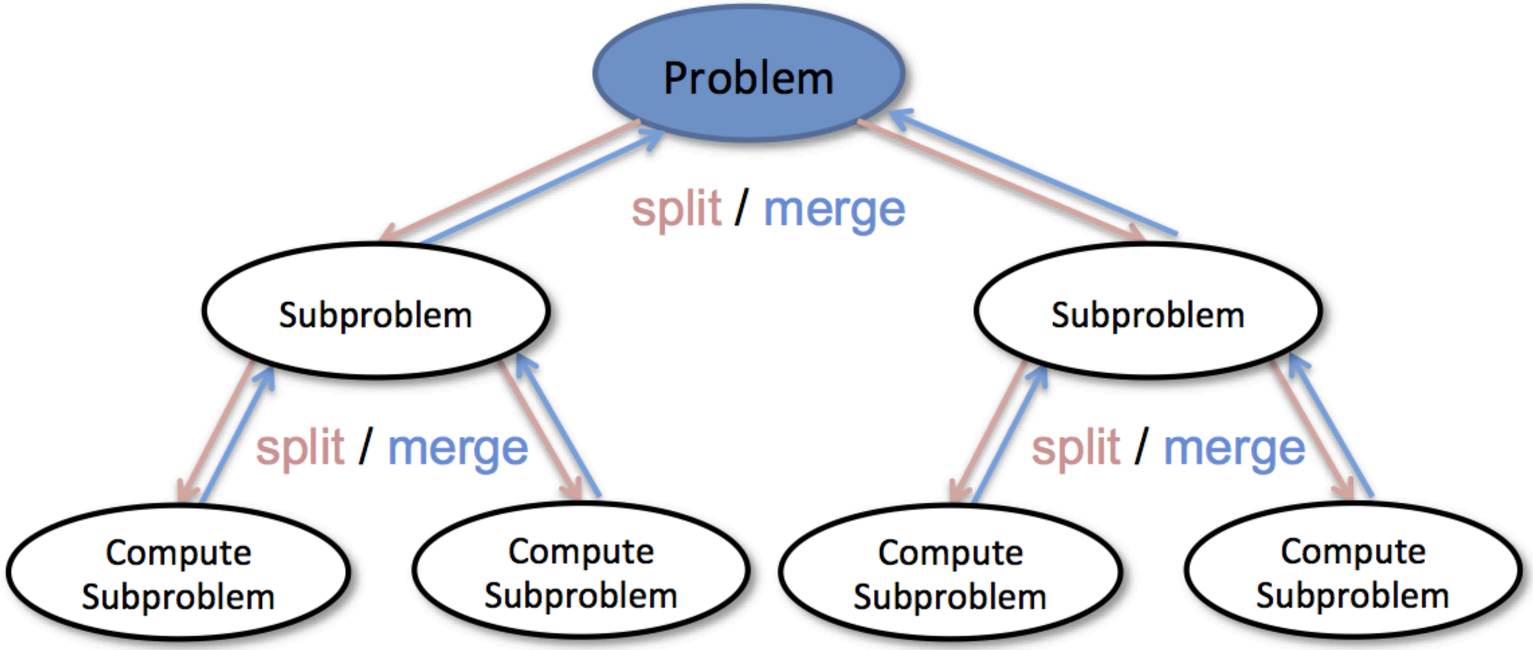
Algorithms & Data Structures I: Merge Sort

Mikhail Anukhin

Today's Topics

- Divide & Conquer Technique Recall
- Merge Sort
- Merge Sort Analysis
- Count Inversions in an array

Divide & Conquer Technique



Divide & Conquer Technique

1. Given a problem size of *n* divide it into subproblems size of *n/b. b≥1.*
2. Solve *a* problems recursively
3. Combine solutions of subproblems to get overall solution.

