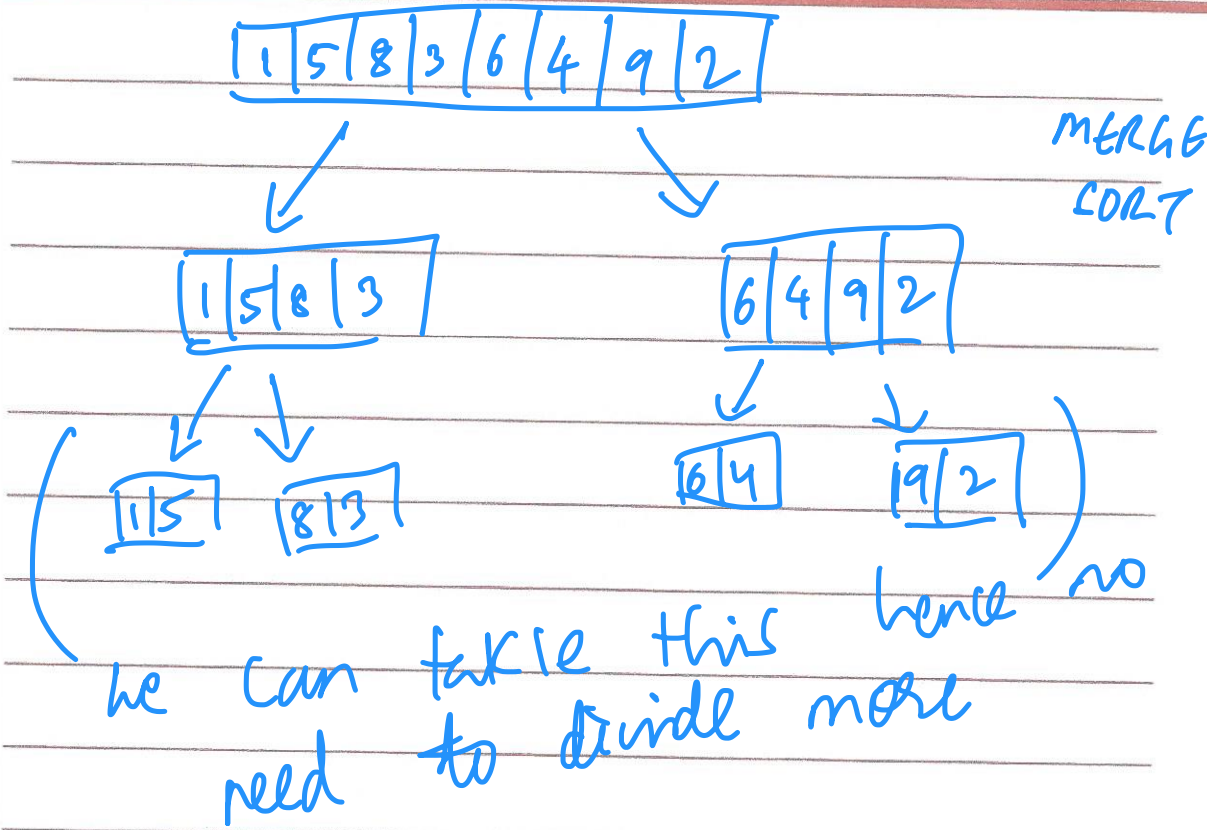
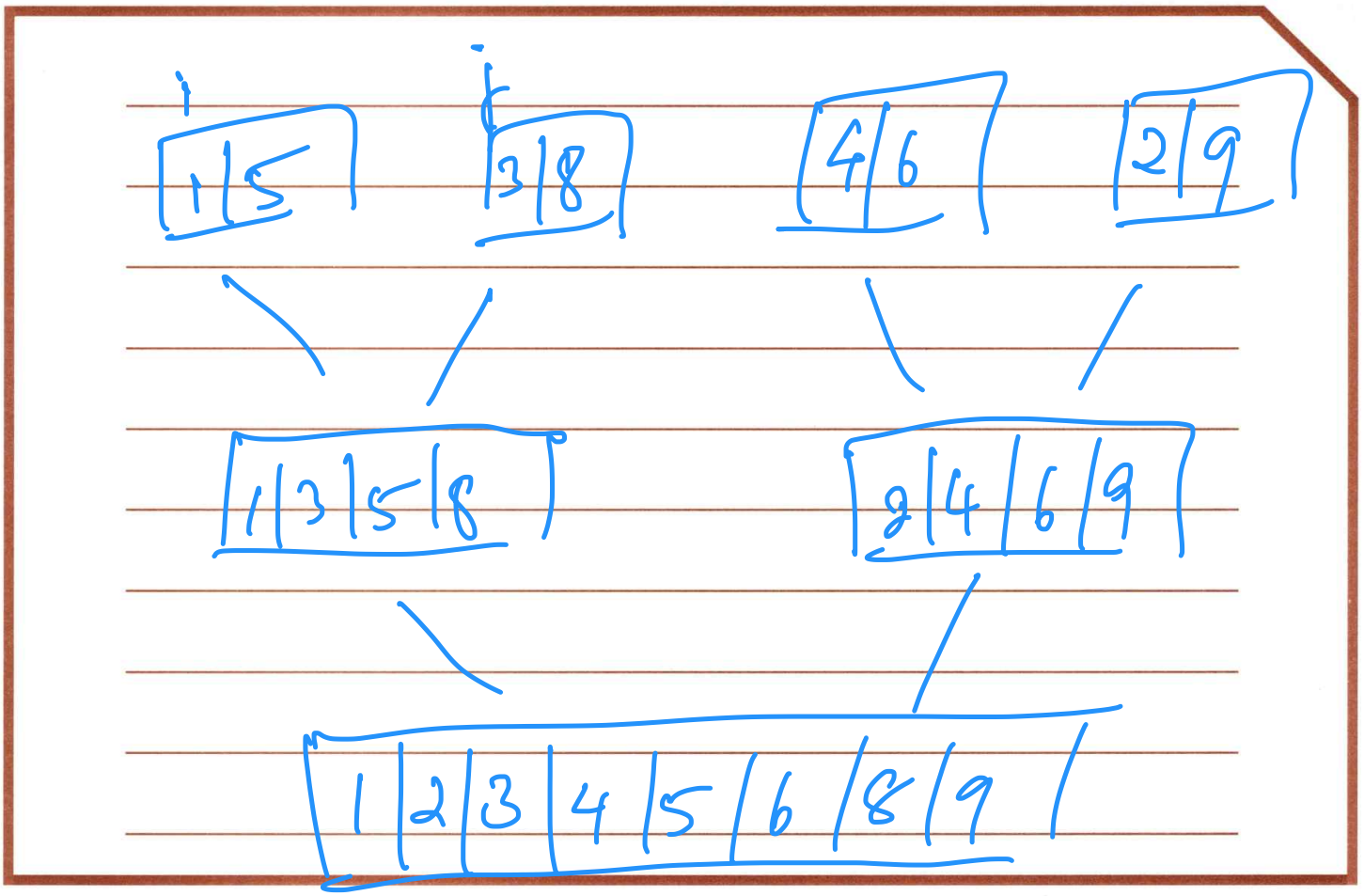
Divide & Conquer

