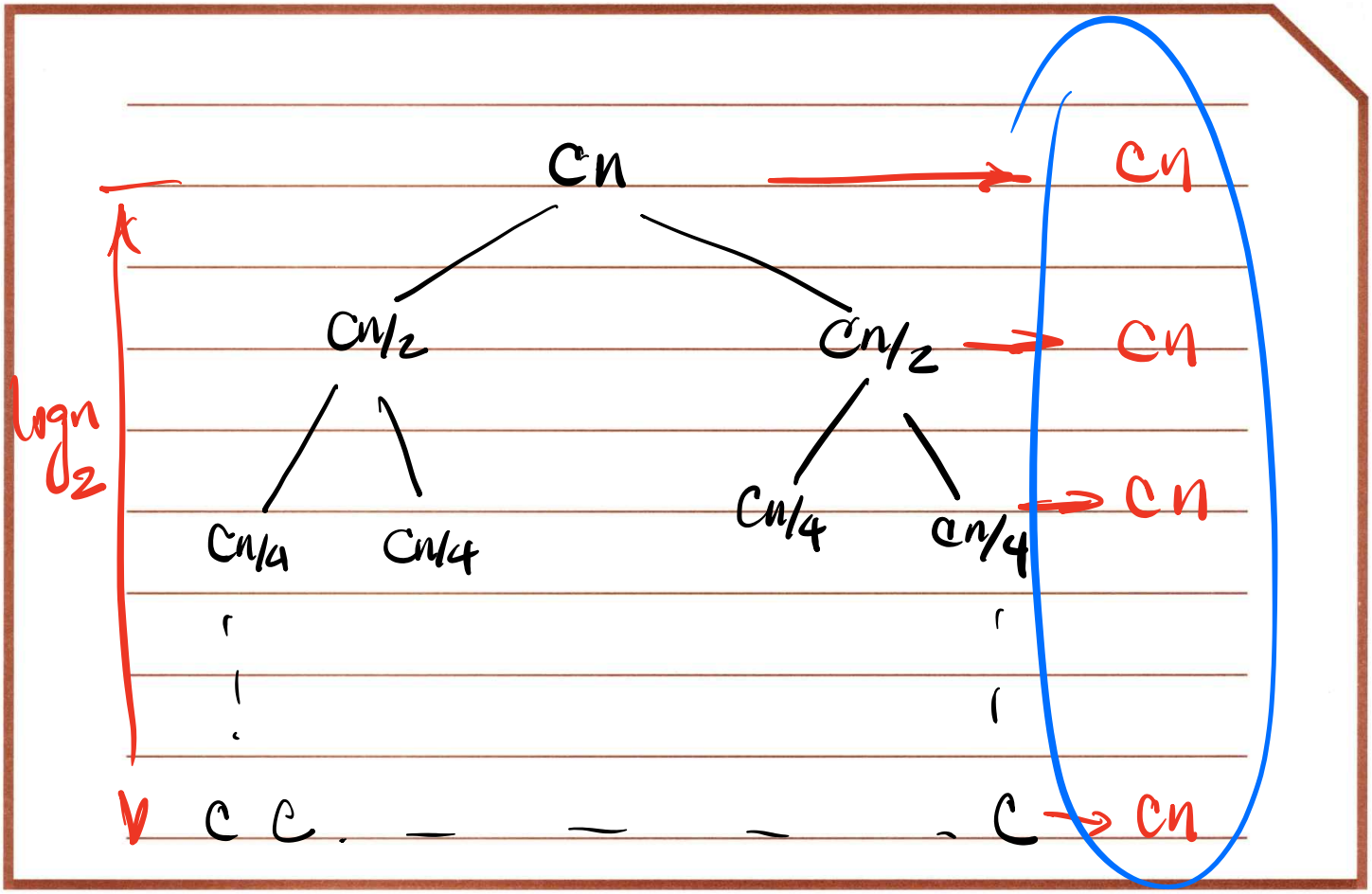
Divide

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

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

- a: # of subproblems
- n/b: size of the subproblems
- $D(n)$: time to divide
- $C(n)$: time to combine