General formula of asymptotic:

T(n) = a * T(n/b) + [work for merge]

Divide & Conquer Technique

General formula of asymptotic:

T(n) = a * T(n/b) + [work for merge]

- **Theorem statement:** Let we have a recurrence

A

$$T(n) = \begin{cases} a \cdot T\left(\frac{n}{b}\right) + O(n^c), & n > 1 \\ O(1), & n = 1 \end{cases}$$

then asymptotic solution will be:

B

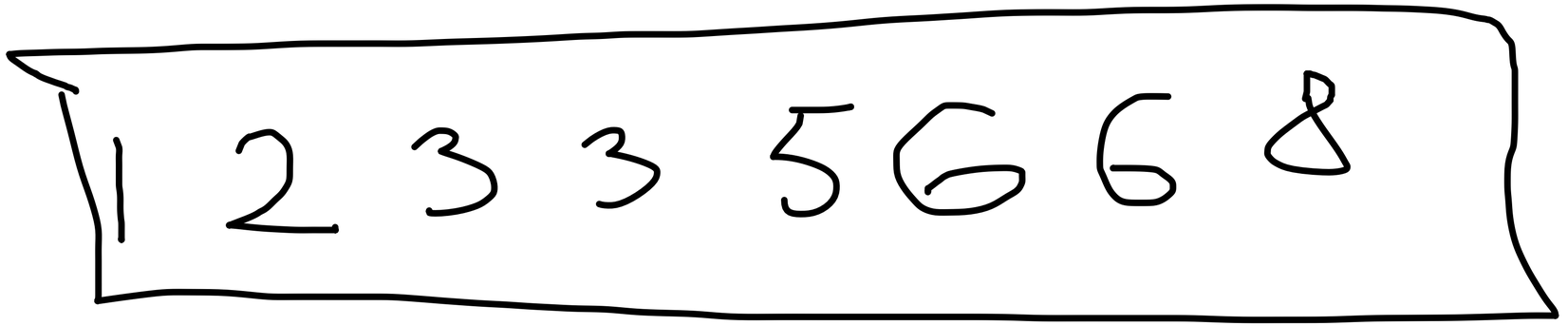