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 5
3 8
4 6
2 9
1 3 5 8
2 4 6 9
1 2 3 4 5 6 8 9

Start index

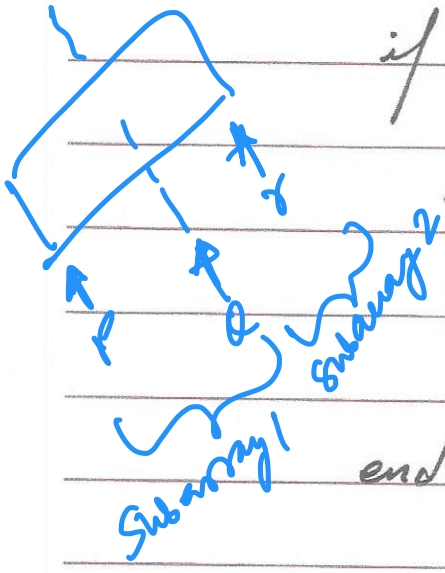
```
MERGE-SORT(A, p, r)
    if p < r then
        q = ⌊(p+r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q+1, r)
        MERGE(A, p, q, r)
    end if
```

end index

Divide

Conquer

Combine



Analyzing Merge-sort

Divide – Takes $\underline{\Theta(1)}$

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

Combine – Takes $\underline{\Theta(n)}$ on array size $= n$

Recurrence equation for merge sort

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

$2T(n/2)$ ← Conquer

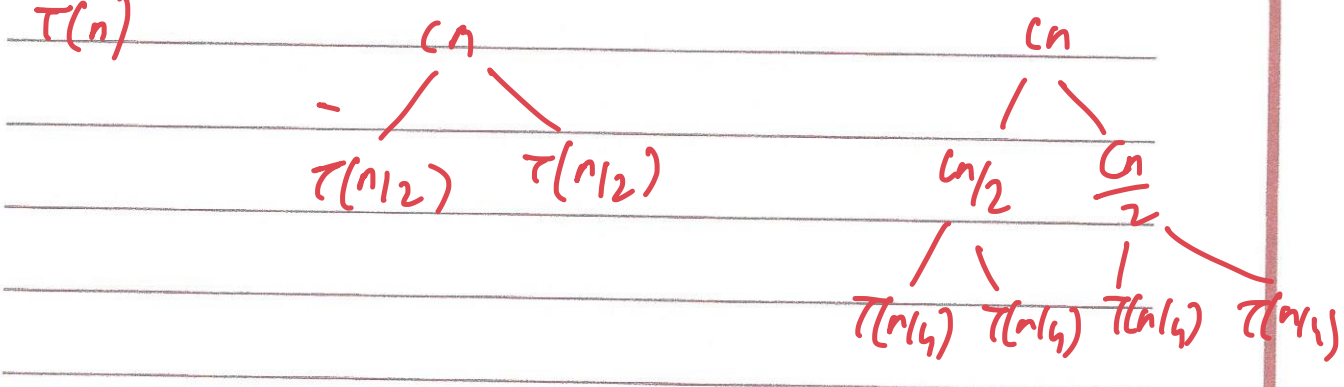
$\Theta(n)$ ← MERGE (Combine)

