L31
Quick Sort and more

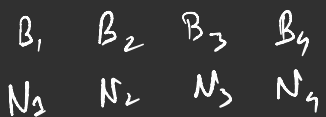
If interested, check out the System Design course.
Early Bird discount ends TONIGHT.

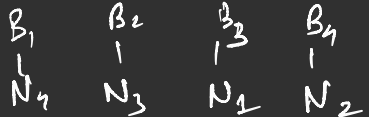
Join Discord - https://bit.ly/ly-discord

RECAP

Let’s play a game

Nuts & Bolts Problem





Basically, K nuts are given & K bolts are given:

1. Each nut has a unique perfect bolt.
2. Our Target is to find the matches for all.

But, there are a few constraints as well:

1. We can't compare 2 nuts directly.
2. Nor can we compare 2 bolts directly.

Then, what can we do? xD
We can compare a nut and a bolt in 1 operation.

$N_i \longleftrightarrow B_j$

Possible Outcomes :

1) $N_i < B_j$

2) $N_i == B_j$

3) $N_i > B_j$

Brute Force

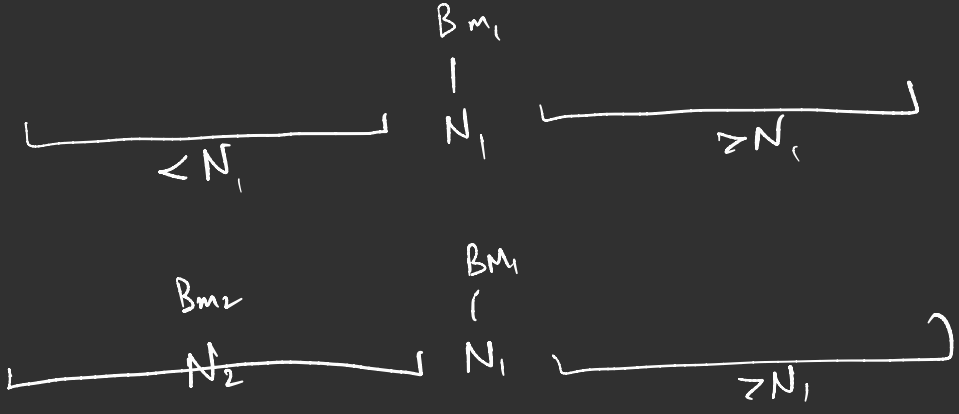
$N_1 \Rightarrow B_{m_2}$ (K)

