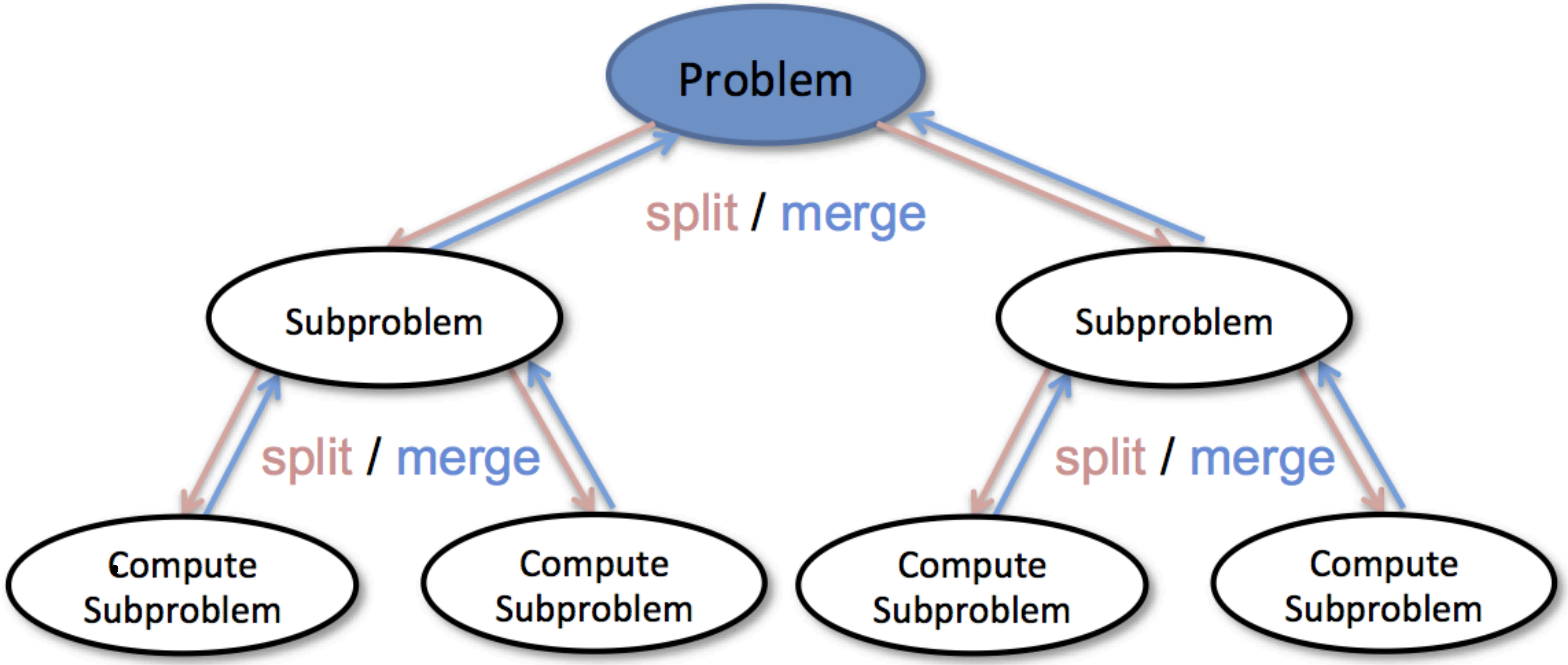
Algorithms & Data Structures I: Quick Sort