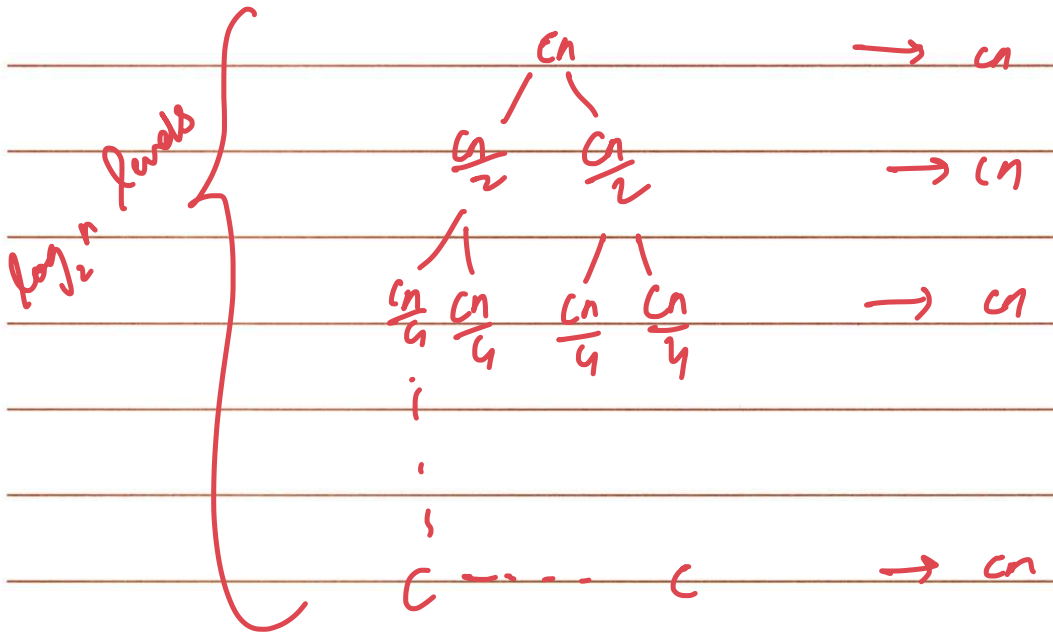
$\Theta(1)$ ← Divide

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: no. of subproblems
- n/b: size of subproblem
- $D(n)$: time to divide
- $C(n)$: time to combine





Total Time = $cn * \log_2 n = \Theta(n \log n)$

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

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

$f(n) = n^{1.0001}$ #case3

$n^{\log_b a} = n$

CASE 2 GENERALIZATION

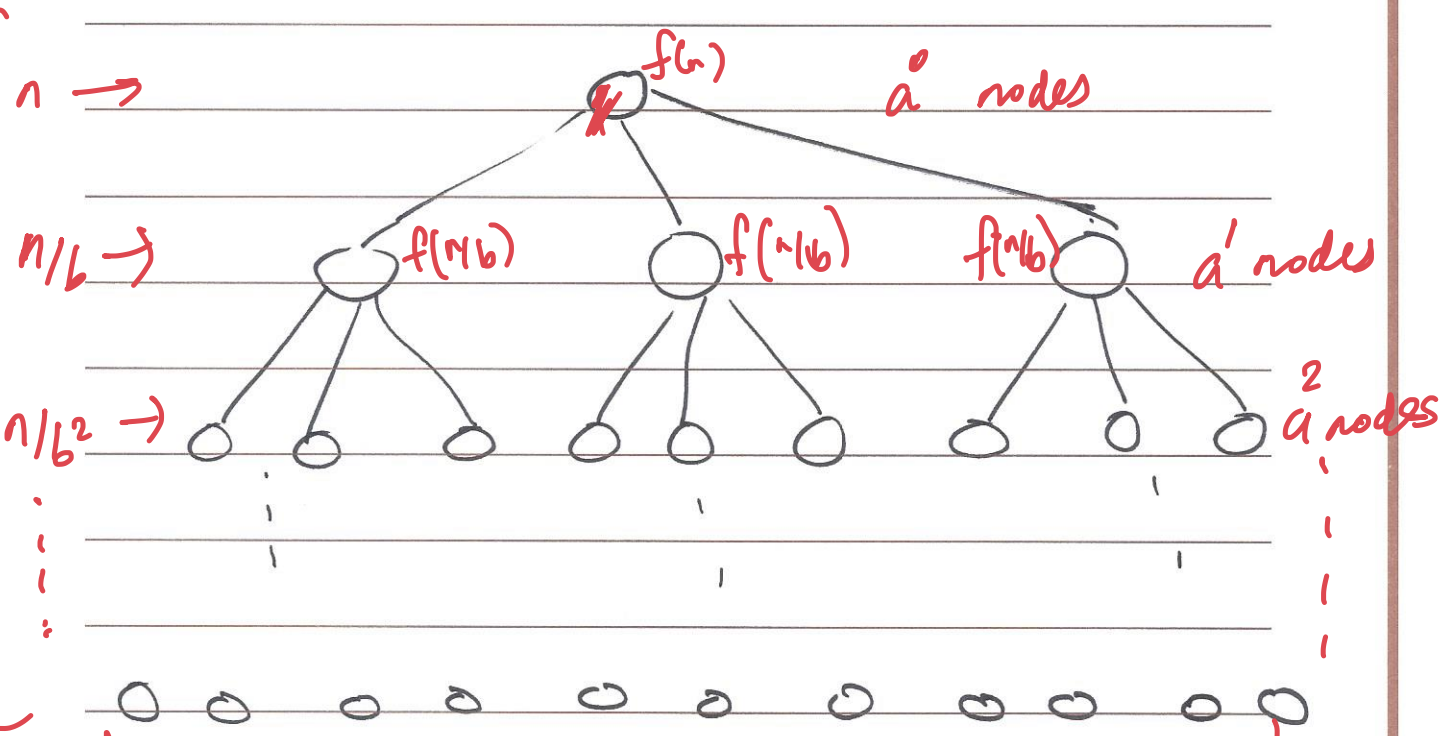
in general if $f(n) = \Theta(n^{\log_b a} \lg^k n)$