$N_2 \Rightarrow B_{m_2}$ $(K-1)$

$(K-2)$

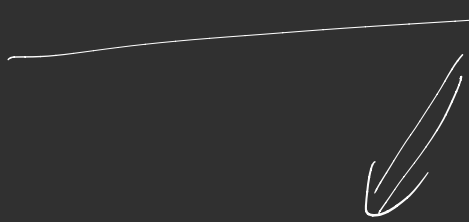
operations

$O(K^2)$



N_3

$1 + 2 + 3 + 4 \ldots \longrightarrow -(K-1)$

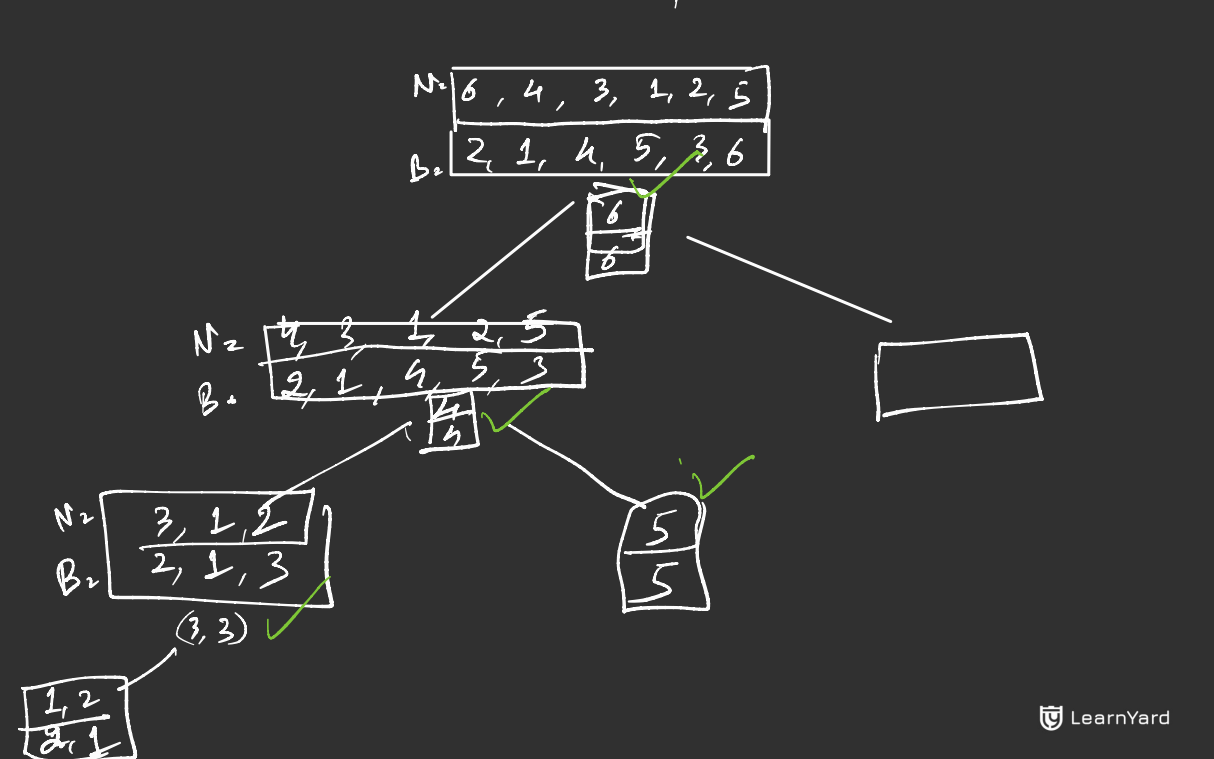


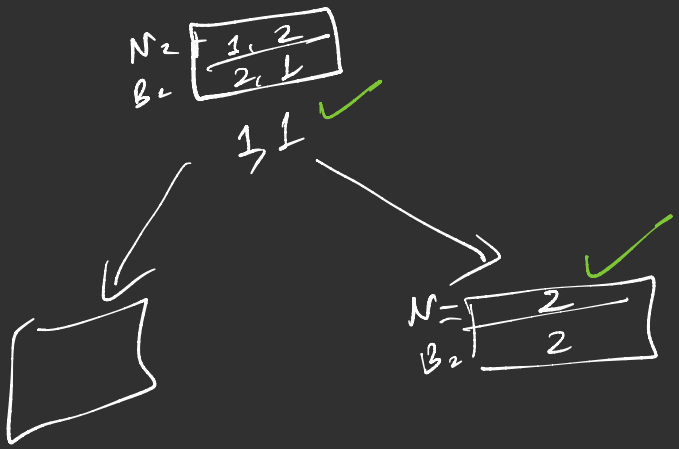
$O(K^2)$

K - 8

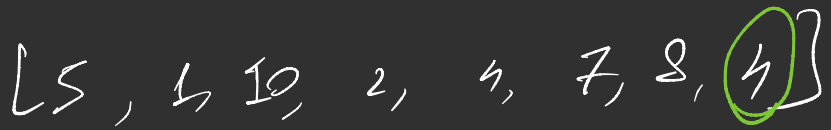
B_8 B_1 B_5 B_4 B_6 B_2 B_7 B_3

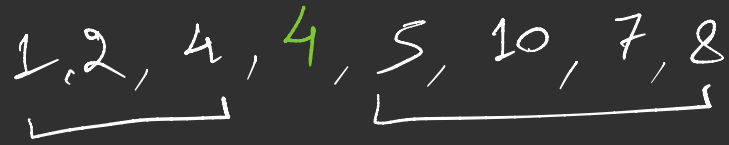
N_2 N_5 N_3 N_8 N_1 N_4 N_7 N


N₂ 6, 4, 3, 1, 2, 5
B₂ 2, 1, 4, 5, 3, 6
6
6
N₂ 4, 3, 1, 2, 5
B₂ 2, 1, 4, 5, 3
4
3
N₂ 3, 1, 2
B₂ 2, 1, 3
(3, 3)
5
5
1, 2
2, 1



Let’s come to Quick Sort now





Comparison b/w Merge & Quick Sort

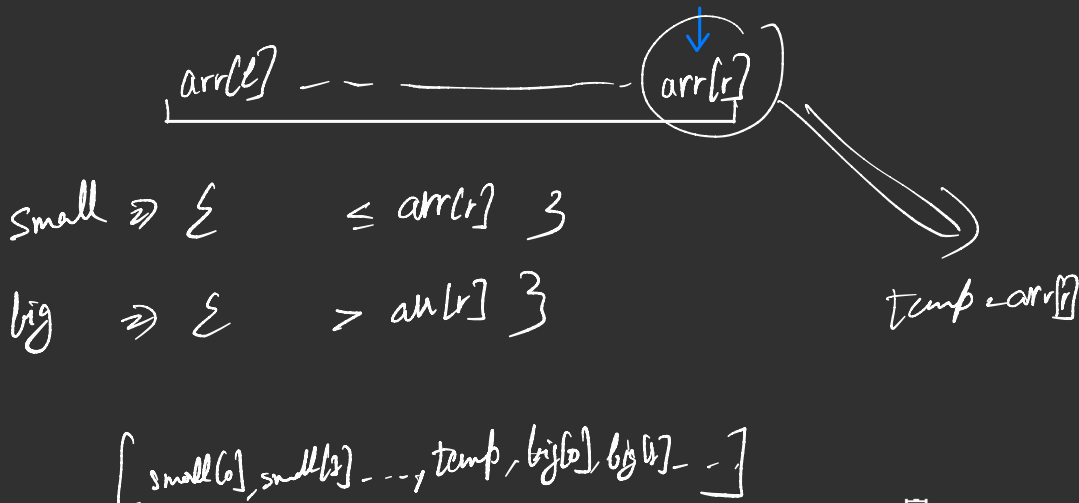
Merge Sort

- Divide ⟹ $\frac{N}{2}$ $\frac{N}{2}$ — O(1)
- Conquer ⟹ sort(first half) sort(sec half) — Recursion
- Combine ⟹ Main Logic — O(N)