1. If $c > \log_b a$, then $T(n) = O(n^c)$
2. If $c = \log_b a$, then $T(n) = O(n^c \cdot \log n)$
3. If $c < \log_b a$, then $T(n) = O(n^{\log_b a})$

Merge Sort

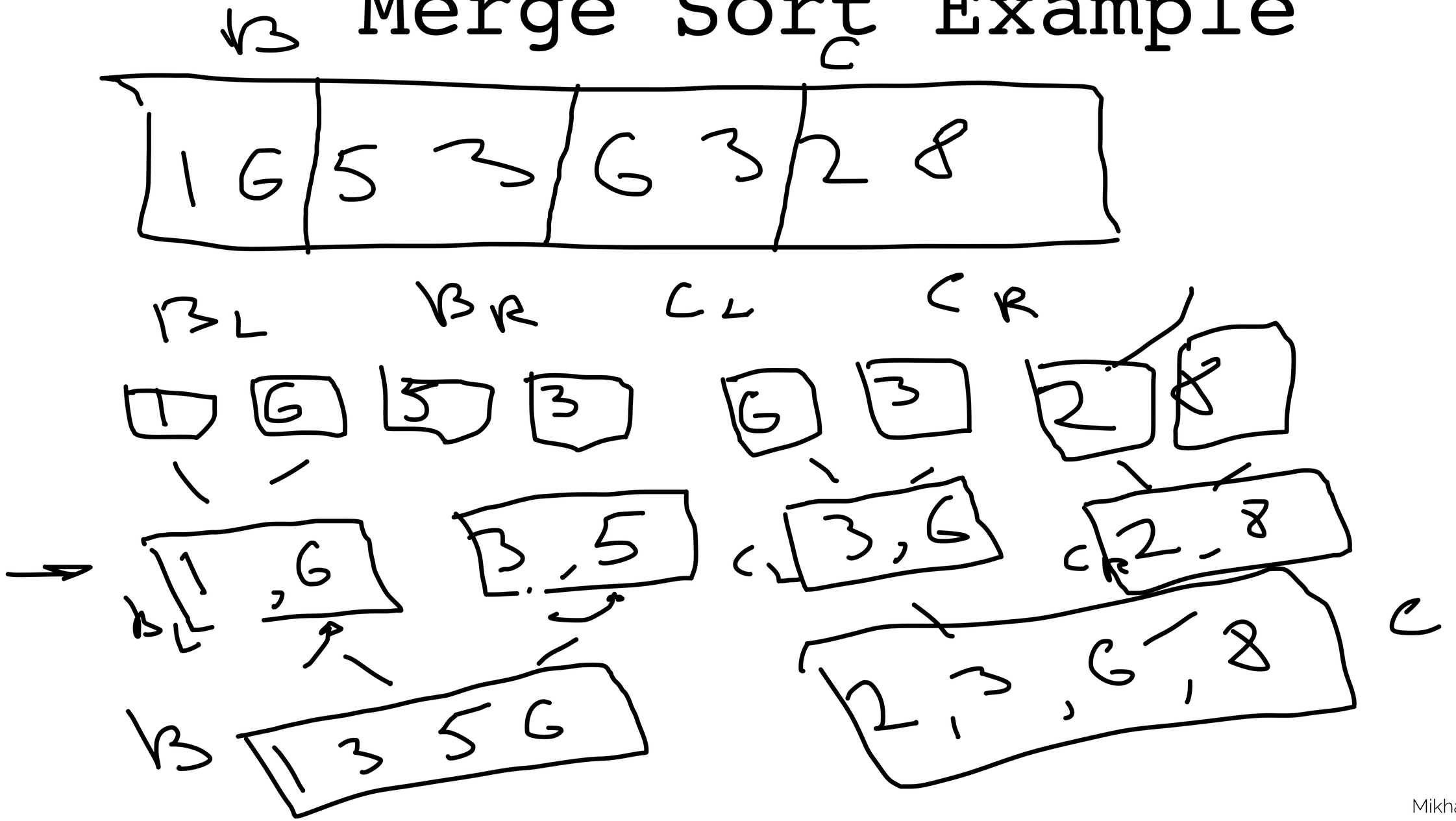
n - size of an array

```
MergeSort(A[l..r])
    1. If n = 1, done
    2. Otherwise, recursively sort A[1..n/2] and
       A[n/2+1..n]
    3. Merge the two sorted sub-arrays
```