where $k \geq 0$

Then

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

Intuition Behind The Master Method



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

Total nodes

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

$$n^{\log_b a} \left(1 + \frac{1}{a} + \frac{1}{a^2} + \dots + \right)$$

$$n^{\log_b a} (\text{constant})$$

CASE 1 $\Theta(n^{\log_b a})$

Bottom Heavy

$f(n) + af(n) + a^2 f(n) + \dots$
$= Cf(n)$
$= \Theta(f(n))$

CASE 3

Top Heavy

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

$f(n) =$ divide + combine

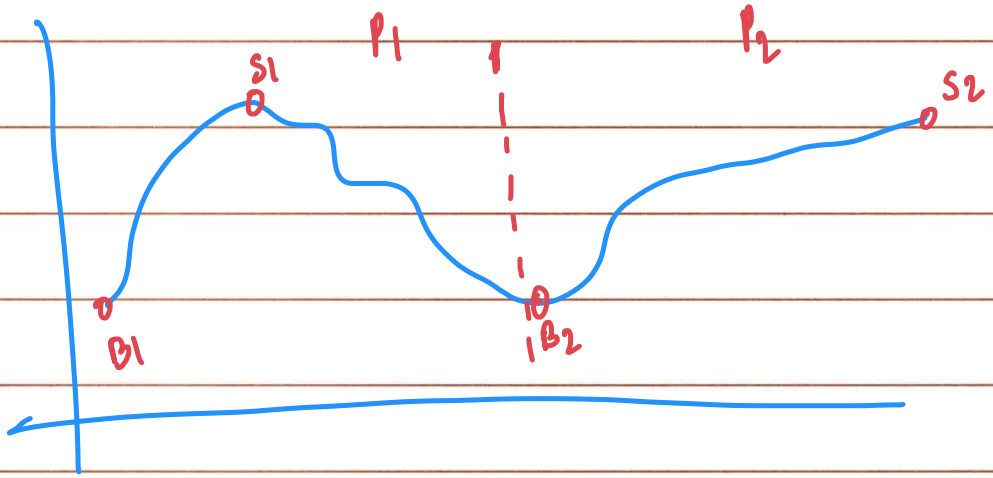
$D(n) + C(n)$

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

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

Case 2 $\longrightarrow$ $T(n) = \Theta(n \log n)$

Stock Market Problem



Brute Force

Buy on day 1 sell on all (n-1) days

" day sell on all (n-2) days

.

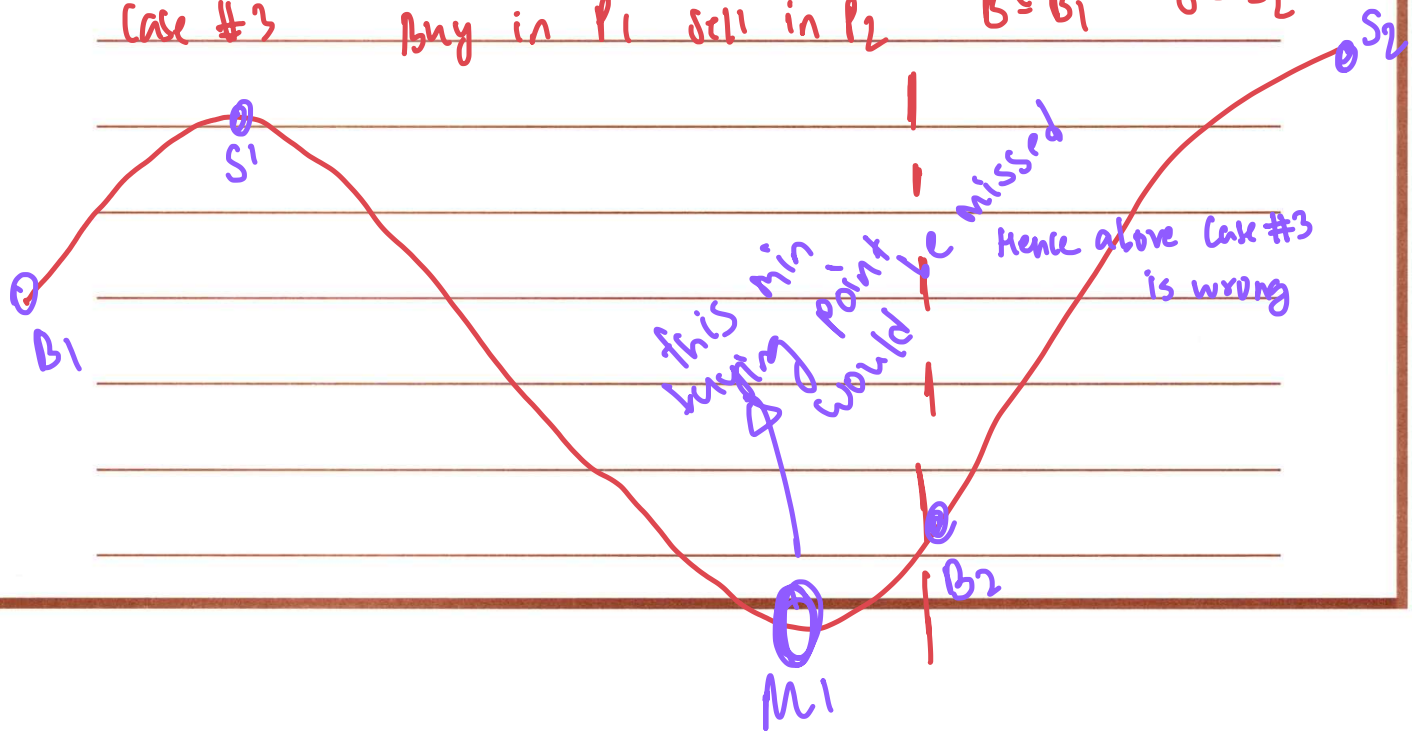
.