total cost = cn log n

Worst Case Complexity = O(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

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

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

Case#3

$f(n) = n^{1.0001}$

$n^{\log_b a} = n$

$f(n) = n \lg n$

$n^{\log_b a} = n$

Diff is NOT polynomial

This is not a Case#3

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

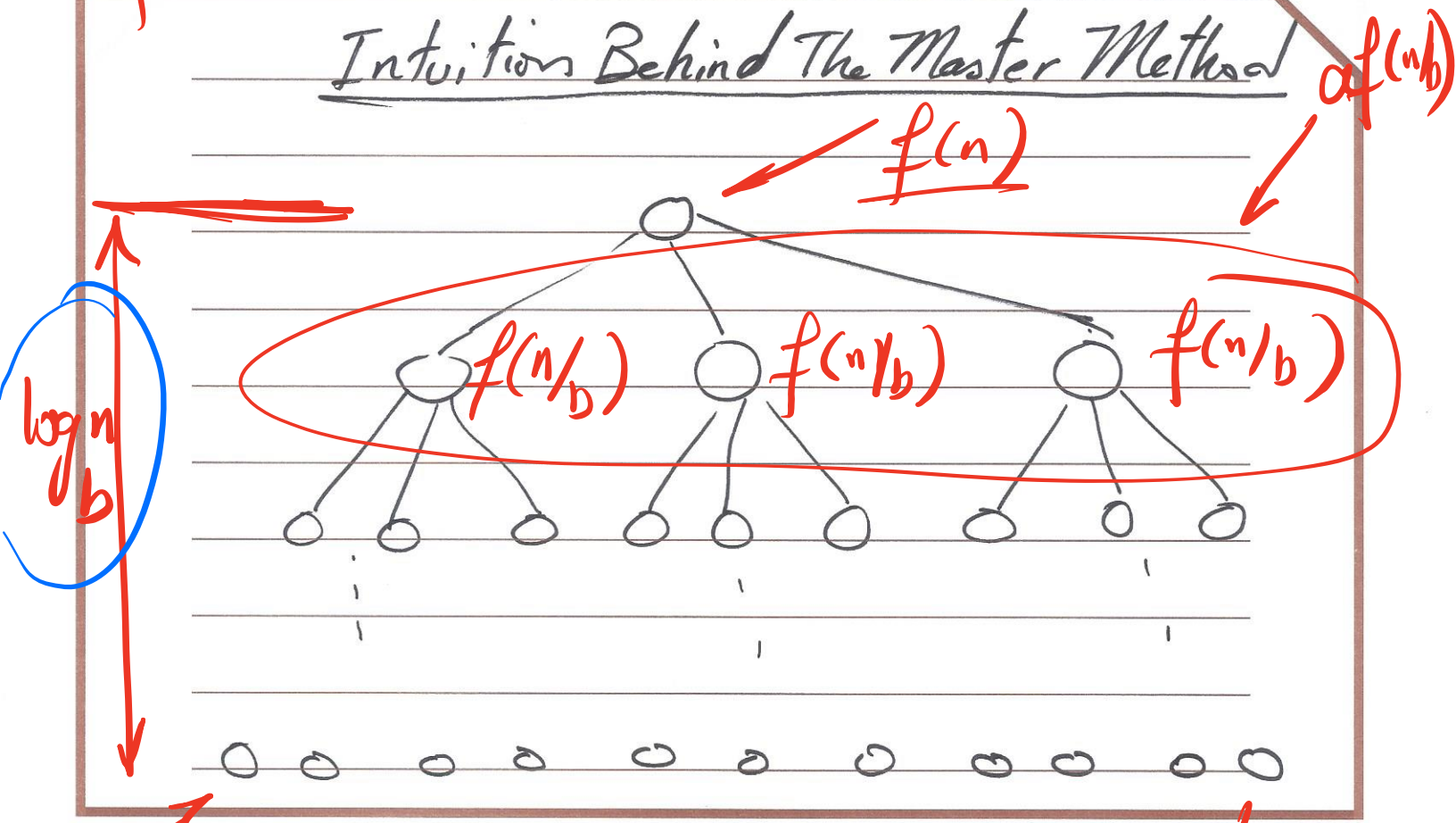
where $k \geq 0$

Then

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

$f(n) + \alpha f(n) + \alpha^2 f(n) \Rightarrow \Theta(f(n))$

Intuition Behind The Master Method



# of nodes at the bottom of tree $= a^{\log_b n}$

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

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

$$c\, n^{\log_b q} = \Theta(n^{\log_b q})$$

$$\sum x + \alpha x + \alpha^2 x + \cdots = cx$$

if $\alpha < 1$ $\Rightarrow$

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.