Mikhail Anukhin

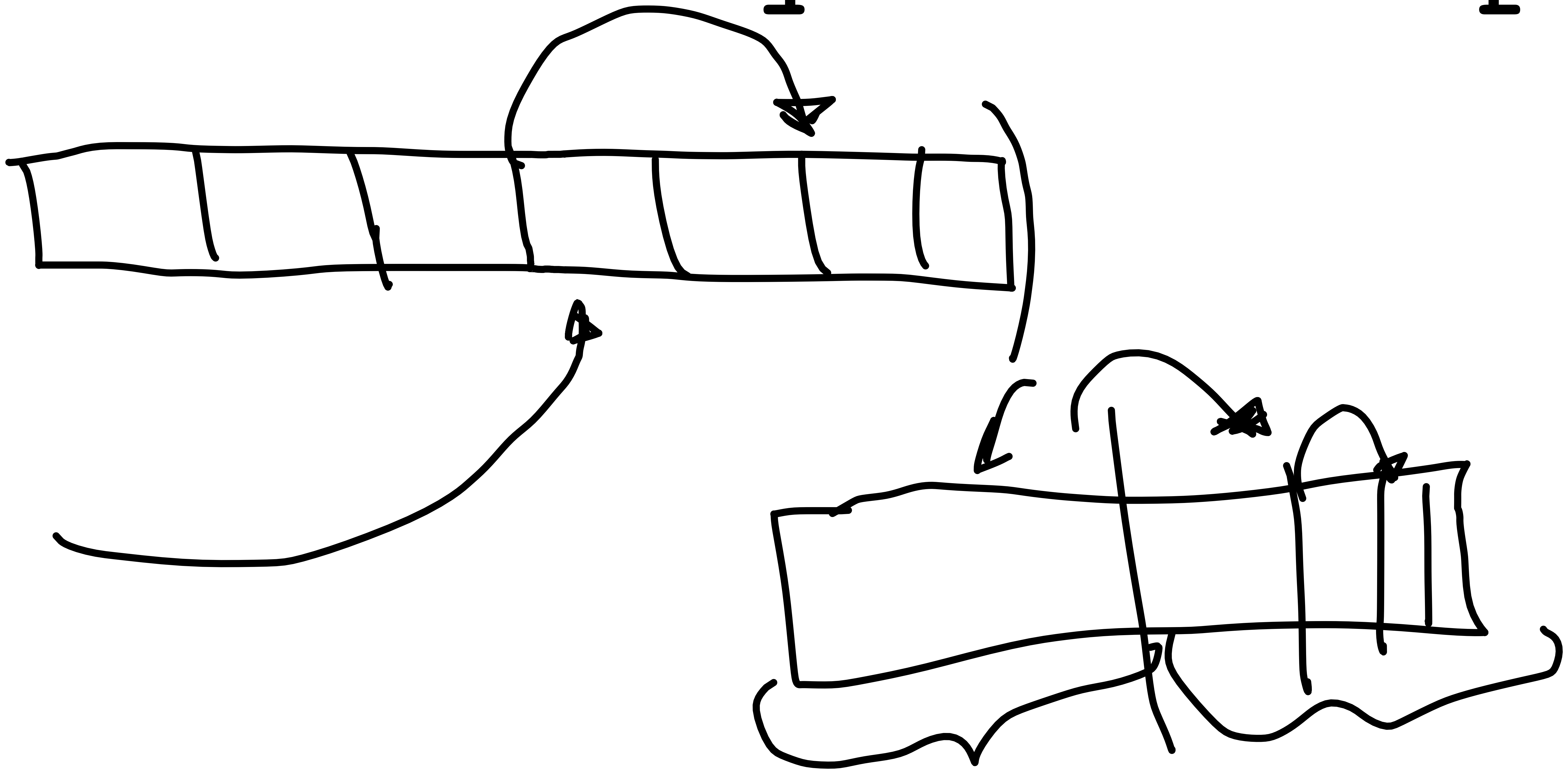
Today's Topics

- Divide & Conquer Technique Overview
- Quick Sort
- Quick Sort Analysis
- Master Theorem

Divide & Conquer Technique



Divide & Conquer Example



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]

Mikhail Anukhin

Quick Sort

- Proposed By Toni Hoar in 1962
- Divide-and-conquer algorithm
- Very practical. Used in std::sort

Quick Sort Divide & Conquer

Quick sort an *n*-element array:

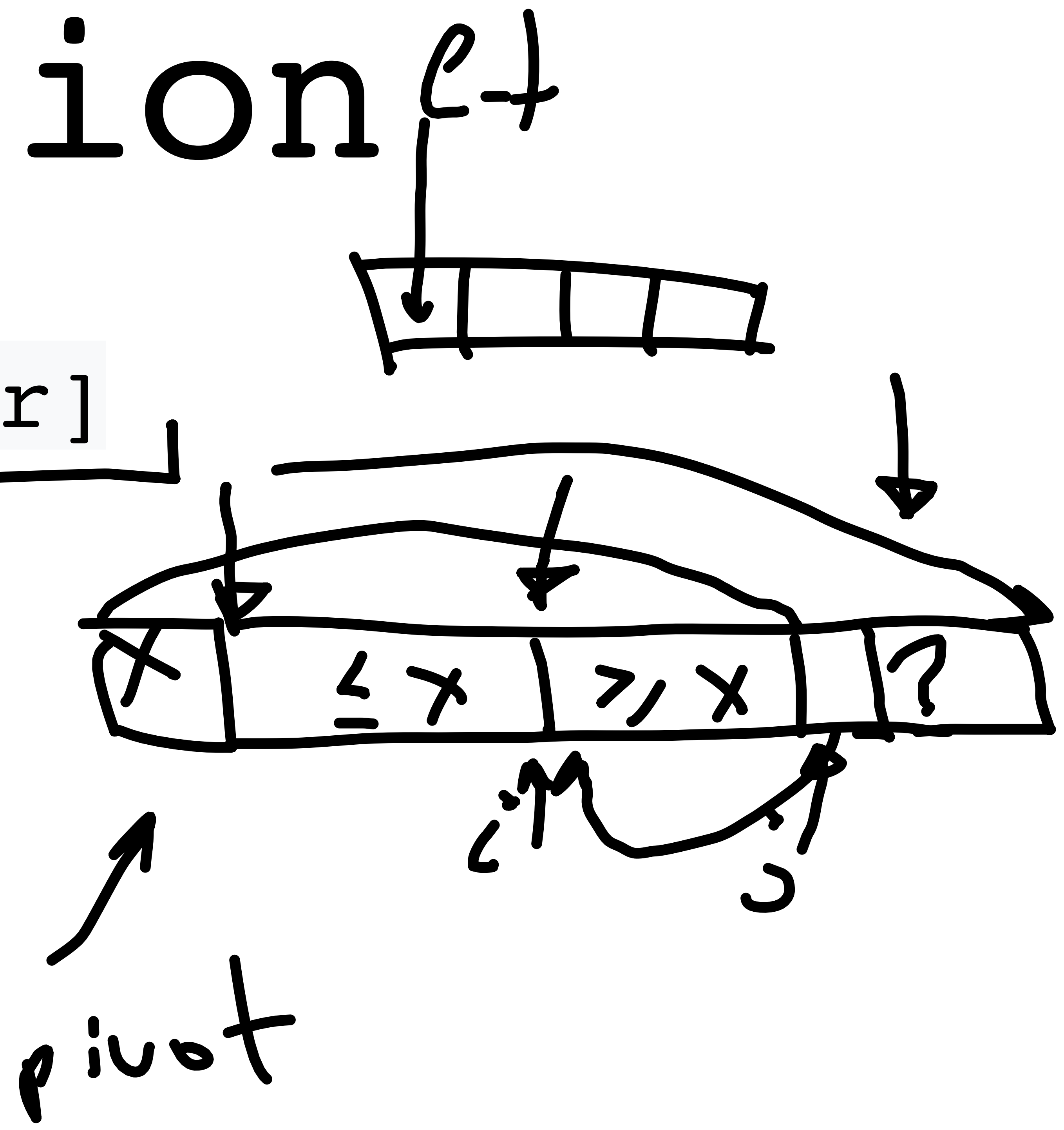
- **Divide:** Partition the array into two subarrays around a *pivot* **X** such that elements in lower subarray ≤ x ≤ elements in upper subarray

