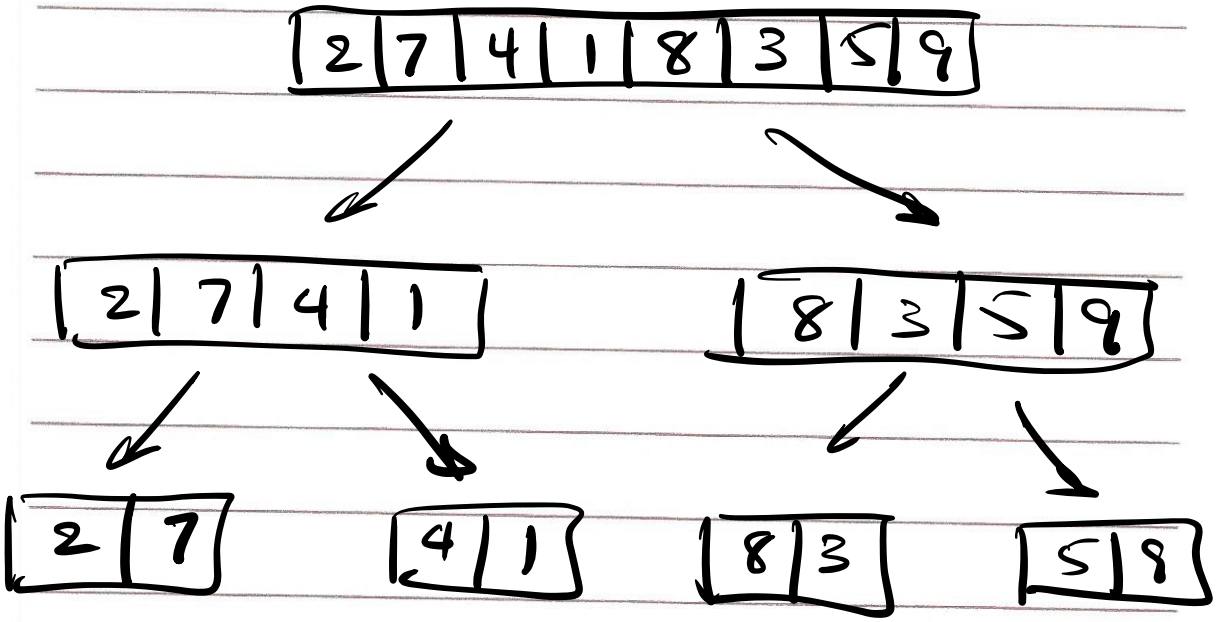
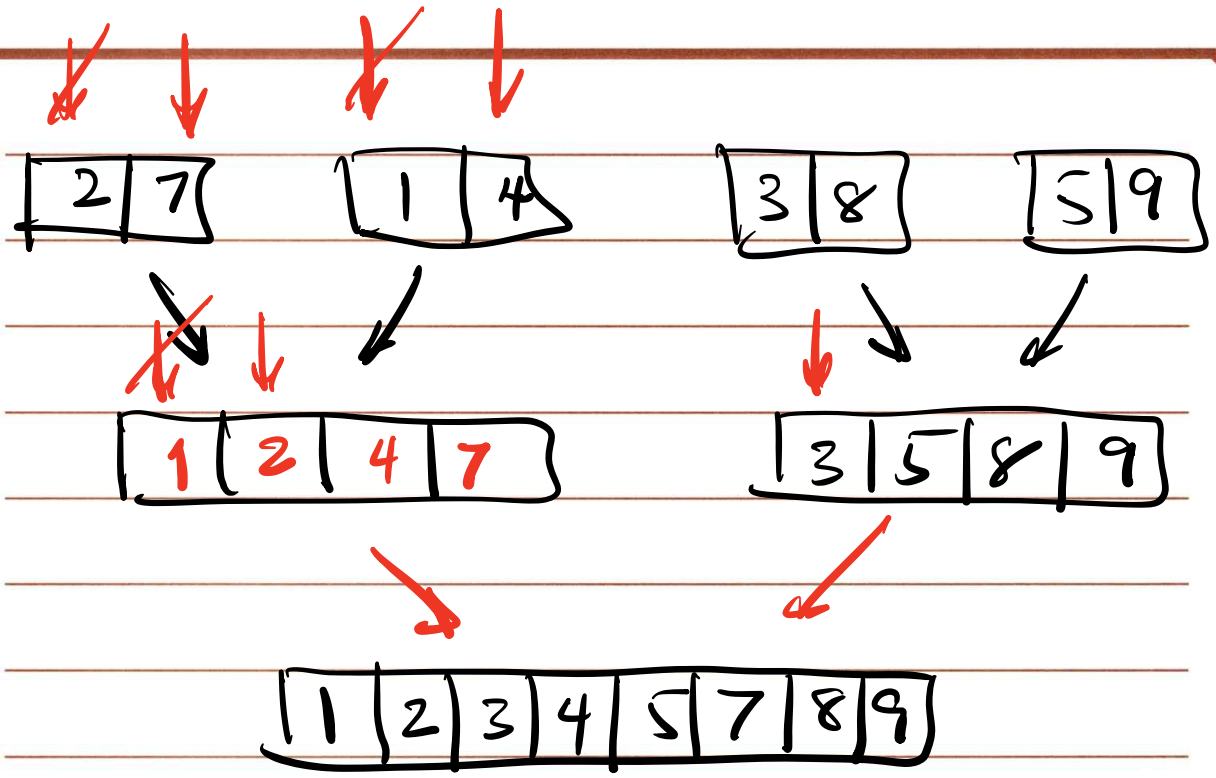
Divide & Conquer

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          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 Eq. for mergesort:

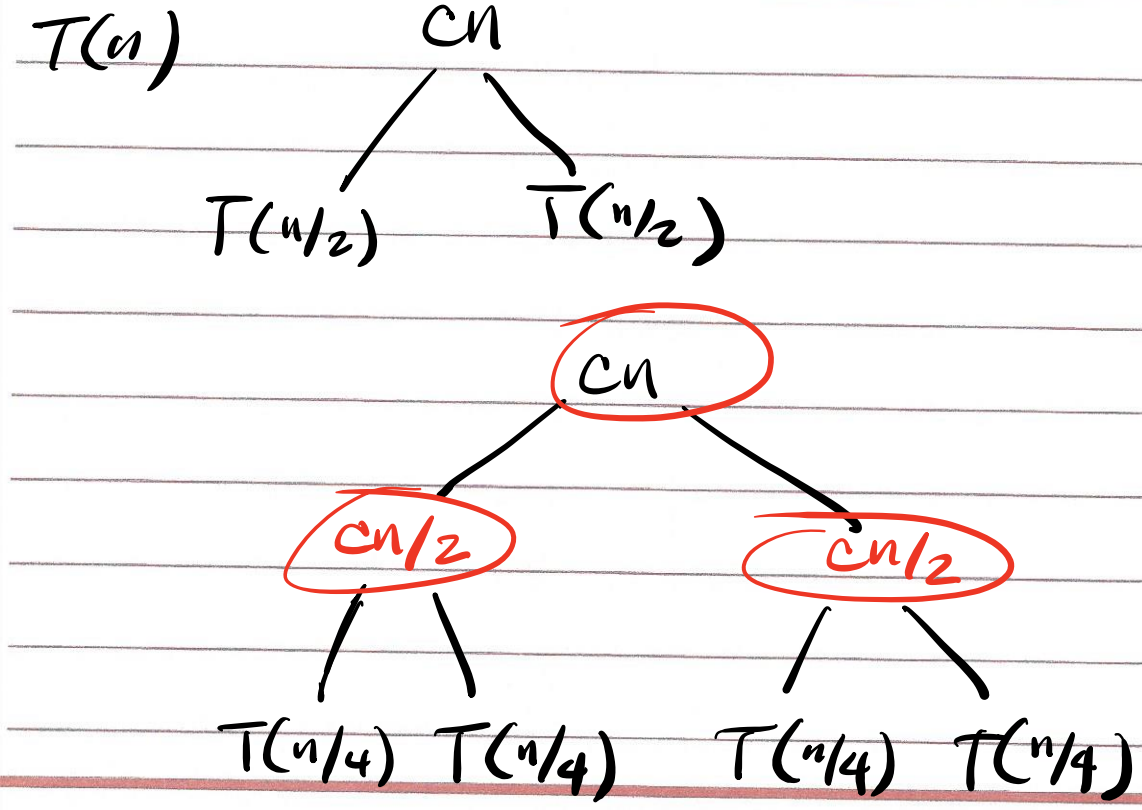
$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2 \cdot T(n/2) + O(n) + O(1) & \end{cases}$$