$O(n^2)$

Case #1 Buy and sell in P_1 $B = B_1$ $S = S_1$

Case #2 Buy and sell in P_2 $B = B_2$ $S = S_2$

Case #3 Buy in P_1 sell in P_2 $B = B_1$ $S = S_2$



Case #3 modified

$B = M_1$

$S = X_2 = S_2$

$M = \min(M_1, M_2)$

$X = \max(X_1, X_2)$

Complexity Analysis

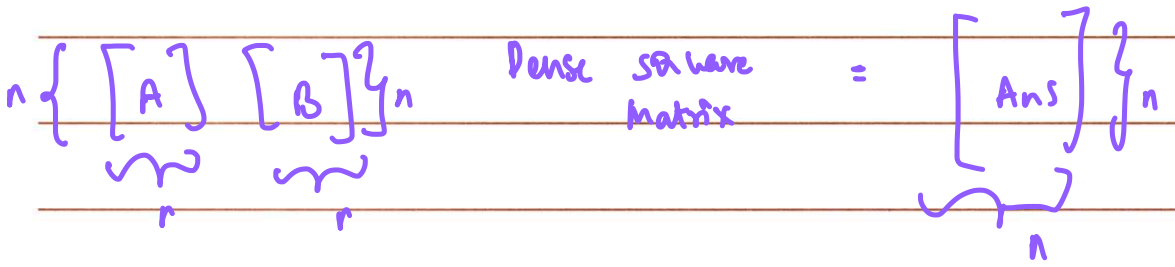
$f(n) = O(n) + C(n) = \Theta(1)$

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

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

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

Dense Matrix Multiplication



Brute Force $\Rightarrow \theta(n^3)$

$$\begin{bmatrix} A_1 \\ \hline A_2 \end{bmatrix} [B] = \begin{bmatrix} C_1 \\ \hline C_2 \end{bmatrix}$$ WRONG

