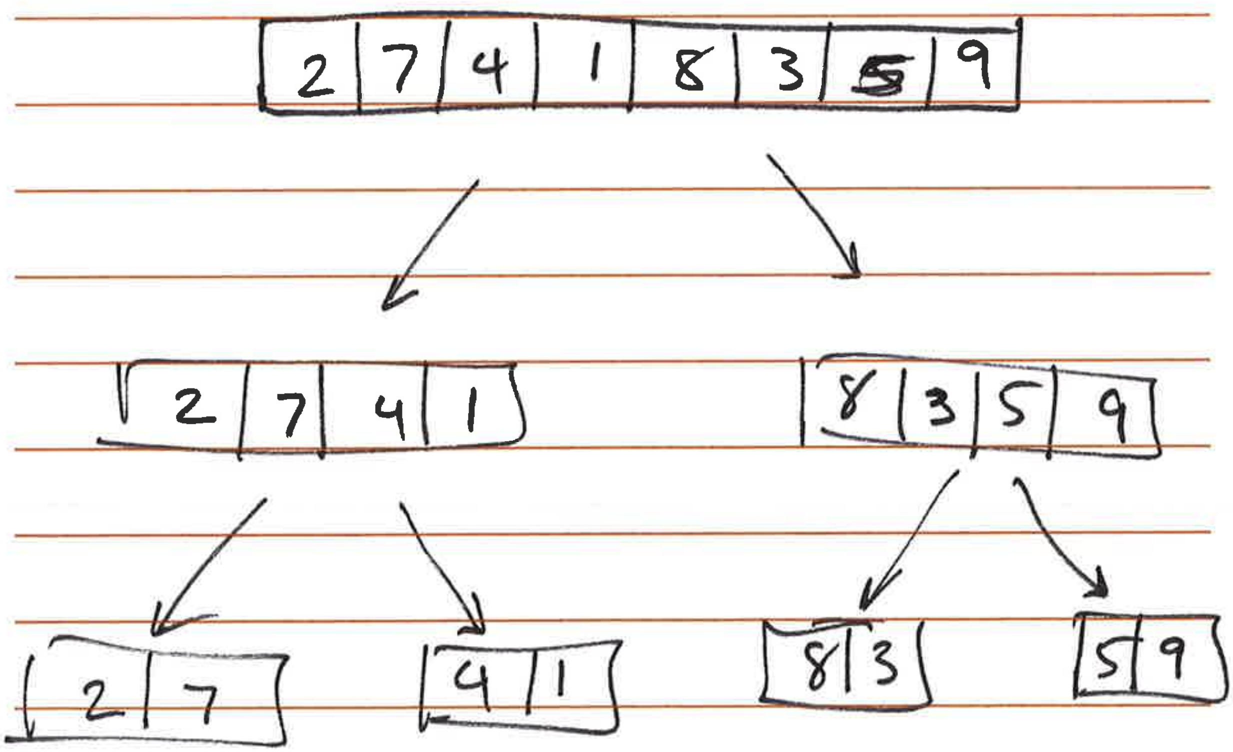
Divide & Conquer

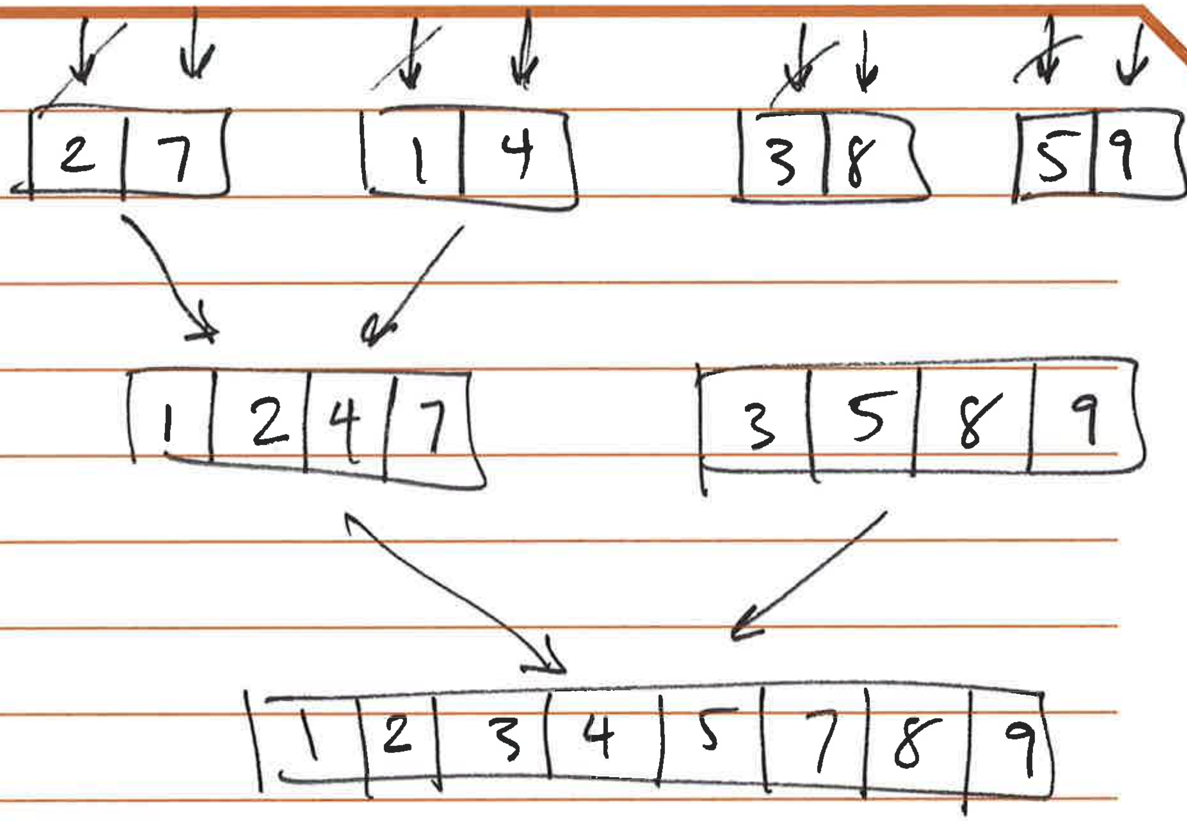
Divide & Conquer

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





MERGE-SORT (A, p, r)

if $p < r$ then

$q = \lfloor (p+r)/2 \rfloor$ ← Divide

Conquer $\left\{ \begin{array}{l} \text{MERGE-SORT}(A, p, q) \\ \text{MERGE-SORT}(A, q+1, r) \end{array} \right.$

MERGE (A, p, q, r) ← Combine

endif

Analyzing Merge-Sort

Divide - Takes $O(1)$