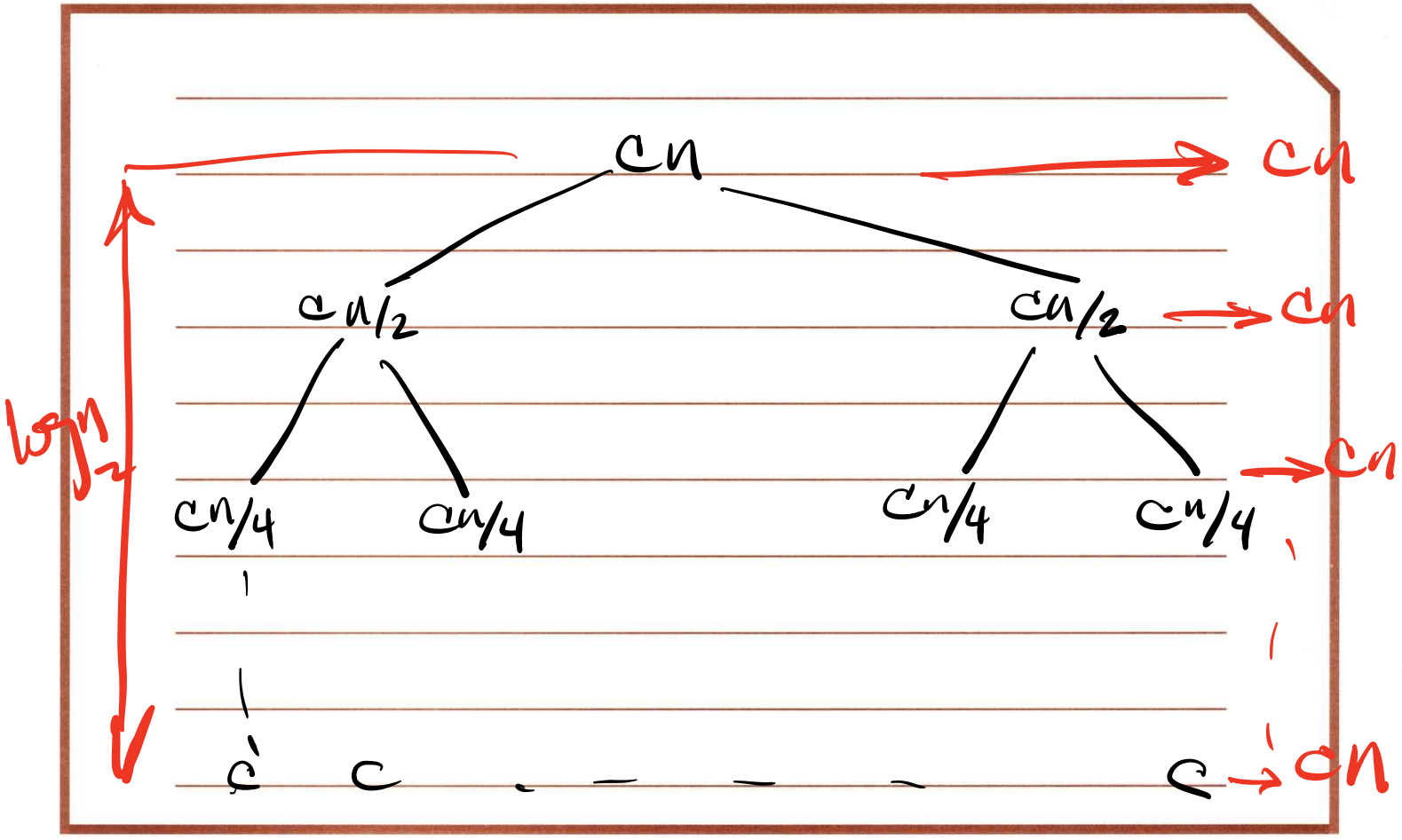
Conquer — Combine — Divide

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

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

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





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

Master Method

It is a cookbook method for solving recurrences of the form

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

D(n) + C(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

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

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

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

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

General version of Case#2

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

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

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

Case#3

$T(n) = \Theta(n^{1.0001})$

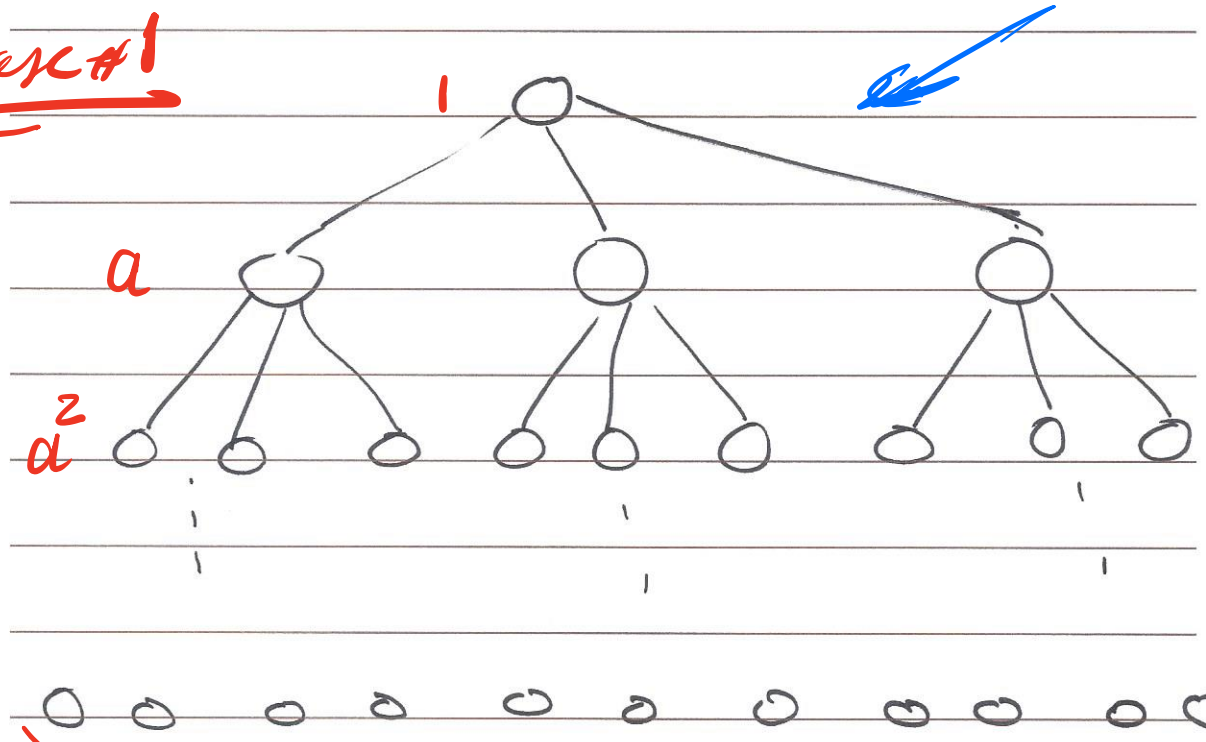
in general if $f(n) = \Theta(n^{\log_b a} \lg^k n)$ where $k \geq 0$