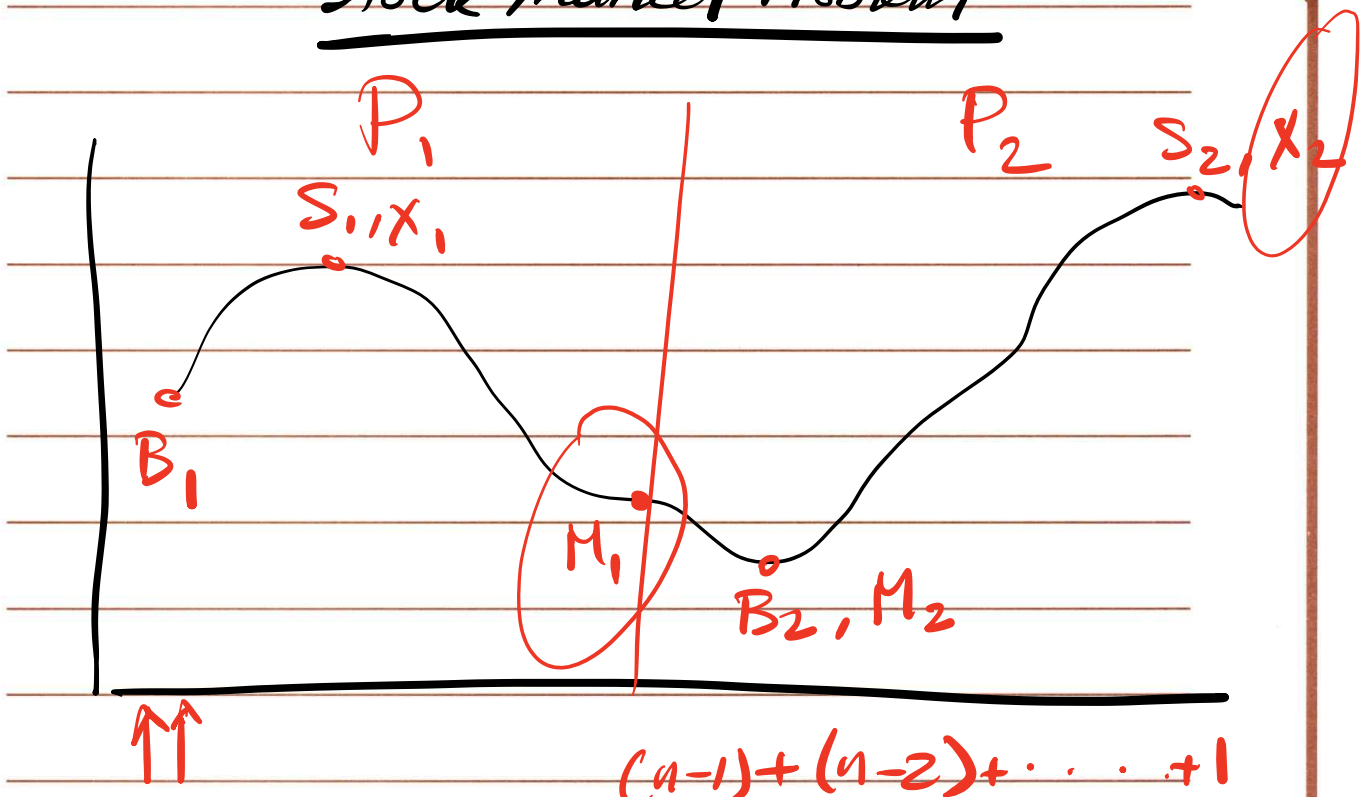
Complexity of Merge Sort

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

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

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

Stock Market Problem



Case #1

Buy in P_1 & Sell in P_1

$B = B_1$ $\quad M = Min(M_1, M_2)$

$S = S_1$ $\quad X = Max(X_1, X_2)$

Case #2

Buy in P_2 & Sell in P_2

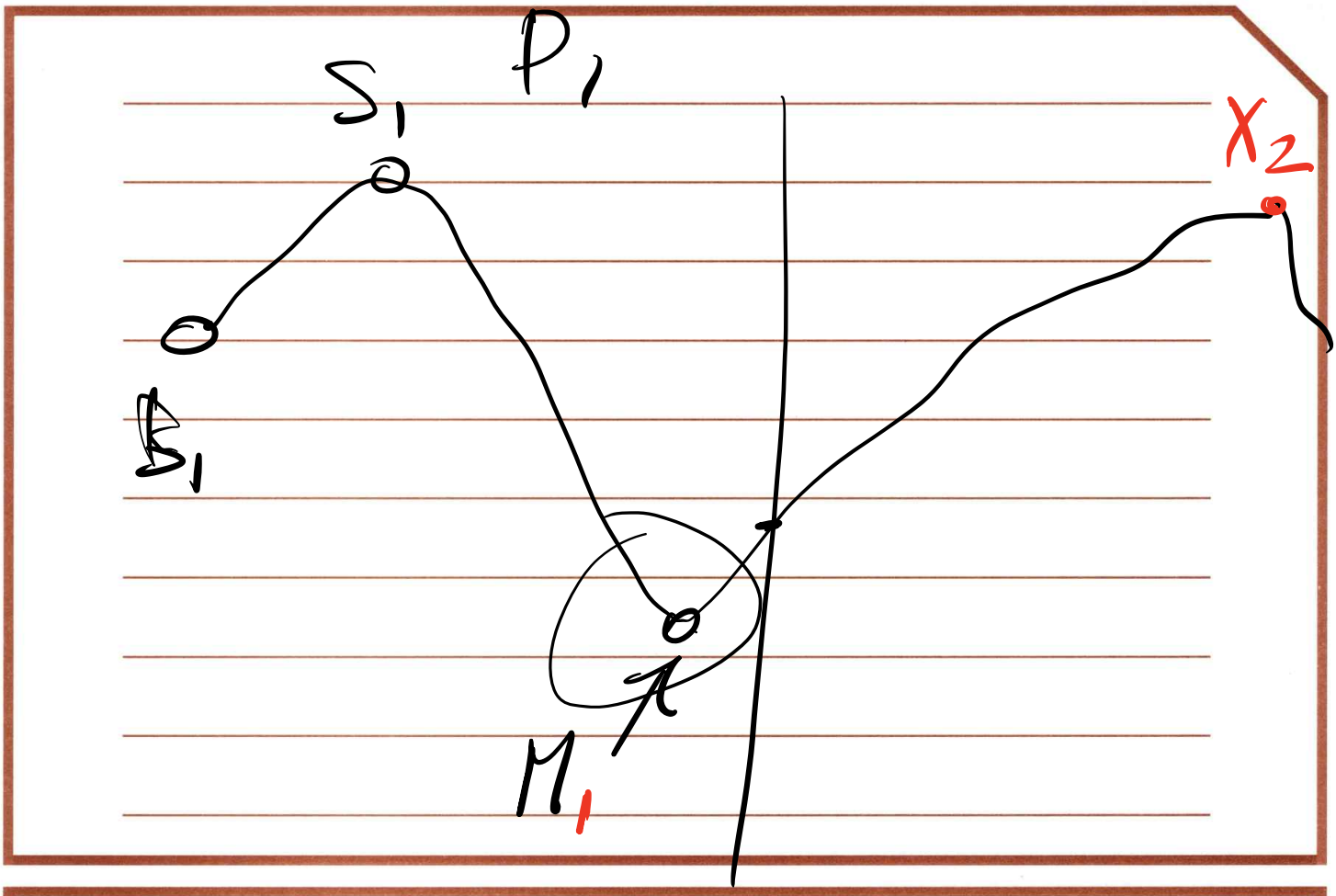
$B = B_2$ $\quad M = Min(M_1, M_2)$

$S = S_2$ $\quad X = Max(X_1, X_2)$

Case #3 Buy in P_1 & Sell in P_2

$B = B_1$ $\quad M = Min(M_1, M_2)$

$S = S_2$ $\quad X = Max(X_1, X_2)$



$\Theta(1)$ $\Theta(1)$

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

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

→ Case # 1 → $T(n) = \Theta(n)$

Dense Matrix Multiplication

$$\underbrace{\left[\underbrace{A}_{n}\right]\left[\underbrace{B}_{n}\right]}_{n} = \underbrace{\left[\underbrace{C}_{n}\right]}_{n}$$