Quick Sort

- Divide ⟹ Main Logic — O(N)
- Conquer ⟹ sort(left part) sort(right part)
- Combine ⟹ Nothing to do

Let's first get the partition function out of the way



Now, let’s understand the time complexity

Best Case

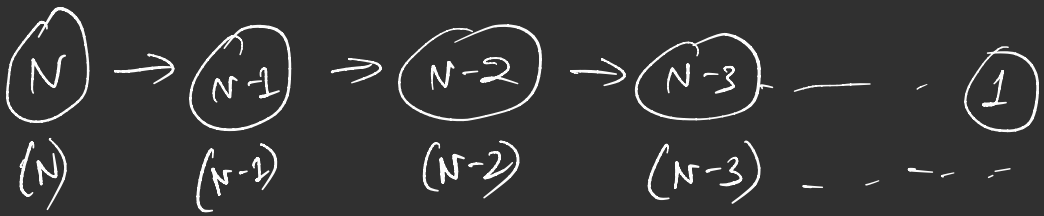
$$T(N) = 2 * T(N/2) + N$$

Time $\Rightarrow O(N \log N)$

Worst Case

$\Rightarrow T(N) = T(N-1) + N$

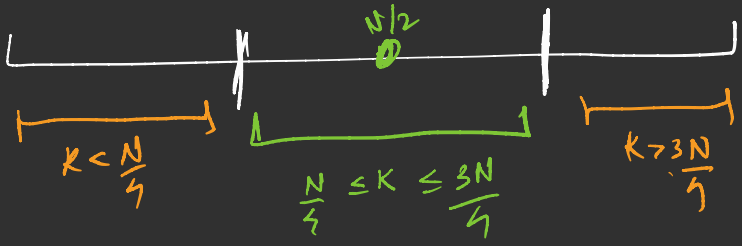
$T(N-1) = T(N-2) + (N-1)$



Time = $O(N^2)$

$$T(N) = T(K) + T(N-K) + N$$

Average Case



0.5 ⇒ good Scenario ⇒

0.5 ⇒ Bad Scenario

Proving the Average Case Time Complexity

Good Scenario

$K \geq N/4$

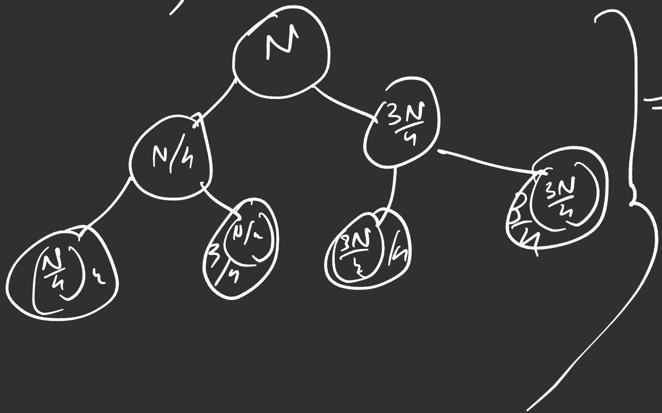
$K \leq \frac{3N}{4}$

$$T(N) = T\left(\frac{N}{4}\right) + T\left(\frac{3N}{4}\right) + N$$

$\frac{1}{2}^{th}$

$\Downarrow$

$\log_2 N$



$\Rightarrow (3/4)^{th}$

$\log_{4/3} N$

$N * \log_{2/3} N$

$2.6\, N \log_2 N$

$O(N \log_2 N)$

(Avg. Case)

In 2 steps, $N \xrightarrow{2 \text{ steps}} \frac{3N}{4}$

$2.6\, N \log N$

$\Downarrow$

$5.2\, N \log_2 N \Longrightarrow$ Time $\Downarrow$ $O(N \log_2 N)$

Pseudo-code Comparison b/w MS & QS

```
ms(arr, l, r) {

    left = arr(l, mid);
    right = arr(mid+1, r);

    ms(left)
    ms(right);

    merge(left, right);
}
```

```
qs(arr, l, r) {

    id = partition(arr, l, r);

    qs(l, id-1);
    qs(id+1, r);
}
```

Thank You!

Reminder: Going to the gym & observing the trainer work out can help you know the right technique, but you'll muscle up only if you lift some weights yourself.

So, PRACTICE, PRACTICE, PRACTICE!