Then

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

Intuition Behind The Master Method

Case #1



$a^{\log_b n}$ nodes at the very bottom

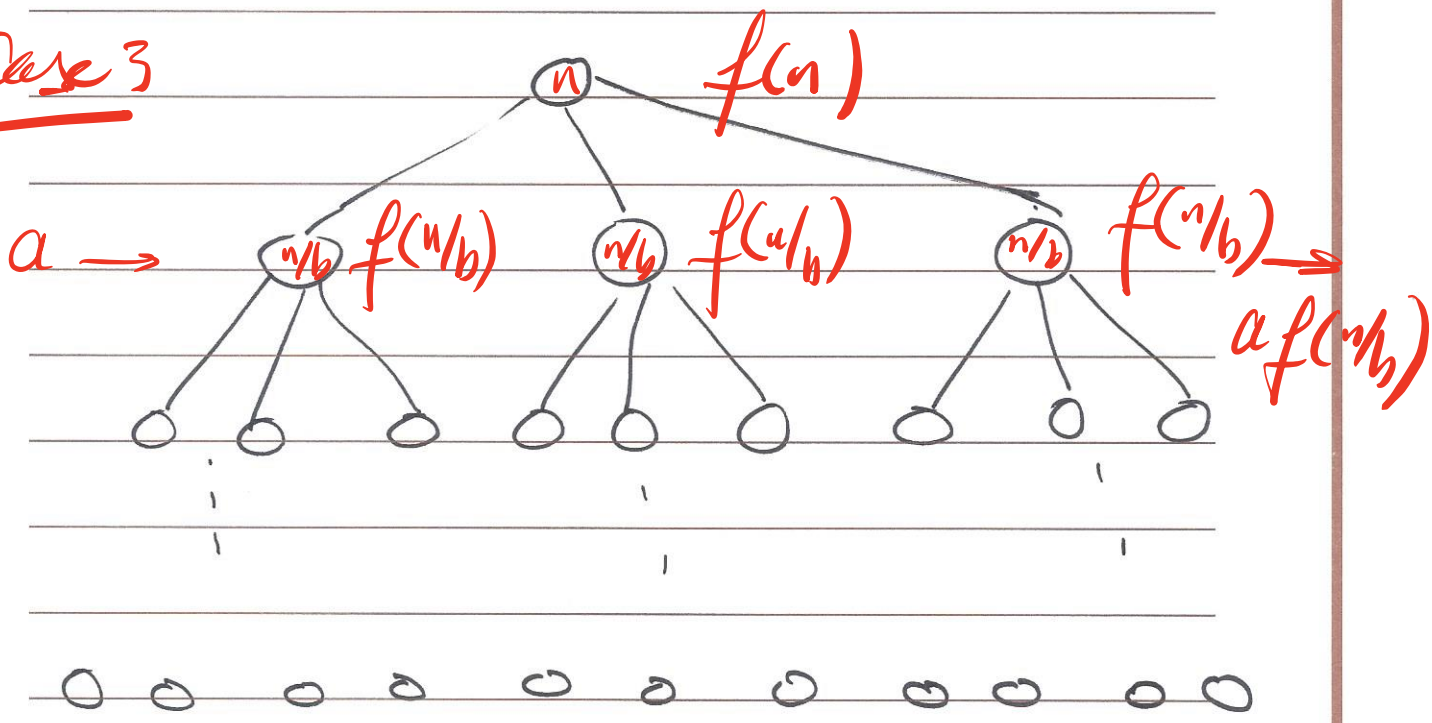
$= n^{\log_b a}$

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

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

Intuition Behind The Master Method

Case 3



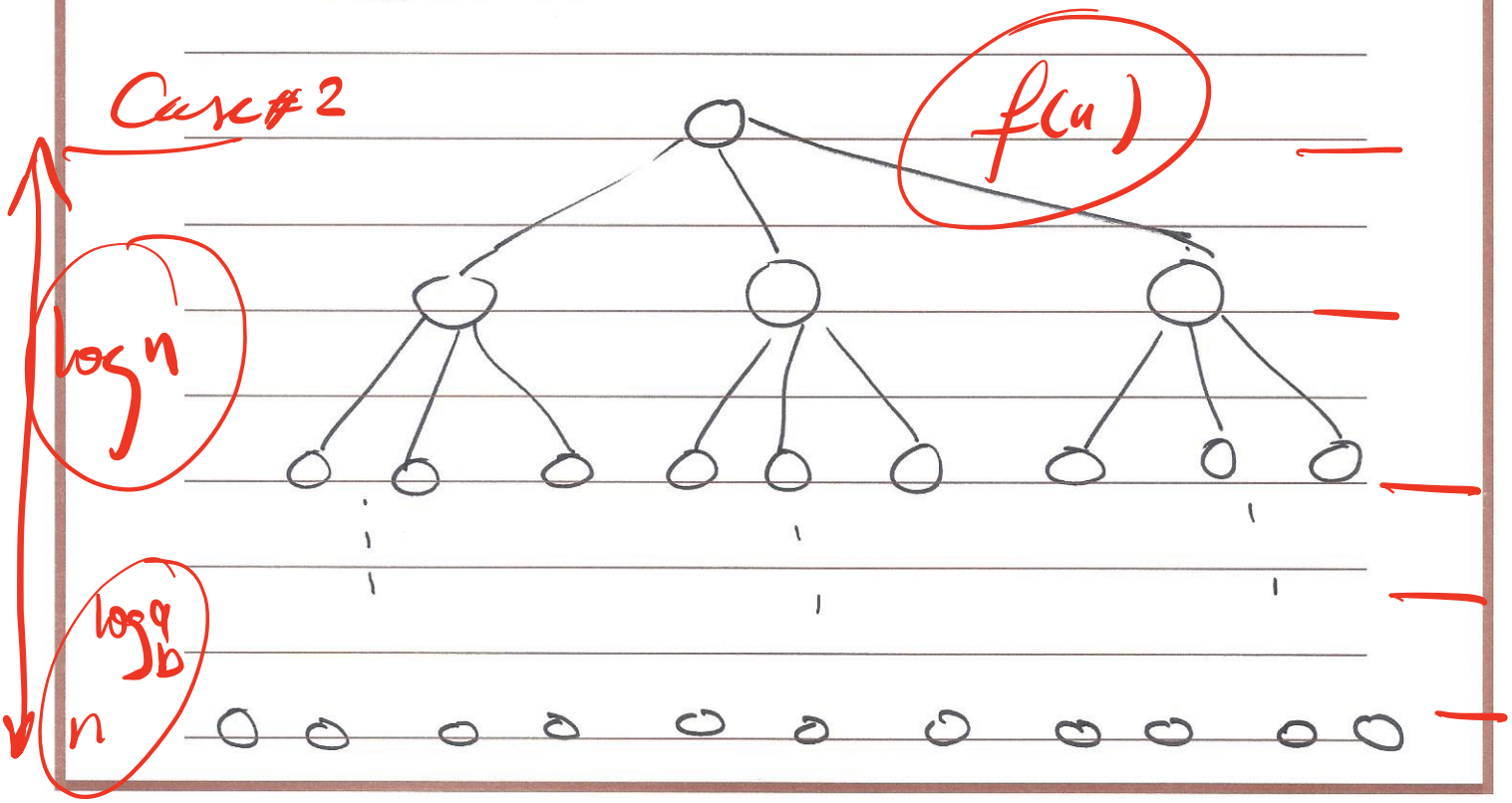
if $a f(n/b) \leq c f(n)$ for $c < 1$

then cost of Combine + Divide at the next level must be a fraction of that in the previous level

$$f(n) + \alpha f(n) + \alpha^2 f(n) + \cdots = \Theta(f(n))$$

$\alpha < 1$

Intuition Behind The Master Method



cost at each level is the same.