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

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

Combine step

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

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

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

case #1 $\longrightarrow T(n) = \Theta(n^3)$

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

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

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

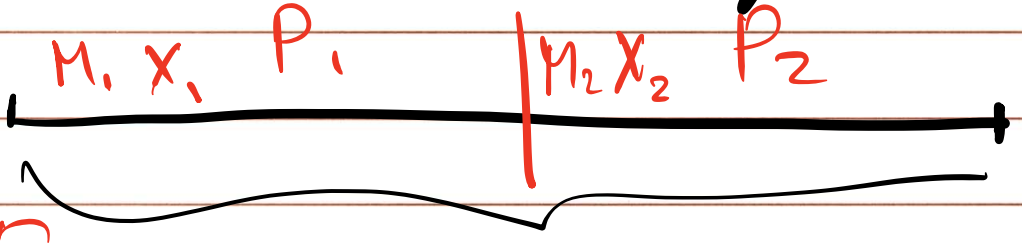
Strassen's Method

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

$f(n) = \Theta(n^2)$

Case #1 $\rightarrow T(n) = n^{2.81}$

Finding Min & Max in an unsorted array

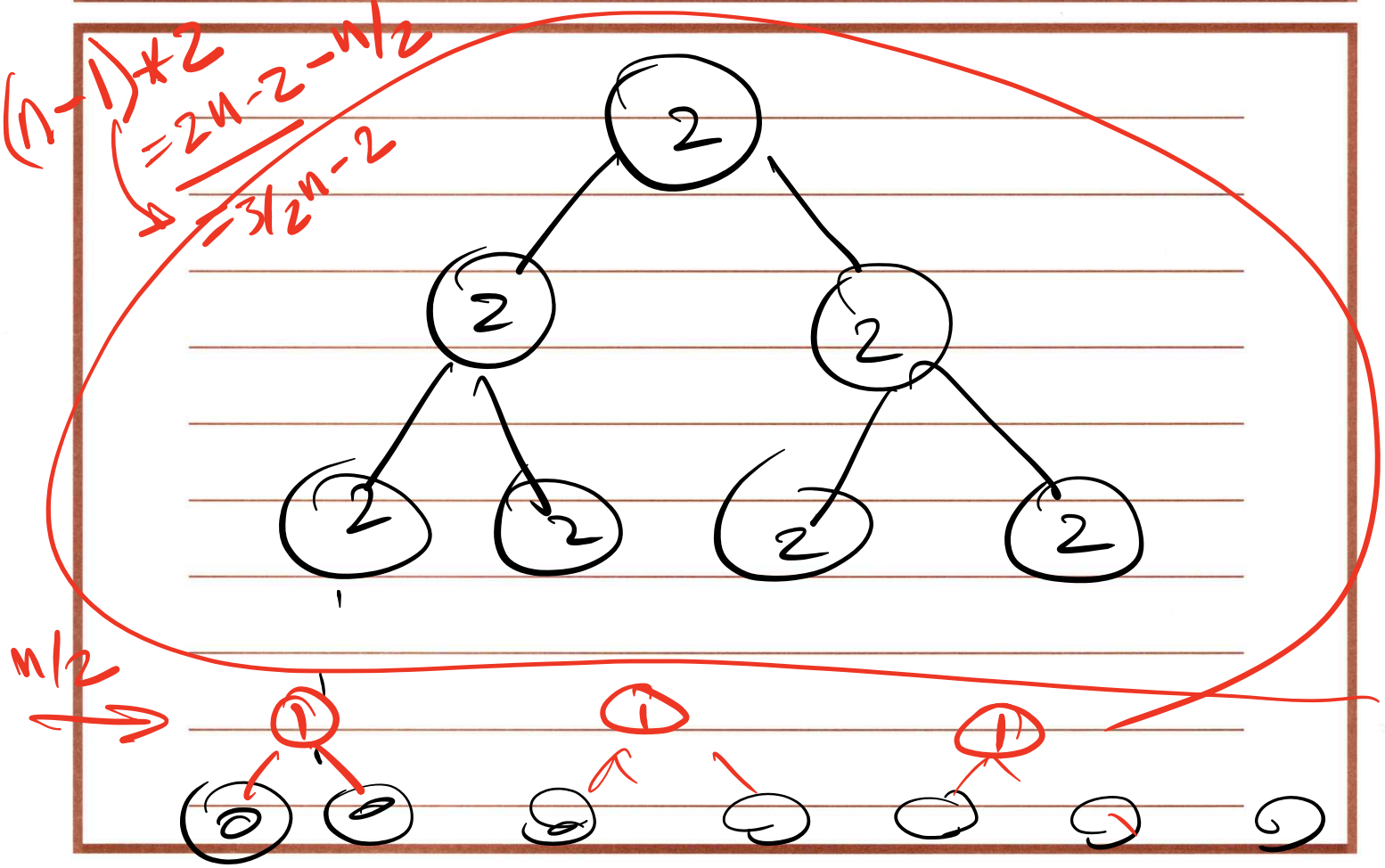


Brute Force:

# of comparisons to find Min = n-1

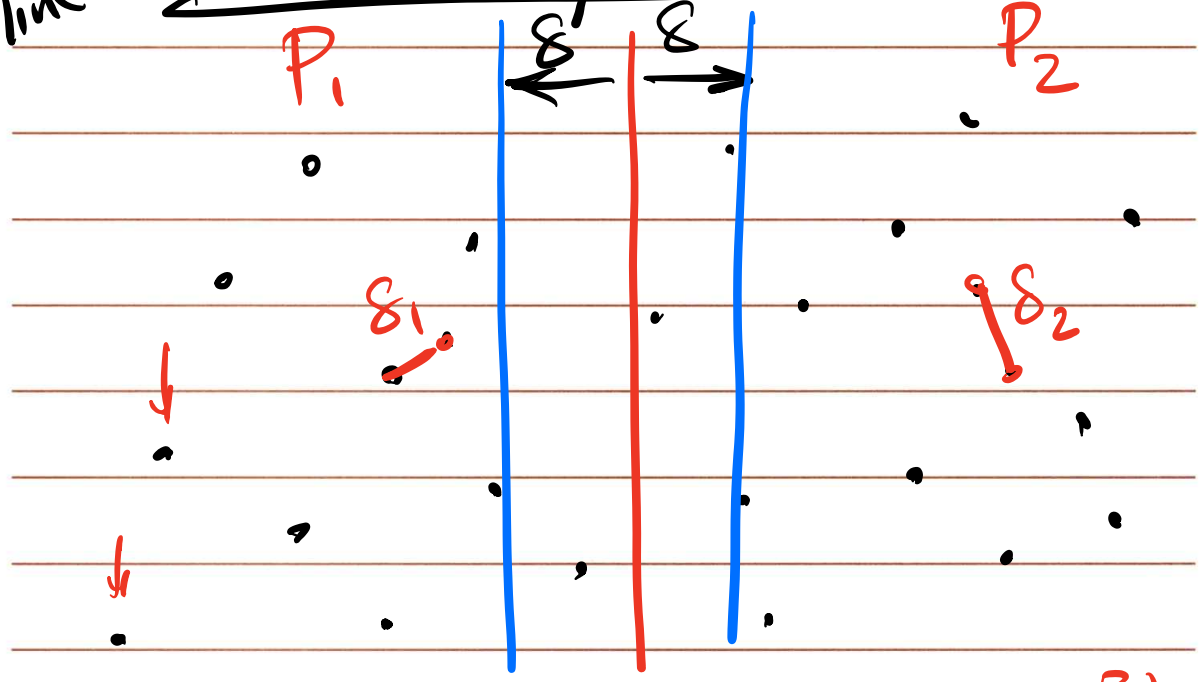
" " " " Max = n-1

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



Closest pair of points problem (2D)