Merge Sort Example

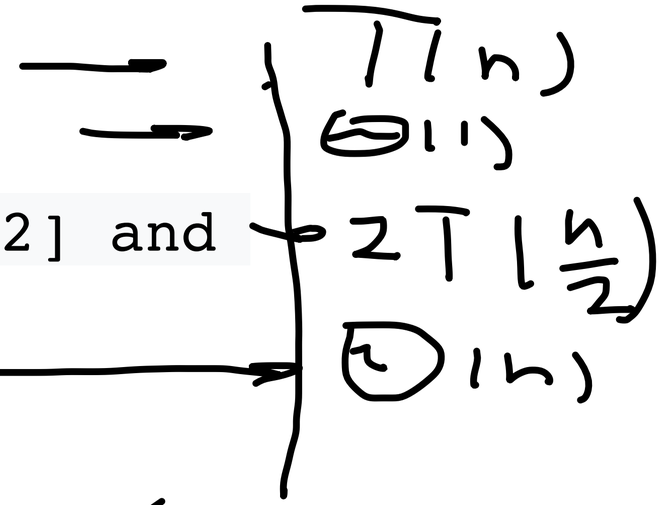


Merge Sort Example



Merge Sort Analysis

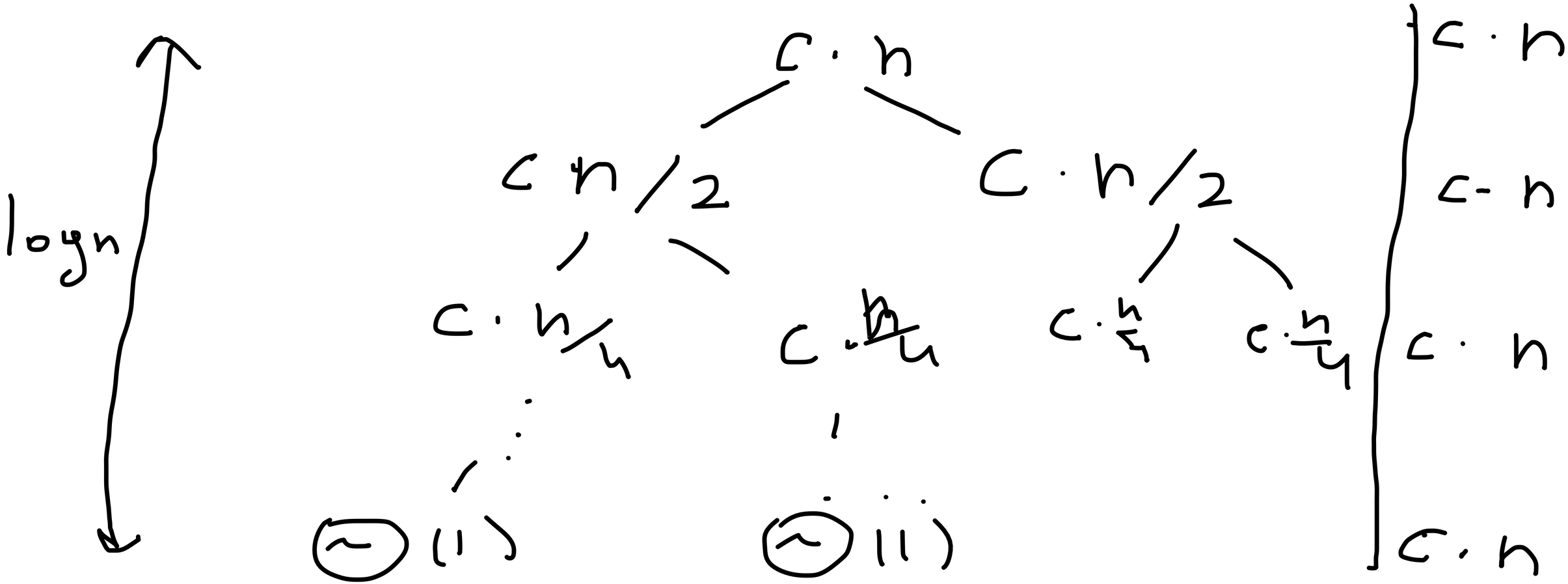
```
MergeSort(A[l..r])
    1. If n = 1, done
    2. Otherwise, recursively sort A[1..n/2] and
       A[n/2+1..n]
    3. Merge the two sorted sub-arrays
```



$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + \Theta(n), & n > 1 \\ \Theta(1), & n = 1 \end{cases}$$

Merge Sort Recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n, \quad c > 0$$