Conquer - if the original problem takes $T(n)$ time, the 2 subproblems take $2 \cdot T(n/2)$

Combine - Takes $O(n)$ on array size is n.

recurrence equation for merge-sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ \underbrace{2 \cdot T(n/2)}_{\text{Conquer}} + \underbrace{\Theta(n)}_{\text{Combine}} + \underbrace{\Theta(1)}_{\text{Divide}} & \end{cases}$$

In general,

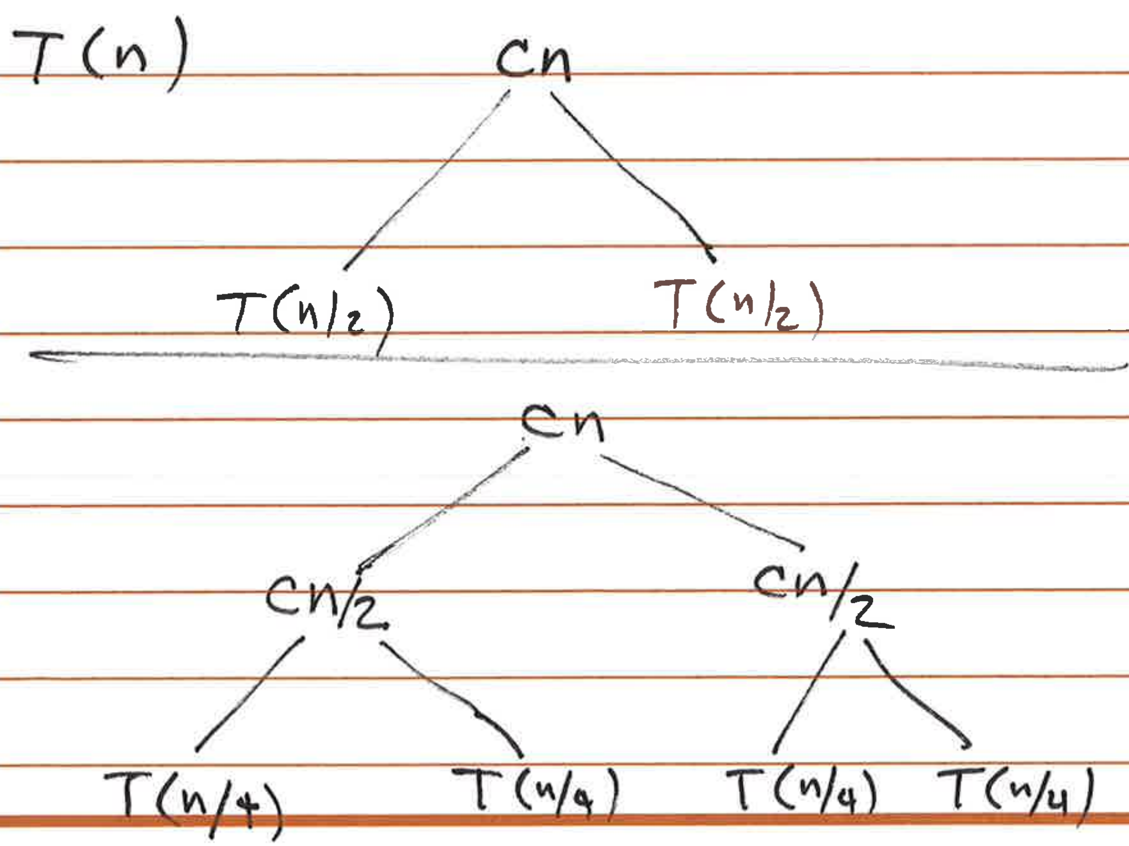
$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ a\,T(n/b) + D(n) + C(n) \end{cases}$$