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

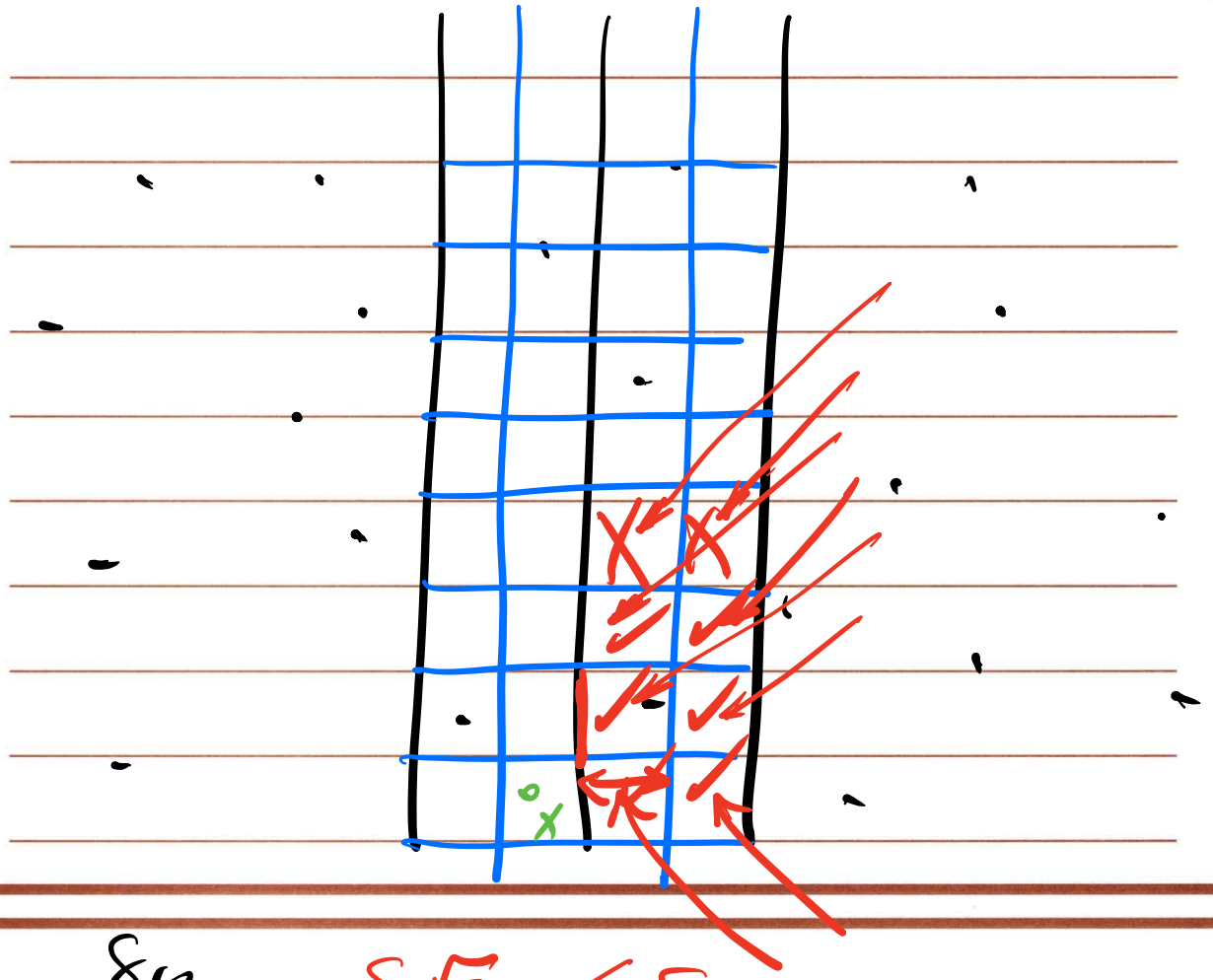


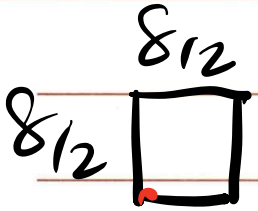
Brute force $\rightarrow (n-1) + (n-2) + \cdots + 1 = \Theta(n^2)$

Case #1 : Both pts are in P_1 ✓

Case #2 : " " " P_2 ✓

Case #3 : one pt is in P_1 & other in P_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

$O(n)$

Divide

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

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

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

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

(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.

$O(n)$ 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 $d(s, s') < \delta$ then
Return (s, s')
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

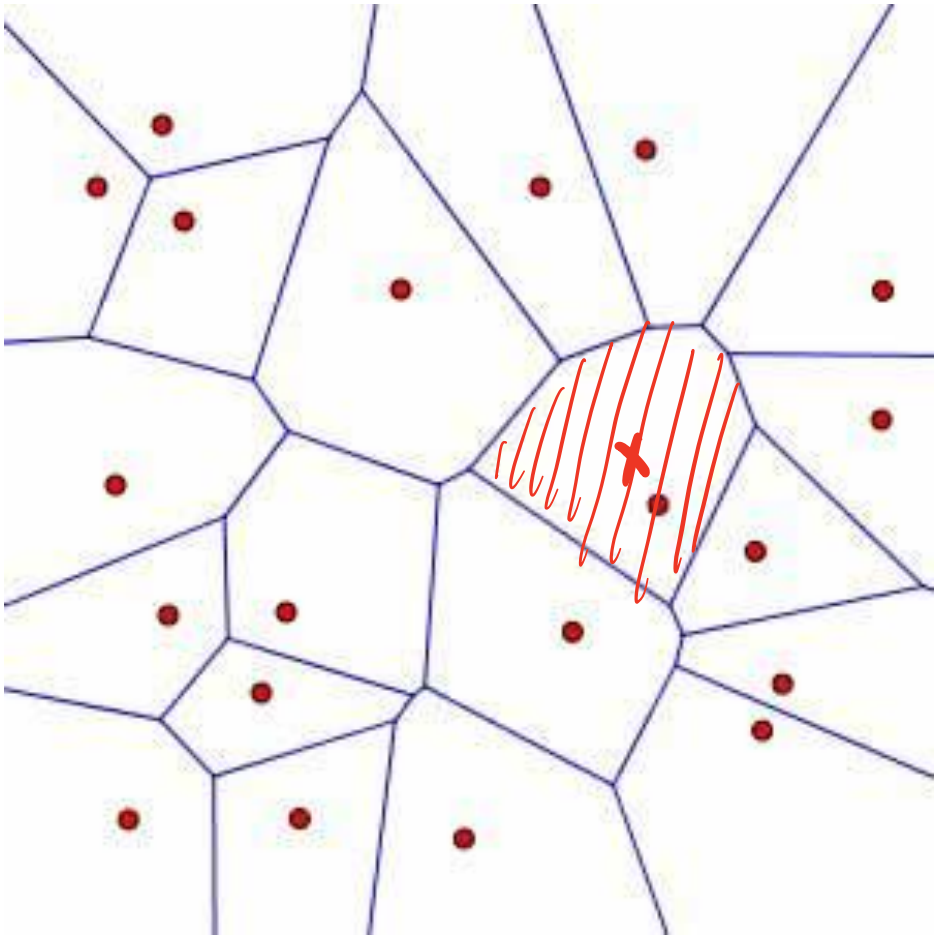
overall complexity:

$f(n) = O(n^{\log_2 2}) = n$

$\log_b n = \log_2 n$

case 2 $\rightarrow$ $T(n) = O(n \log n)$

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.