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

$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$ → combine steps

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

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

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

Here IF we can reduce # and increase # if we can reduce the complexity

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

$\theta(1) + \theta(n^2)$ ⟹ CASE 3

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

a: 8 subproblems

$T(n) = \theta(n^3)$

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

$$f(n) = \theta(1) + \theta(n^2) = \theta(n^2)$$

$n^{\log_b a} = n^{\log_2 7} = n^{2.81}$ → case #1

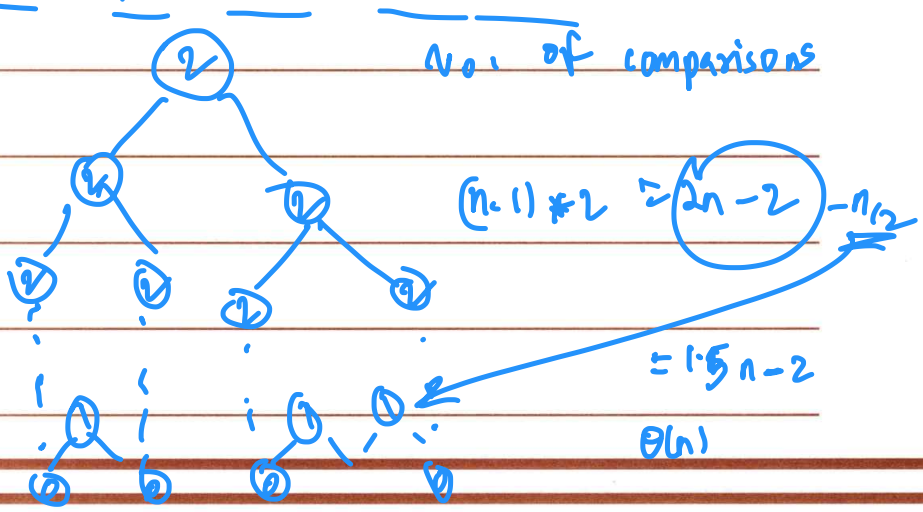
$$T(n) = \theta(n^{2.81})$$

Finding Min & Max in an unsorted array

Brute Force : $(n-1) + (n-1) = 2(n-1) = 2n-2 = \theta(n)$

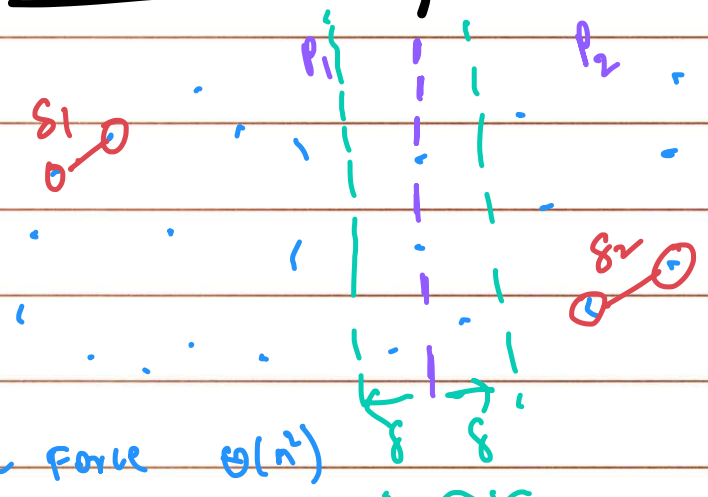
No. of comparisons

Divide & Conquer



Same Time complexity but no. of operations is less

Closest pair of points problem (2D)



Find closest pair of points

Brute force $\Theta(n^2)$

Combining the soln's

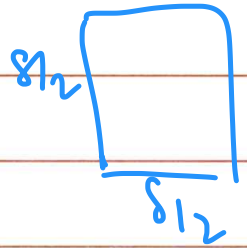
Case #1 Both points in P_1

For case 3 only try for points inside this band

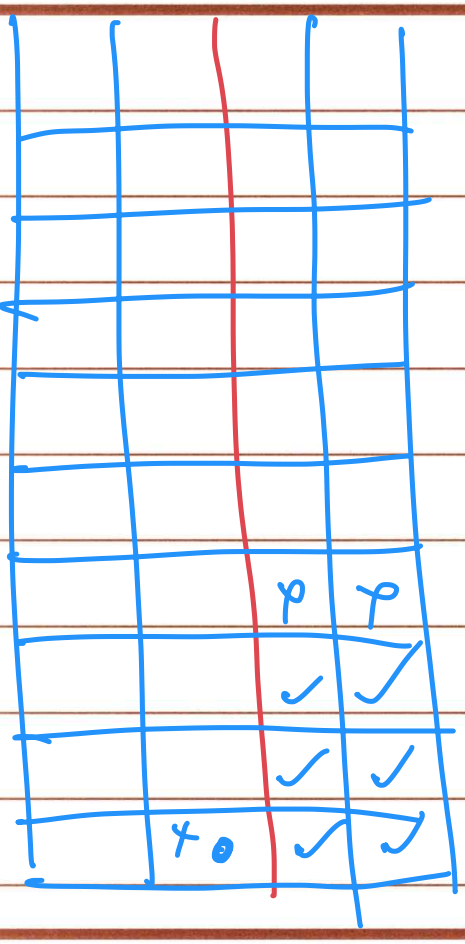
Case #2 Both points in P_2

Case #3 One point in P_1 and other in P_2

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



So no two points can be inside this square as min dist is δ for case1 and case 2



∴ we only look at constant points rather than all points

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 | base case
solve it directly | trivial
else
$O(1)$ 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

left subproblem

$(q_0, q_1) = \text{closest\_pair\_Rec}(Q_x, Q_y)$

right subproblem

$(r_0, r_1) = \text{closest\_pair\_Rec}(R_x, R_y)$

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

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.