we have $\lg n$ levels each costing $n^{\log_b a}$ $(= f(n))$

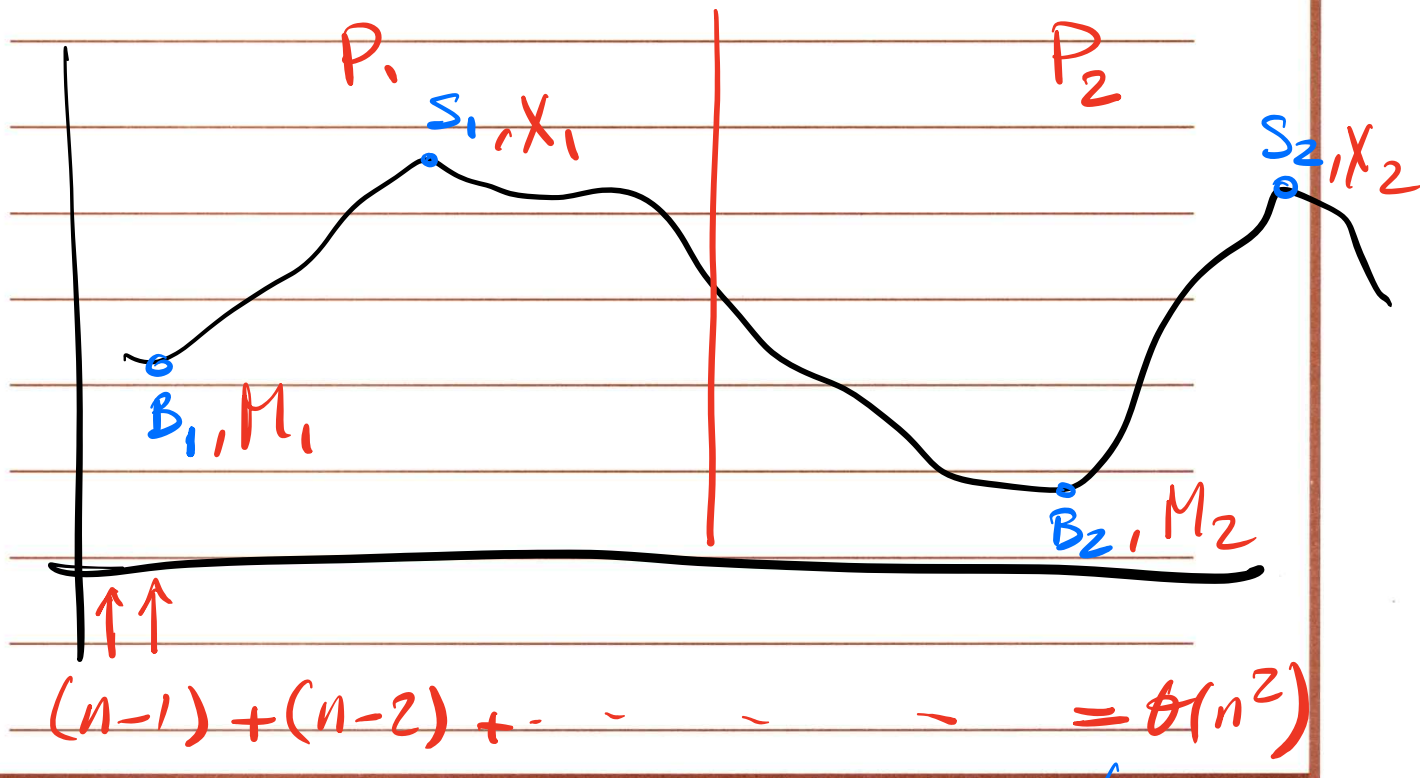
total cost $= \Theta(n^{\log_b a} \lg n)$