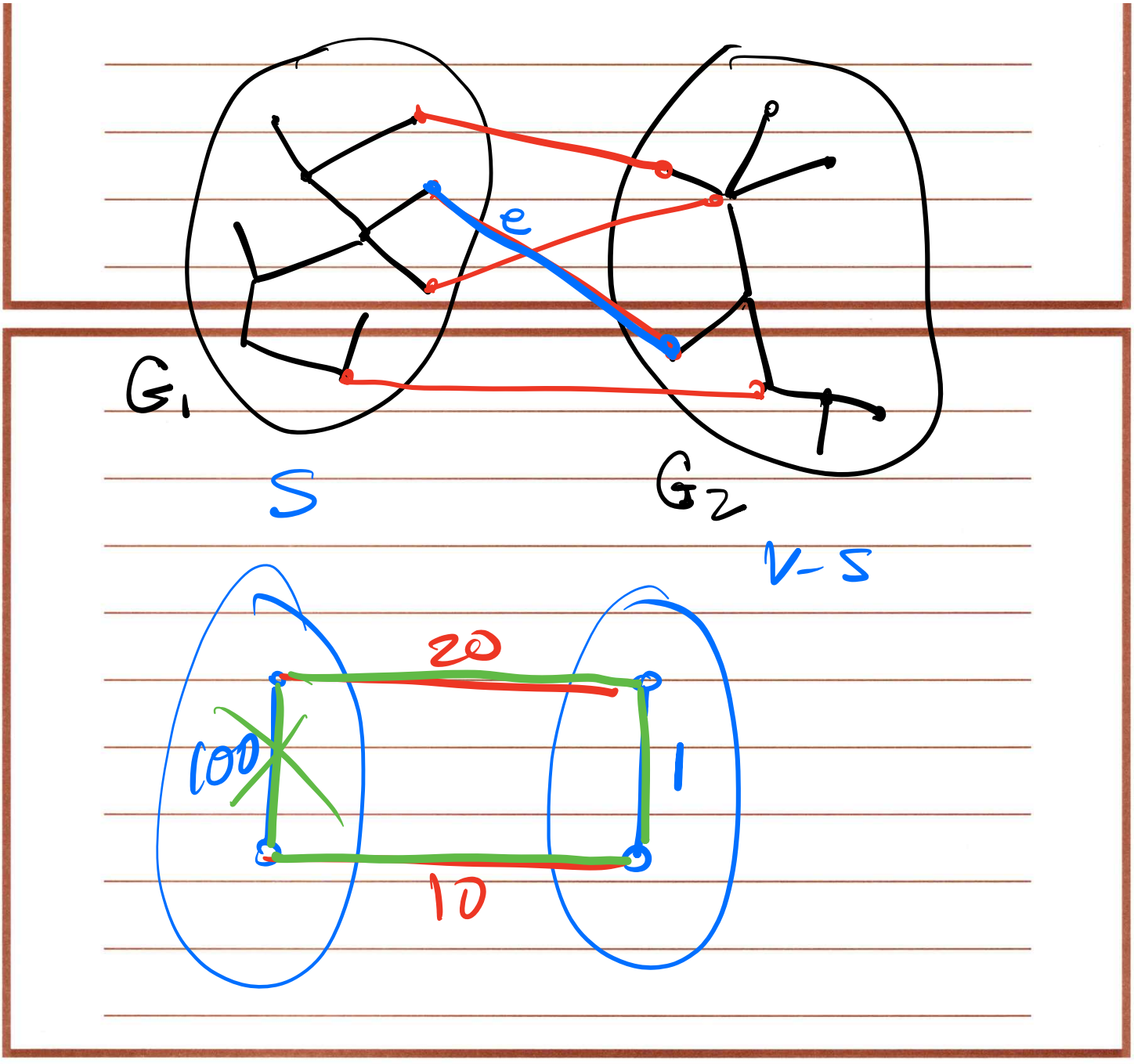
4. A tromino is a figure composed of three 1x1 squares in the shape of an L.
Given a $2^n x 2^n$ checkerboard with 1 missing square, tile it with trominoes.
Design a D&C algorithm and discuss its runtime complexity.