Merge Sort Recurrence Tree

$$T(n) = \log n \cdot c \cdot n = O(n \cdot \log n)$$

Merge Sort Master Theorem

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)$$

$a = 2$
$b = 2$
$c = 1$

$\log_2 2 = 1$

$\log_b a = c$

$$T(n) = O(n \cdot \log n)$$

Mikhail Anukhin

Number of Inversions in an array

0 1 2 3

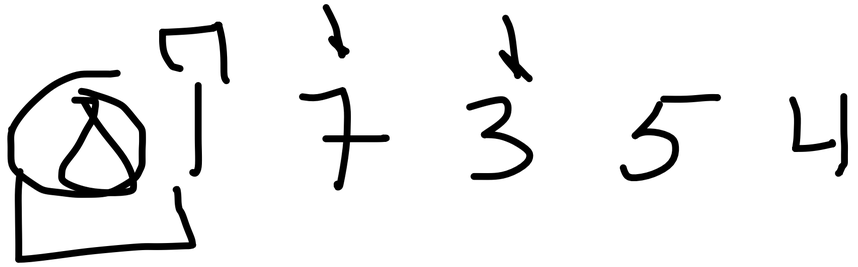
4 1 2 3

$A[a_1, \ldots, a_n]$

(i, j) is an inversion

if $\begin{cases} i < j \\ a_i > a_j \end{cases}$

Number of Inversions Idea



inv_counter = 0

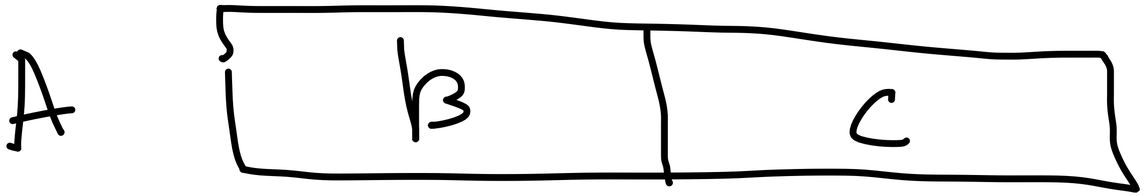
Naive Algorithm

1. Iteratively take every element and compare with every right-side element.

Time Complexity: $\sum_{i=1}^{n-1} i = \Theta(n^2)$

Number of Inversions Example

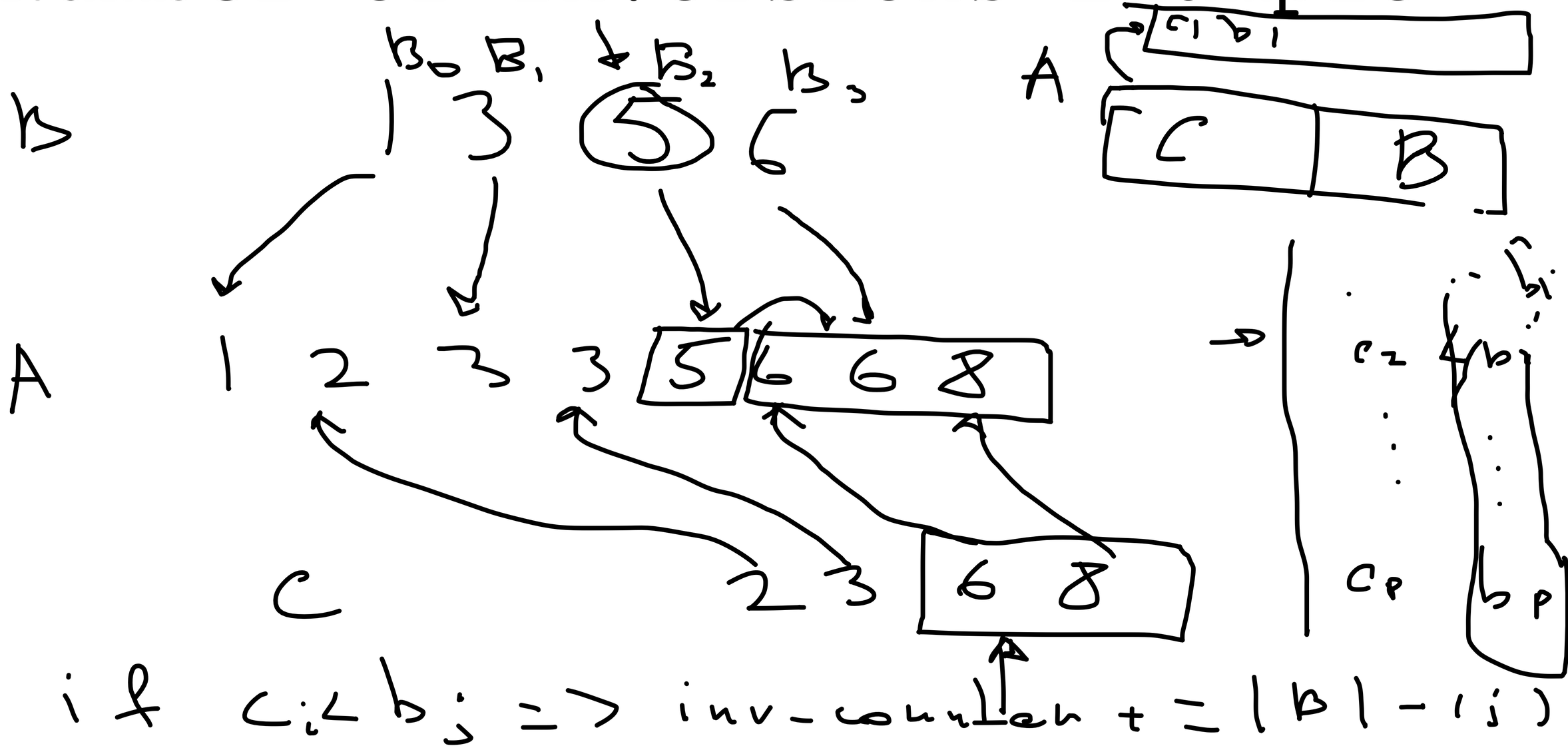
$F(A)$ - number of inversions in A



$F(A) = F(B) + F(C) + \#$ pairs

$(i, j) : B_i > C_j$

Number of Inversions Example



Number of Inversions Time analysis

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = O(n \cdot \log n)$$

Your questions!