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



Case#1 buy & sell in P_1

$B = B_1$

$S = S_1$

Case#2 buy & sell in P_2

$B = B_2$

$S = S_2$

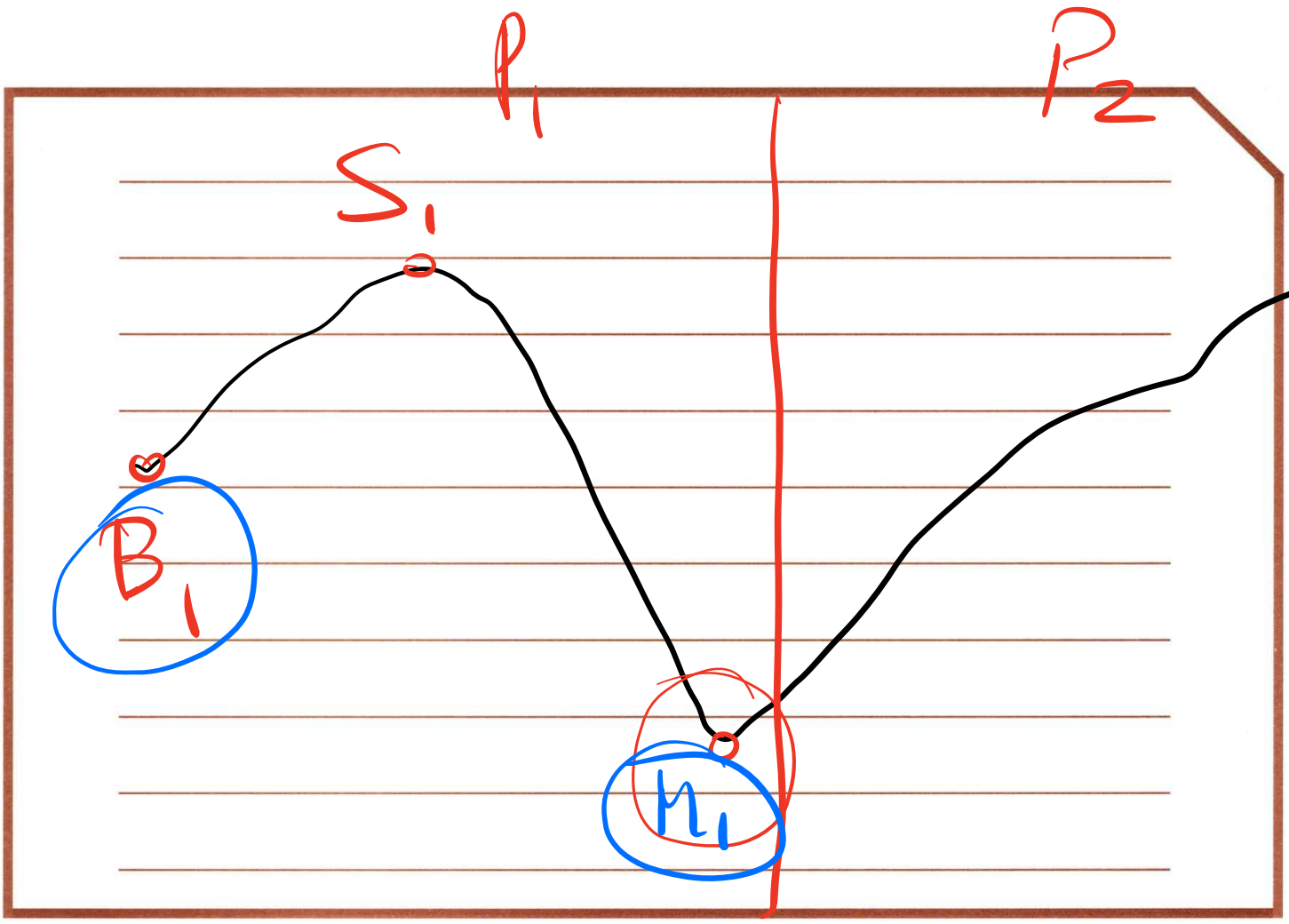
Case#3 buy in P_1 & sell in P_2

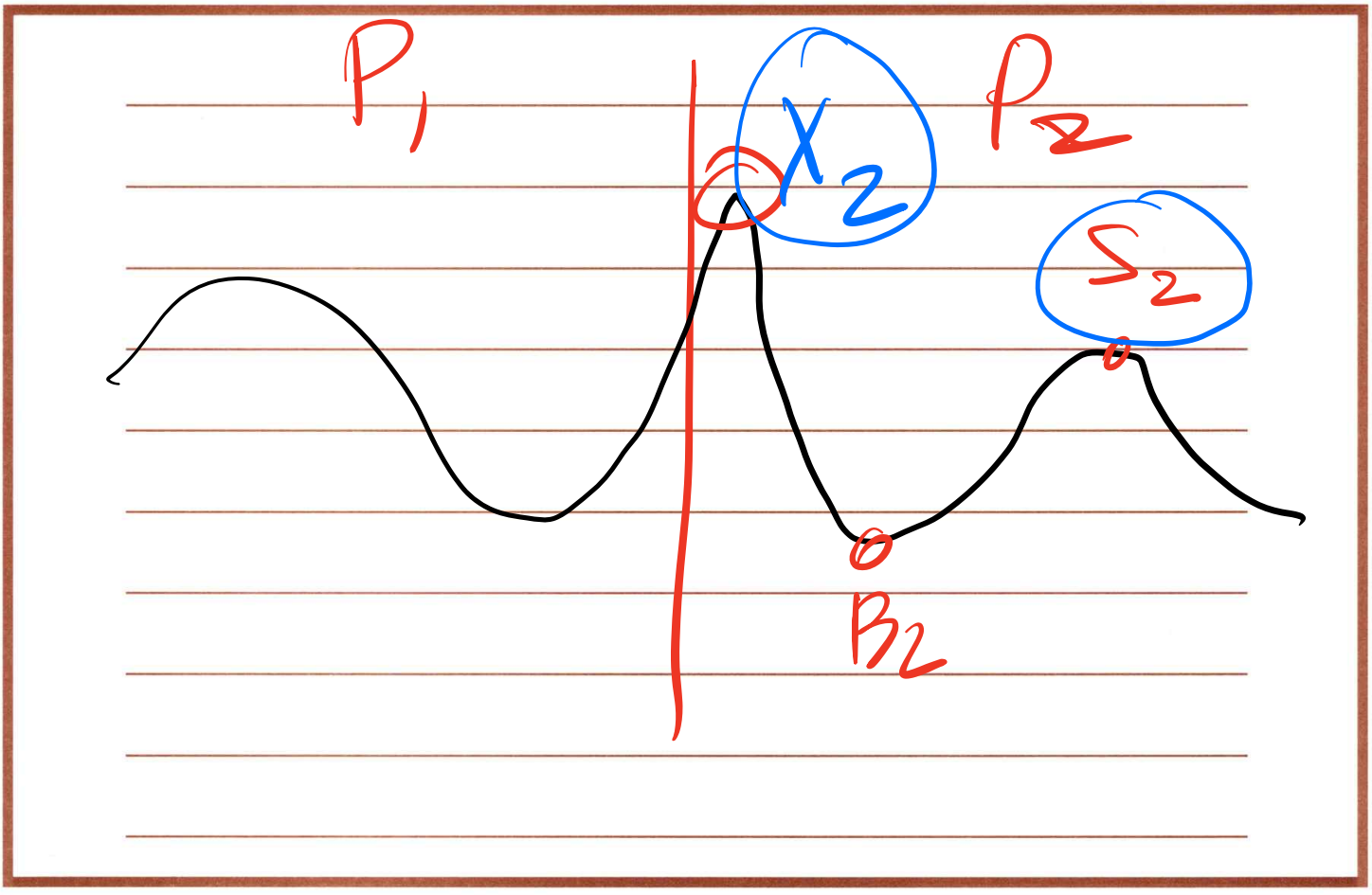
$B = B_1$ ~~B_1~~ M_1

$S = S_2$ ~~S_2~~ X_2

$M = Min(M_1, M_2)$

$X = Max(X_1, X_2)$


P1
P2
S1
B1
M1


P1
X2
P2
S2
B2

total cost of the alg.

$a = 2$

$b = 2$

$\log_b a = \log_2 2 = 1$

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

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

$\Rightarrow$ case 1

$T(n) = \theta(n)$

Dense Matrix Multiplication

takes $\Theta(n)$

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

Brute force $\rightarrow T(n) = \Theta(n^3)$

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

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

$a = 8$

$b = 2$

$f(n) = D(n) + C(n)$

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

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

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

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

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

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

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

Strassen's Alg.