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. $D(n) = 4\,D(n/2) + n$

$A(n) = 3A(n/3) + 15$

$f(n) = 15 = \Theta(1)$

$n^{\log_b a} = n^{\log_3 3} = n^1 = n$ → Case #1

$A(n) = \Theta(n)$

$B(n) = 4B(n/2) + n^3$

$f(n) = n^3$

$n^{\log_b a} = n^{\log_2 4} = n^2$ → Case 3

$a f(n/b) < c f(n)$ for $c < 1$

$4 \cdot \left(\frac{n^3}{8}\right) < cn^3$

$n^3/2 < cn^3$

$.5 < c < 1$ ✓

$\rightarrow B(n) = \Theta(n^3)$

$$C(n) = 4C(n/2) + n^2$$

$f(n) = n^2$

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

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

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

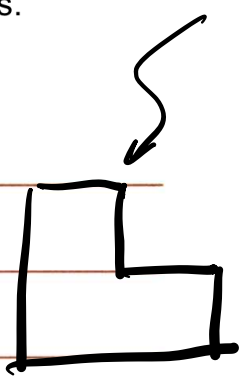
$f(n) = n^2 \lg^2 n$

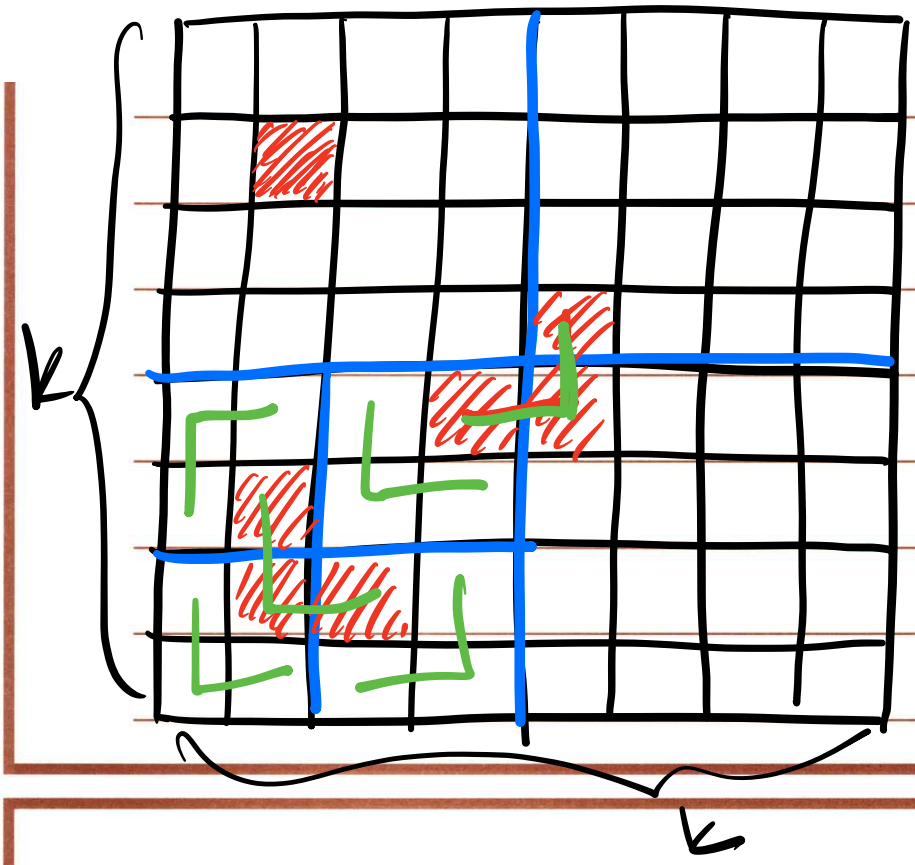
$n^{\log_b a} = n^2$

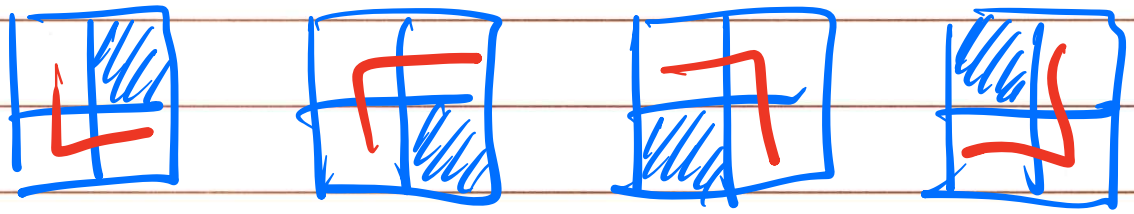
$\rightarrow$ General Case #2

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

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.







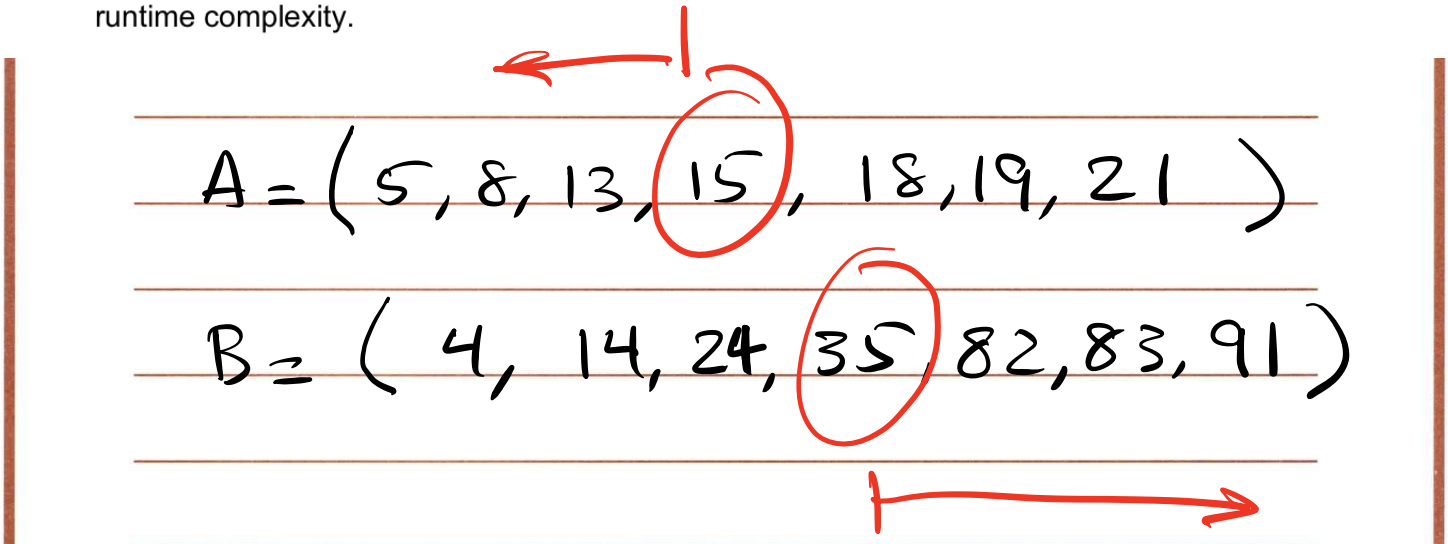
$$f(k) = \Theta(1)$$

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

Case #1 $\Rightarrow$ $T(n) \in \Theta(k^2)$

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.



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

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

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

Case #2 $\Rightarrow T(n) = \Theta(\lg n)$