$O(11 \cdot n)$

Let (s, s') be pair with min. distance

$O(n)$

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

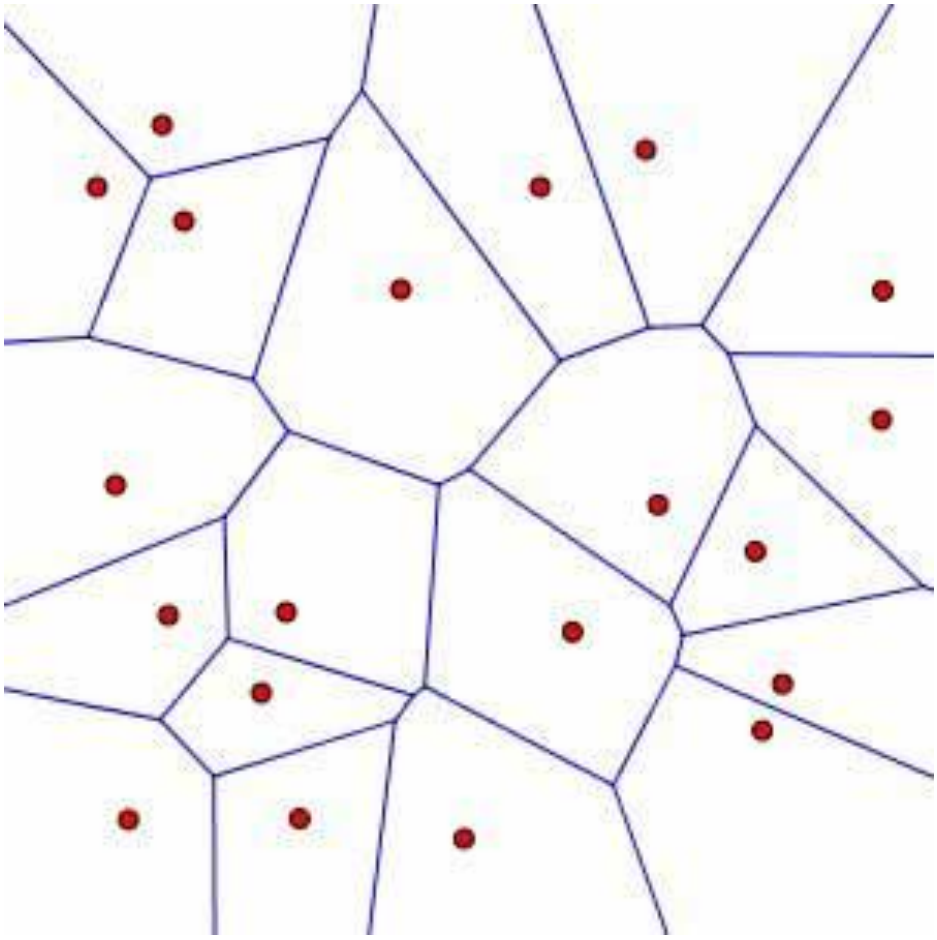
$f(n) = \Theta(n)$

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

CASE #2 $T(n) = \Theta(n \log n)$

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
```

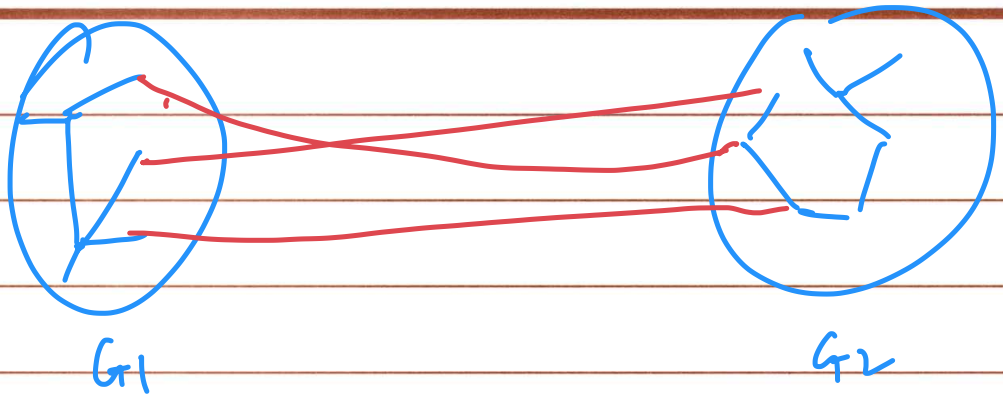
Does this algorithm correctly find a MST of G? Either prove it does or prove it does not.

2. Solve the following recurrences using the Master Method:

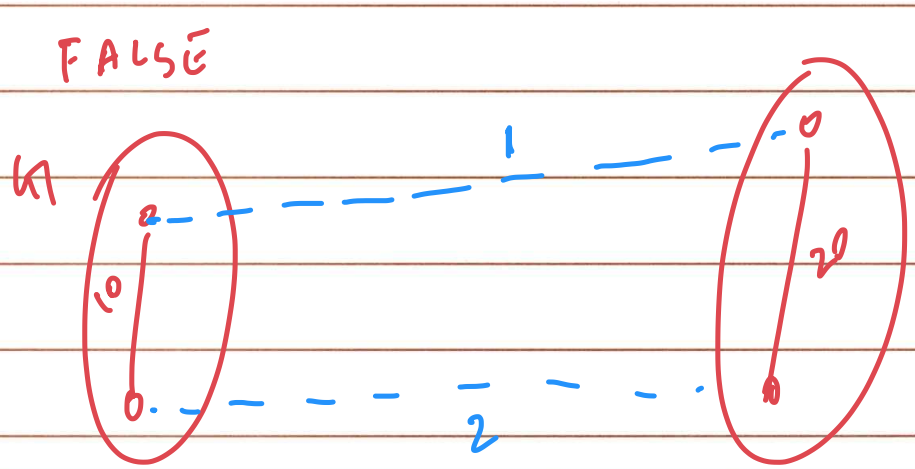
a. $A(n) = 3\,A(n/3) + 15$
b. $B(n) = 4\,B(n/2) + n^3$
c. $C(n) = 4\,C(n/2) + n^2$
d. $D(n) = 4\,D(n/2) + n$

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 

FALSE



Our soln will pick $\{0, 20, \min(1,2)\}$

$\{0, 20, 1\}$

But best ans is $\{0, 1, 2\}$

---

② a. $A(n) = 3A(n/3) + 15$

$f(n) = 15 = O(1)$

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

CASE 1

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

② b. $B(n) = 4B(n/2) + n^3$

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

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

$af(n/b) \le c\, f(n)$ For $c < 1$

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

$0.5 \le c$ For $c < 1$

True

Hence it is CASE 3