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

Complexity =

$a = 7$

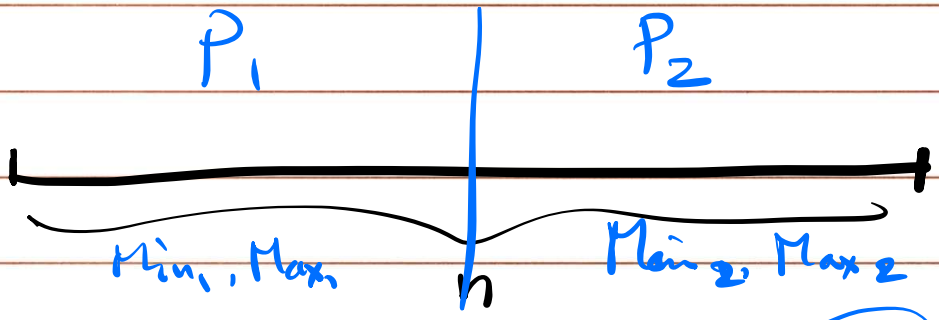
$b = 2$

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

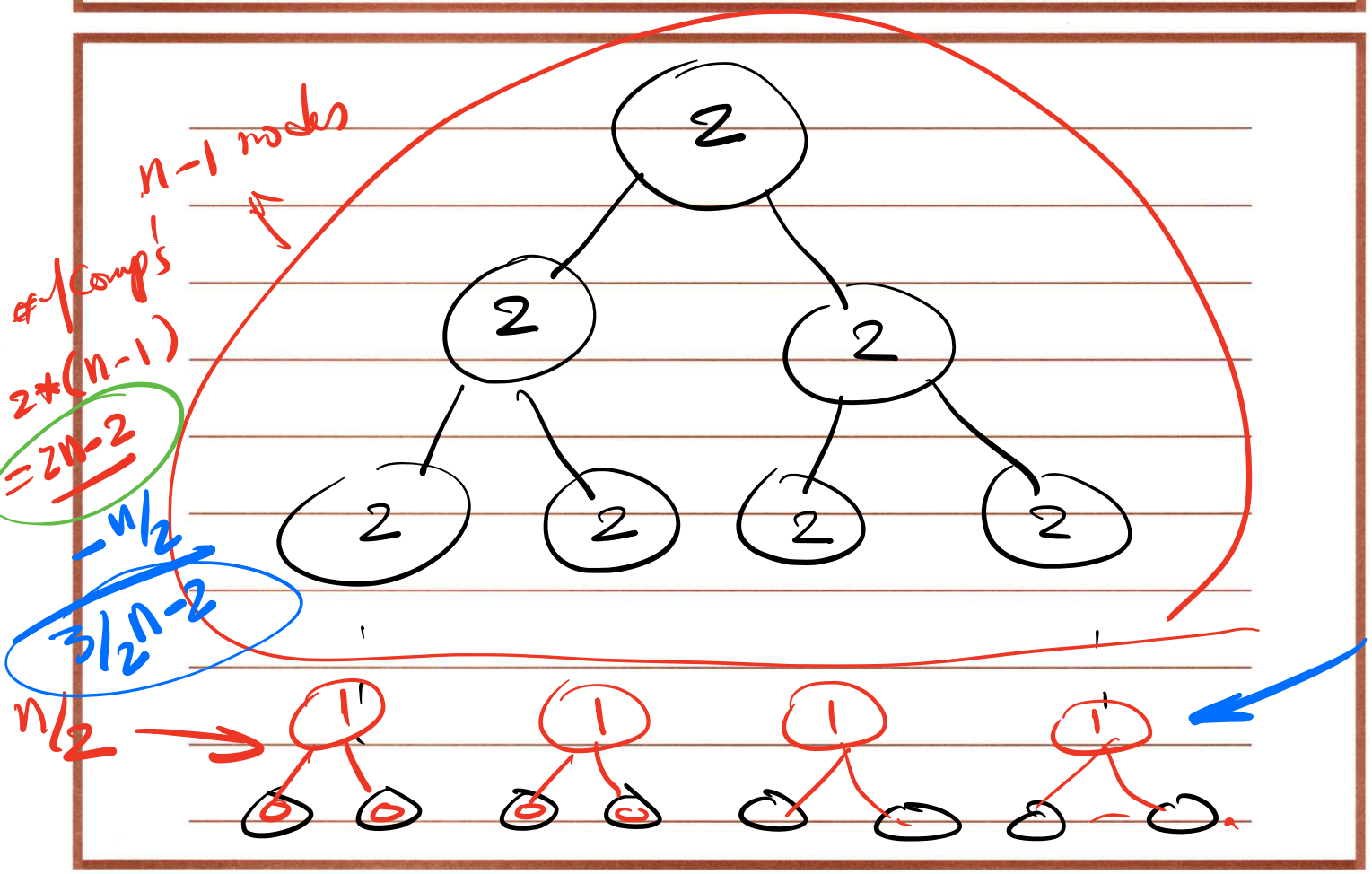
$f(n) = \Theta(n^2)$ $\quad f(n) = \Theta(n^2)$

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

Finding Min & Max in an unsorted array

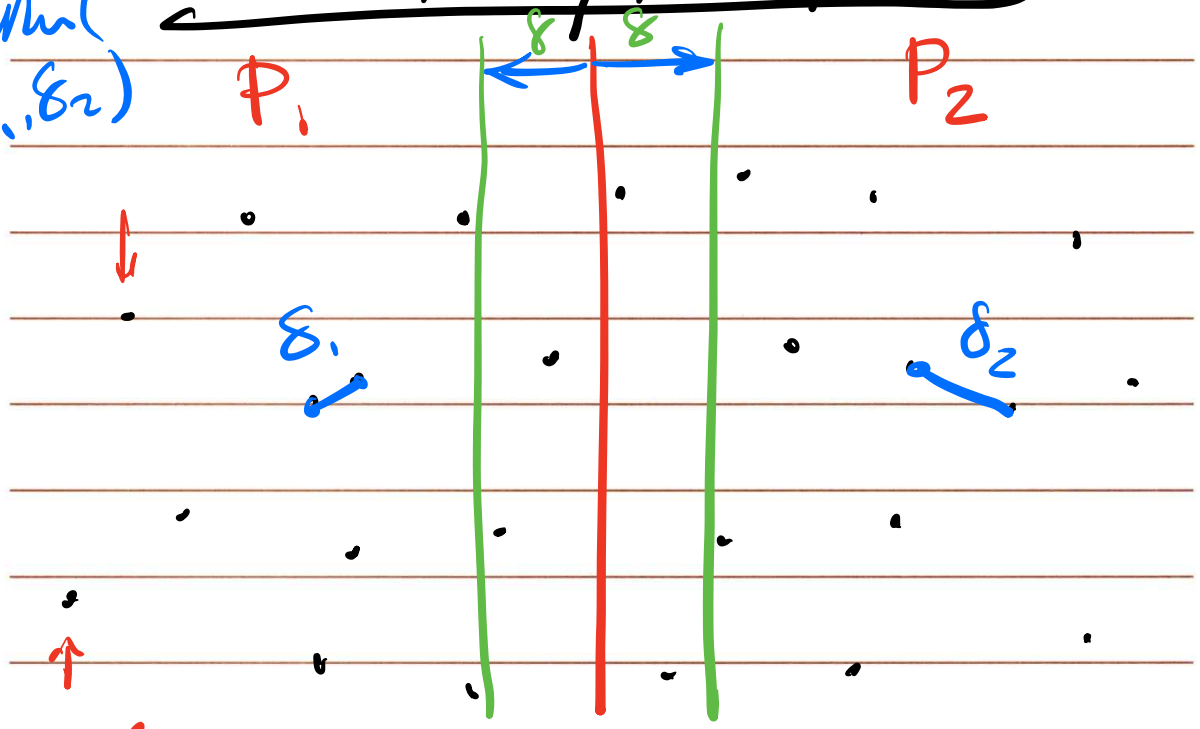


Brute force : $(n-1)+(n-1) = 2n-2 = \Theta(n)$



Closest pair of points problem (2D)

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

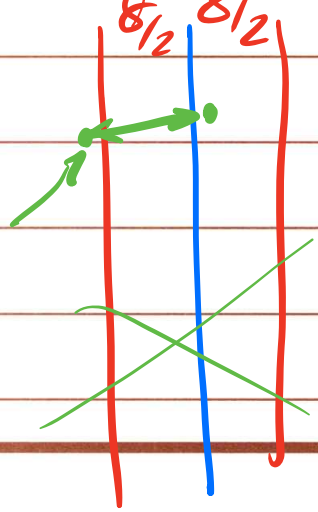


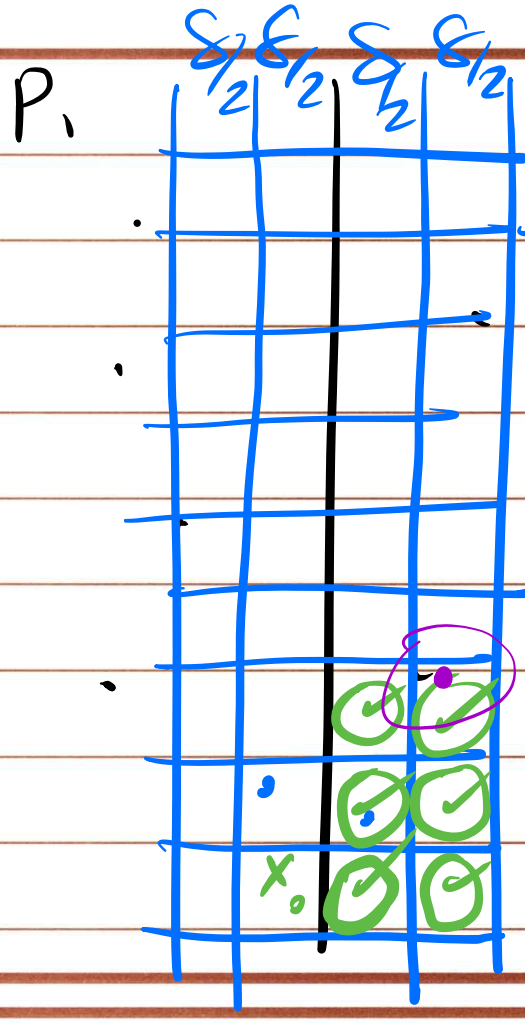
Brute force: $(n-1)+(n-2)+\cdots = \Theta(n^2)$