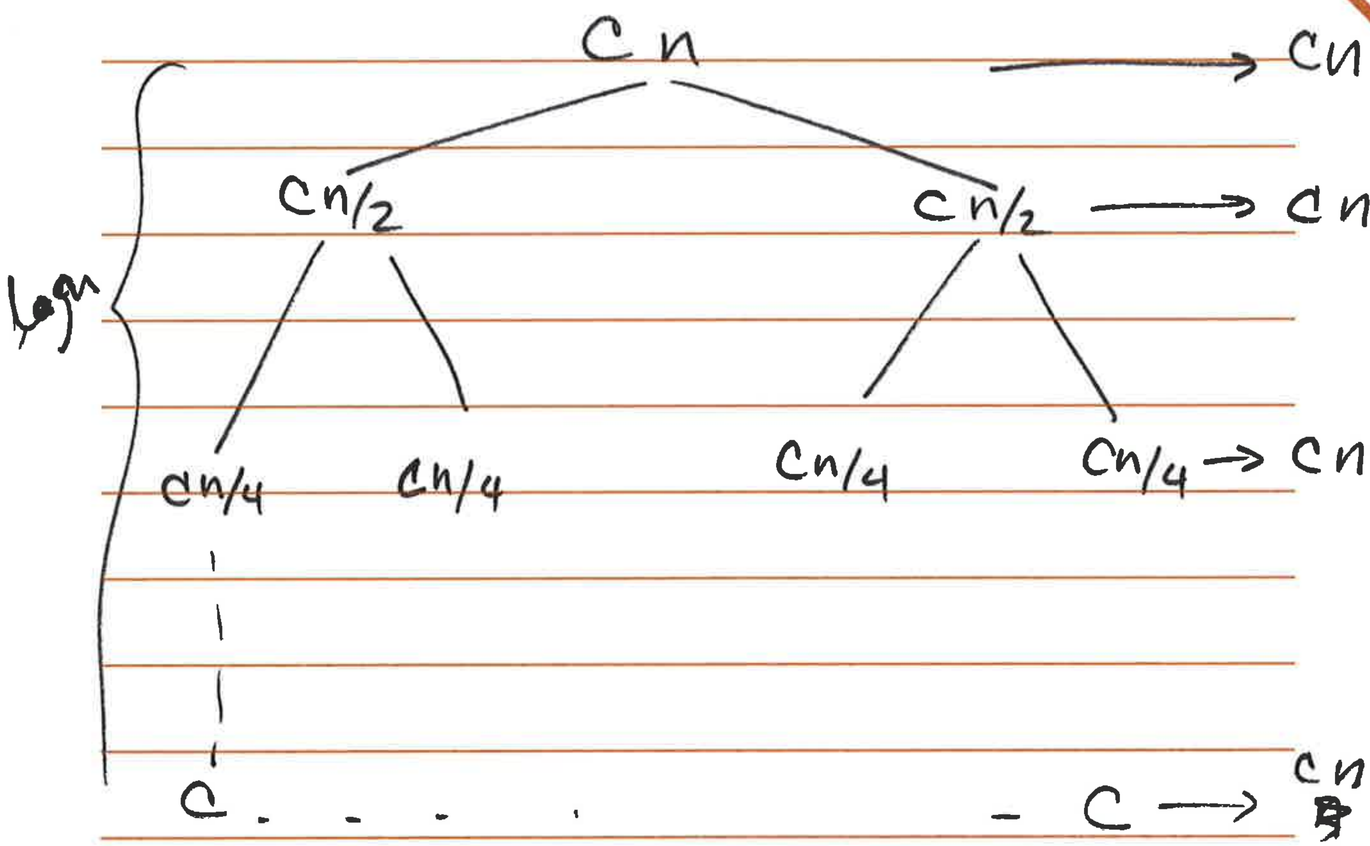
# of subproblems

Size of the subproblem

Time to Divide

time to combine





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

Master Method

It is a cookbook method for solving recurrences of the form:

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

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

Master Theorem

Given the above def. of the recurrence relation, $T(n)$ can be bounded ~~as~~ 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 $f(n) = \Omega(n^{\log_b^a + \epsilon})$ for some constant $\epsilon > 0$ and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficcently large n, then $T(n) = \Theta(f(n))$

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

$$\boxed{\begin{aligned} f(n) &= n \lg n \\ n^{\log_b^a} &= n \end{aligned}} \rightarrow n \lg n$$

$$\Longrightarrow O(n \lg^2 n)$$

Complexity of merge sort using the Master Theorem:

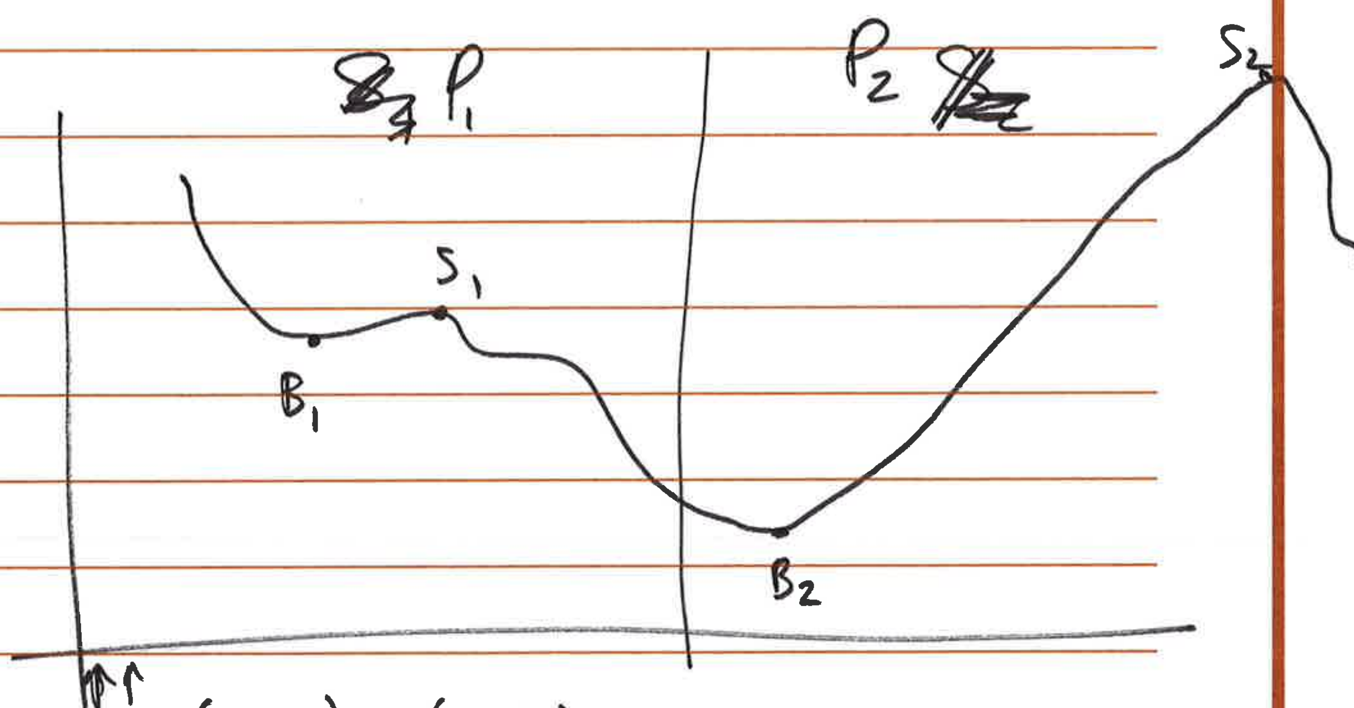
$a = 2$

$b = 2$

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

$$f(n) = cn + O(1) = cn$$

$$\Rightarrow \Theta(n \lg n)$$



$$\underbrace{(n-1) + (n-2) + \cdots + 1}_{O(n^2)}$$

Case 1 buy & sell in P_1

$S = S_1$ $\quad$ $M = \text{Min}(M_1, M_2)$

$B = B_1$ $\quad$ $X = \text{Max}(X_1, X_2)$

Case 2 buy & sell in P_2

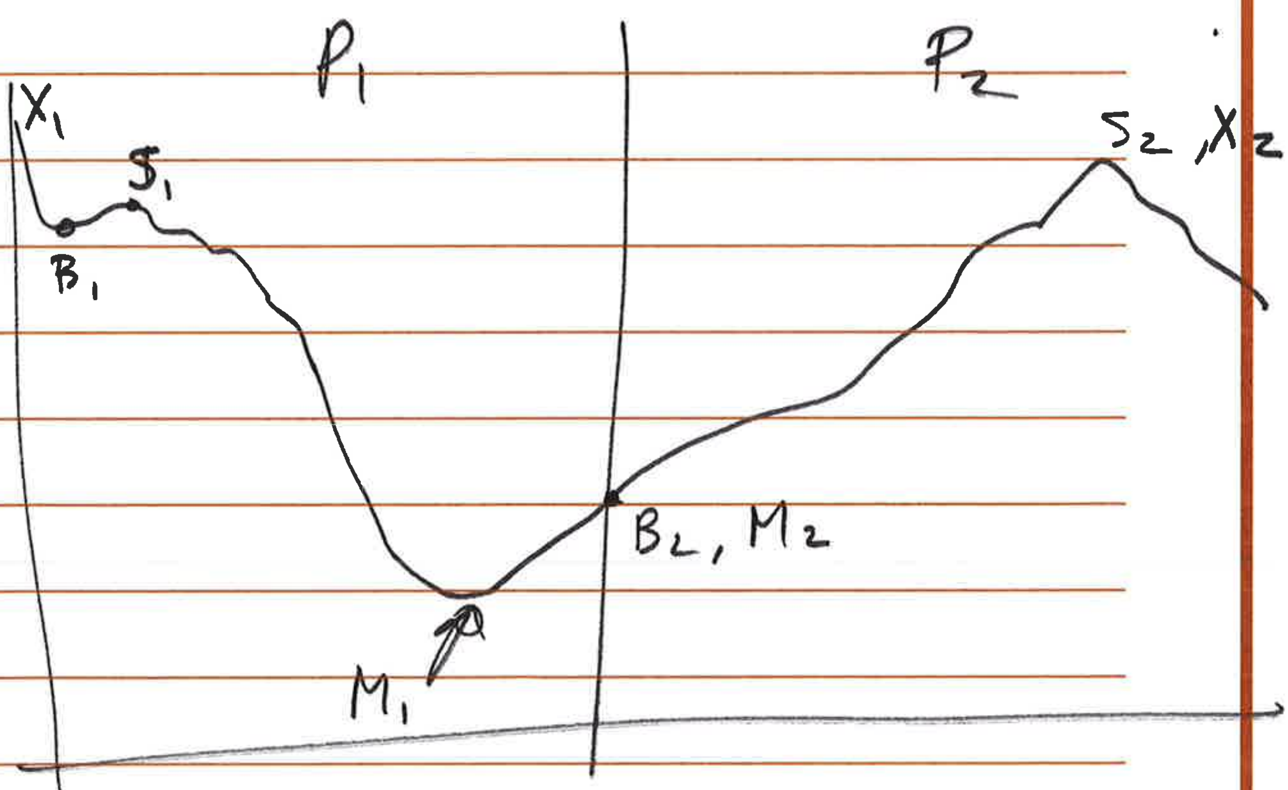
$S = S_2$ $\quad$ $M = \text{Min}(M_1, M_2)$

$B = B_2$ $\quad$ $X = \text{Max}(X_1, X_2)$

Case 3 buy in P_1 & sell in P_2

$S = \cancel{S_2}\, X_2$ $\quad$ $M = \text{Min}(M_1, M_2)$

$B = \cancel{B_1}\, M_1$ $\quad$ $X = \text{Max}(X_1, X_2)$



Complexity

$a = 2$

$b = 2$

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

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

$$O(1) + O(1) = O(1)$$

$\Rightarrow$ $\cancel{\Theta}$ complexity $= \Theta(n)$