#CASE 3 $B(n) = \Theta(n^3)$

② c. $C(n) = 4C(n/2) + n^2$

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

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

CASE #2 $C(n) = \Theta(n^2 \log n)$

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

$f(x) = n^2 \log^2 n$

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

Case 2 $C(n) = \Theta(n^2 \log^3 n)$

(general)

② d. $D(n) = 4D(n/2) + n$

$f(n) = n$

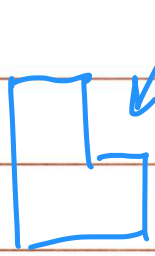
$n^{\log_b a} = n^{\log_2 4} = n^2$) CASE 3?

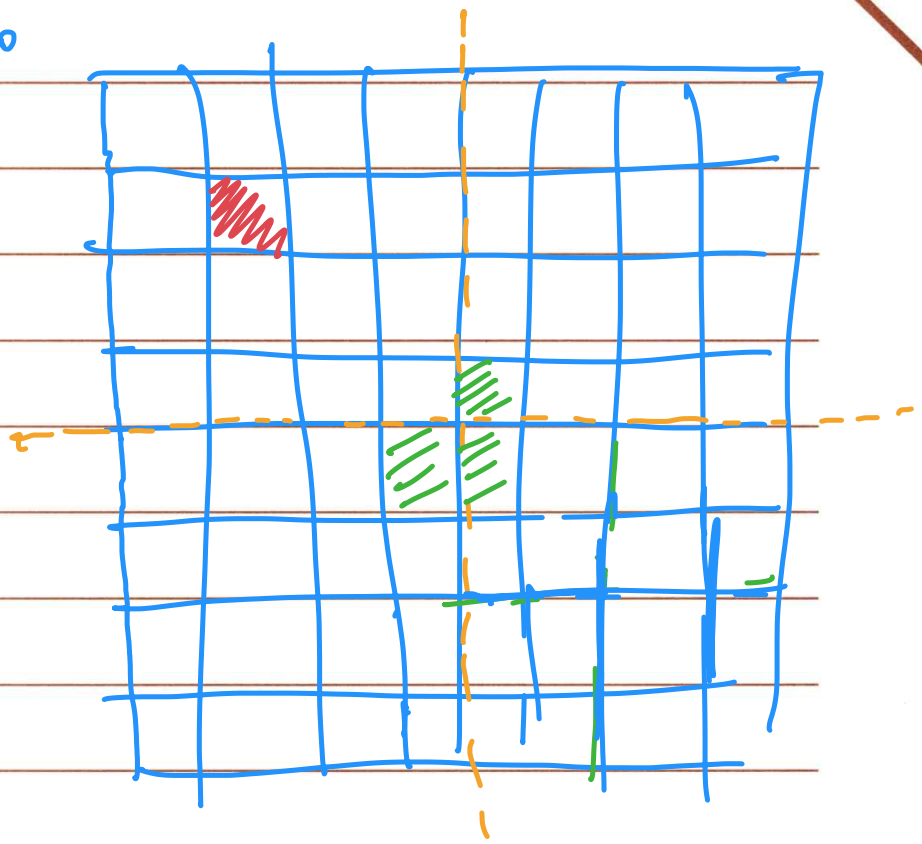
$af(n/b) \le c\, f(n)$ $c < 1$

$4 \cdot n/2 \le cn$

$2 \le c$ False Hence cannot be solved

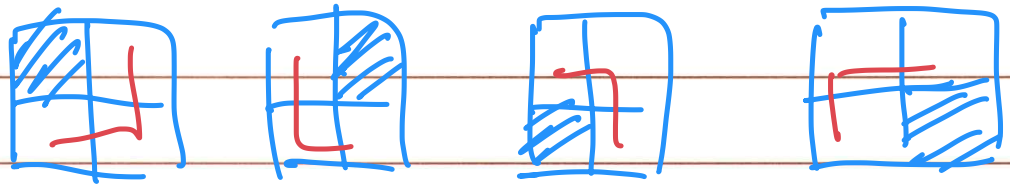
④





→ our created holes to make all 4 subproblems same i.e containing 1 hole

Trivial cases



0g size $(2^n)^k * (2^n)^k$ K*K

$2^{n-1} * 2^{n-1}$

$2^{n-2} * 2^{n-2}$

2x2

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

$= \Theta(1) + \Theta(1) = \Theta(1)$

$K^{\log_b a} = K^{\log_2 4} = K^2$] CASE #1

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

(3)

$A = (5, 7, 11, 15, 20, 21, 30)$ →

$B = (8, 16, 19, 28, 30, 40, 45)$ ←

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

$n^{\log_b a} = n^{\log_2 1} = n^0 = \Theta(1)$] CASE #2

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