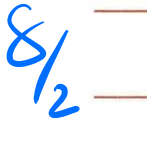
Case1: Both pts are in P_1 ✓

Case2: ✓

Case3: one pt. is in P_1 & the other in P_2

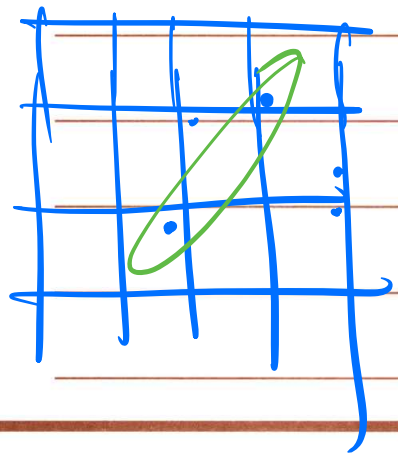






two pts for that from each other in this square are $\frac{\delta\sqrt{2}}{2}$ away from each other

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



Implementation

Driver

closest-pair (P)

Construct P_x : list of points sorted by x-coord. — $O(n \log n)$

" 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.

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

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

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

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

Construct S_y ... set of points in S sorted by y_coord.

$\theta(n)$ 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

$\theta(1)$ 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 cost: $\theta(n \log n)$

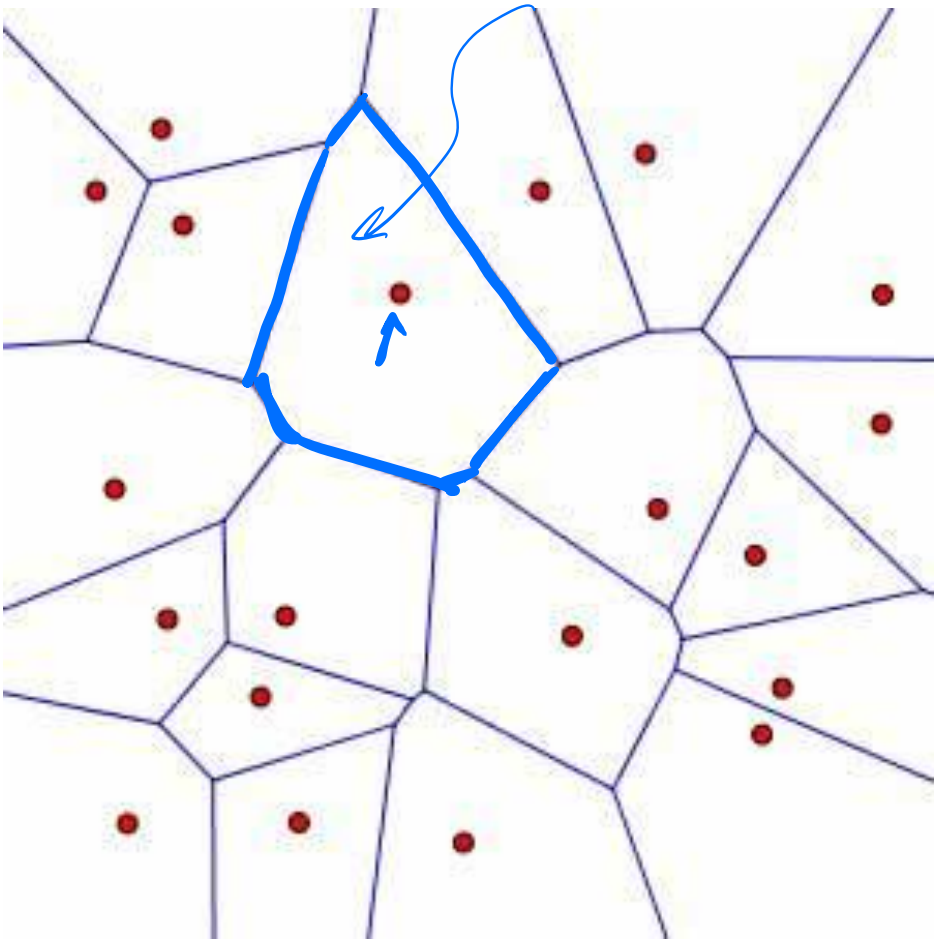
Driver: $\log_b a = 1$

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

$f(n) = \theta(n)$

Case #2 $\rightarrow T(n) = \theta(n \log n)$

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.

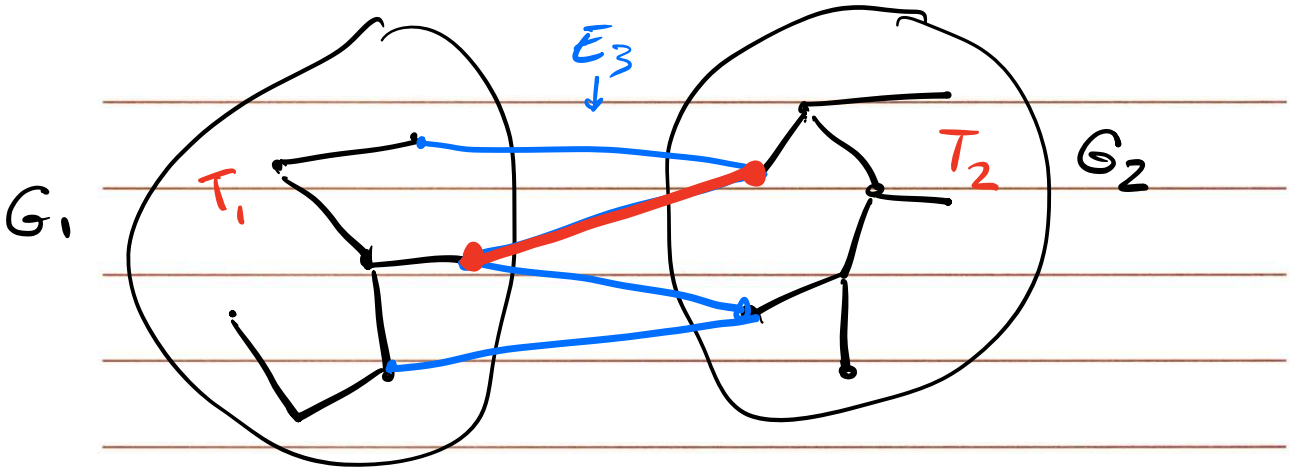
4. A tromino is a figure composed of three 1x1 squares in the shape of an L.
Given a $2^n x 2^n$ checkerboard with 1 missing square, tile it with trominoes.
Design a D&C algorithm and discuss its runtime complexity.

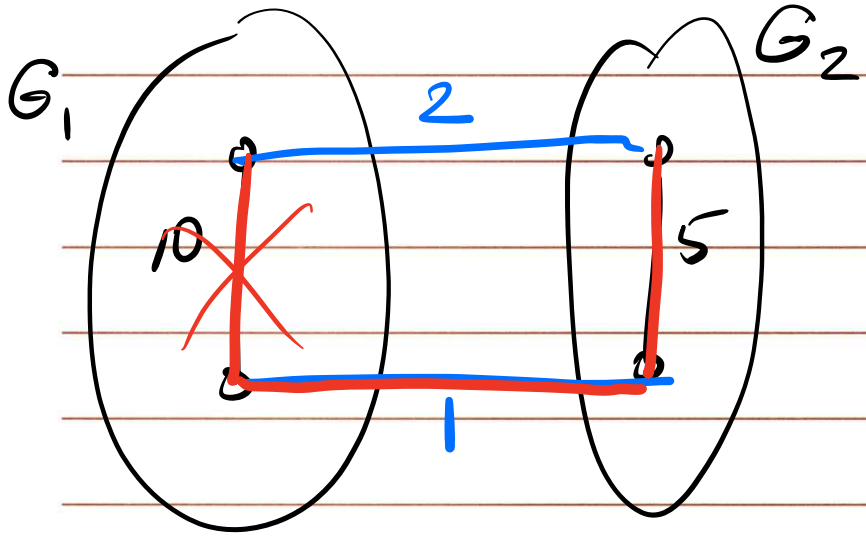
1. Suppose we have two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, along with T_1 which is a MST of G_1 and T_2 which is a MST of G_2. Now consider a new graph $G = (V, E)$ such that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup E_3$ where E_3 is a new set of edges that all cross the cut (V_1,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

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

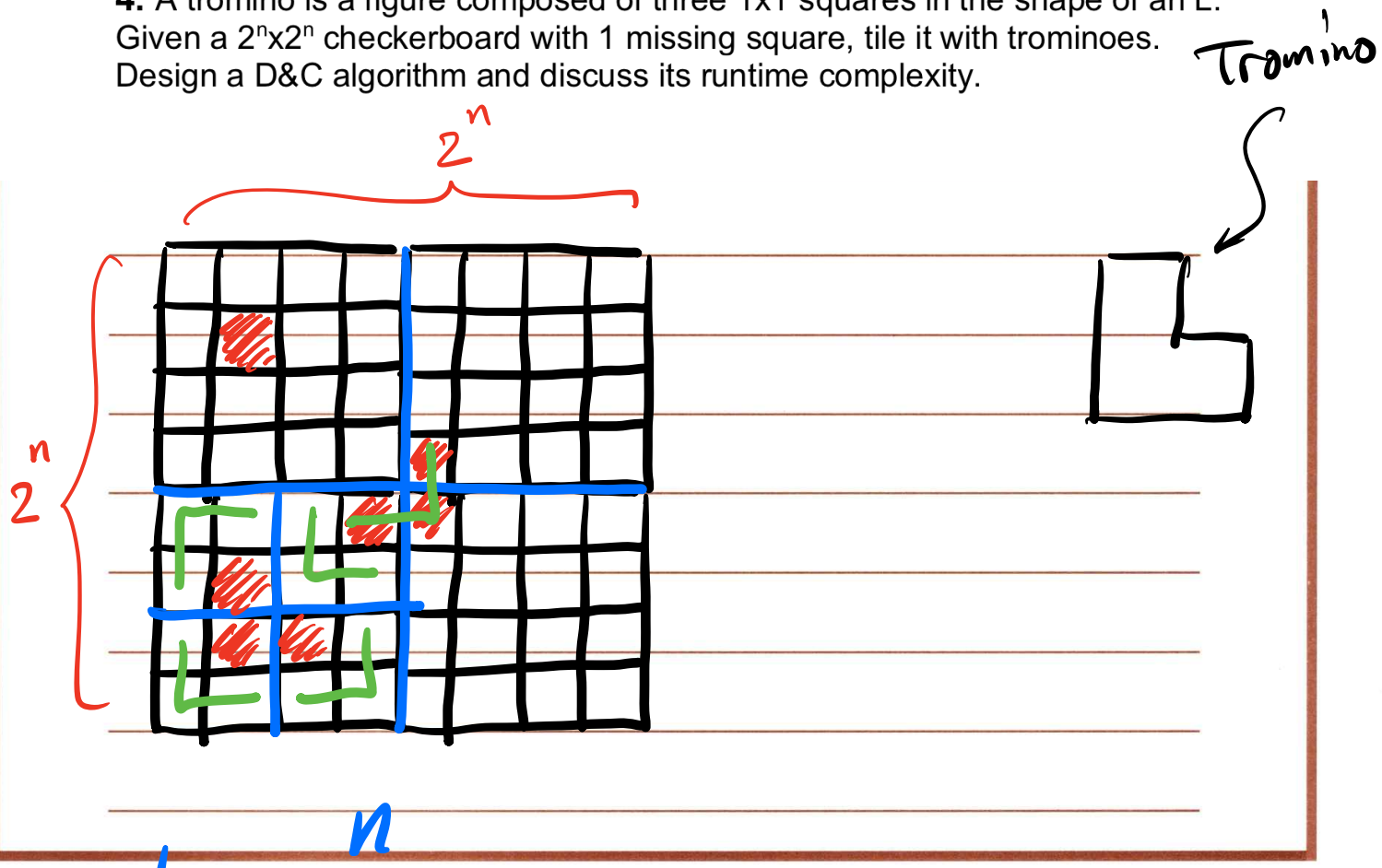
```
Maybe-MST(T1, T2, E3)
    emin = a minimum weight edge in E3
    T = T1 U T2 U { emin}
    return T
```

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



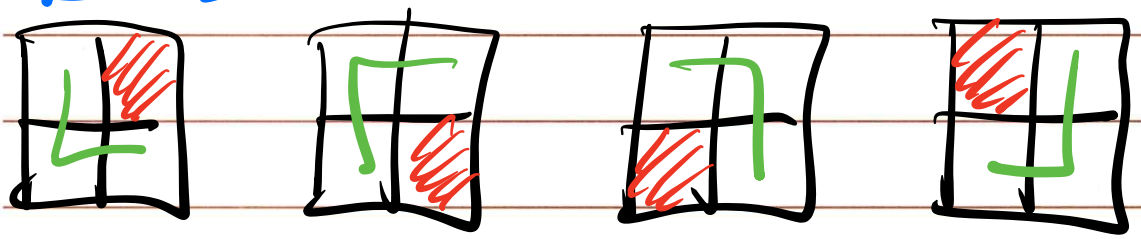


4. A tromino is a figure composed of three 1x1 squares in the shape of an L. Given a 2^nx2^n checkerboard with 1 missing square, tile it with trominoes. Design a D&C algorithm and discuss its runtime complexity.



n

$k = 2$



$a = 4$

$b = 2$

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

$\log_2 4 \quad 2$

$\Rightarrow k^{\quad} = k$

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

d. $D(n) = 4D(n/2) + n$

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.

$A = (2, 5, 7, 12, 15, 18, 21)$

$B = (4, 6, 7, 18, 22, 25, 32)$

$a = \geq 1$ $\quad \log_b^a$ $\quad \log_2^1$ $\quad > 0$

$b = 2$ $\quad n$ $\quad = n$ $\quad = n$

$f(n) = \theta(1)$ $\quad f(n) \geq \theta(1)$

~~Case #1~~ $\rightarrow T(n) = \theta(n)$

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

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

$n^2 \lg n$

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

$a = 4$

$b = 2$

$f(n) = \Theta(n^3)$

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

$f(n) = \Theta(n^3)$

$a f(n/b) \le c f(n) \qquad c < 1$

$4\left(\frac{n^3}{8}\right) \le c n^3$

$\frac{n^3}{2} \le c n^3 \qquad c < 1$

$.5 \le c < 1$

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