≤ X	X	≥ X

- **Conquer:** Recursively sort the two subarrays
- **Combine:** Trivial

Quick Sort Partition

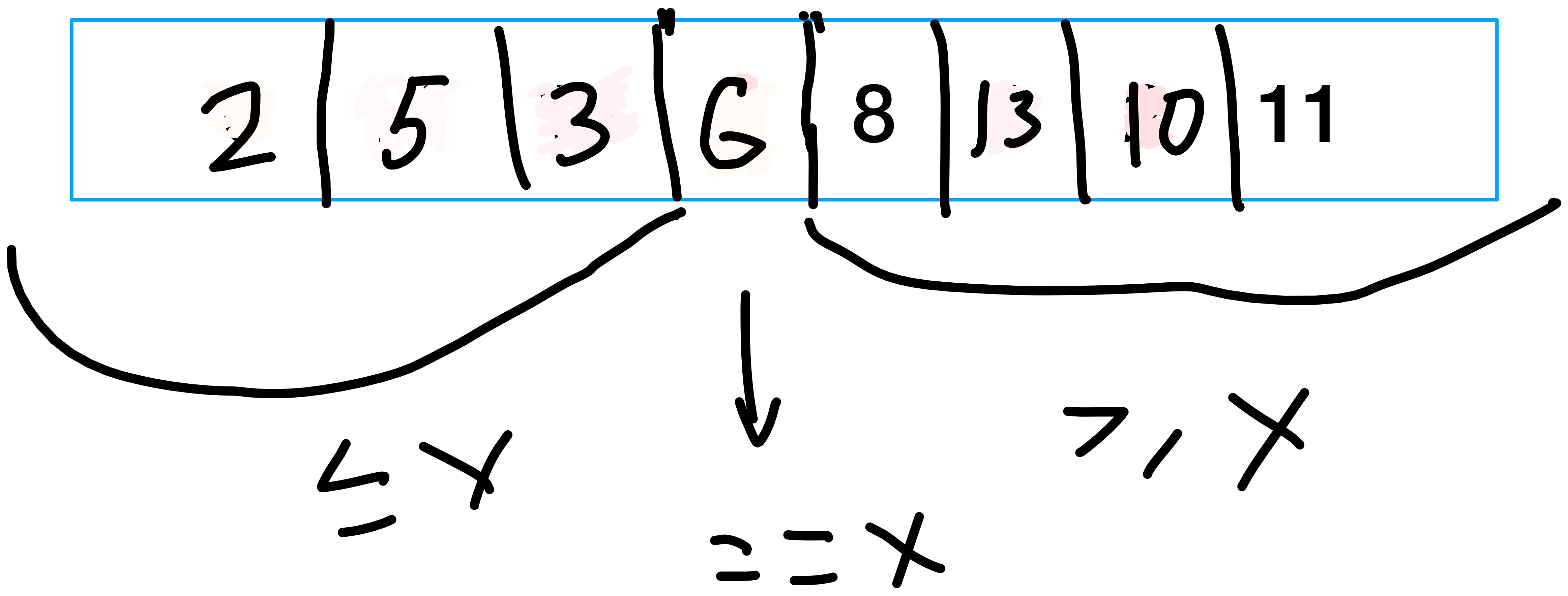
```
Partition(array A, l, r): // A[l..r]
    pivot = A[l]
    i = l
    for j = l + 1 to r
      if A[j] ≤ pivot:
        i = i + 1
        swap (A[i], A[j])
    swap (A[l], A[i]
    return i
```



Time complexity: $O(n)$

Quick Sort Partition Example

```
Partition(array A, l, r):
    pivot = A[l]
    i = l
    for j = l + 1 to r
      if A[j] ≤ pivot:
        i = i + 1
        swap (A[i], A[j])
    swap (A[l], A[i])
    return i
```



Quick Sort Partition Example

```
Partition(array A, l, r):
    pivot = A[l]
    i = l
    for j = l + 1 to r
      if A[j] ≤ pivot:
        i = i + 1
        swap (A[i], A[j])
    swap (A[l], A[i]
    return i
```

6	10	13	5	8	3	2	11

Quick Sort Pseudocode

```
QuickSort(array A, l, r): // A[l..r]
    if l < r:
      q = Partition(A, l, r)
      QuickSort(A, l, q-1)
      QuickSort(A, q+1, r)
```

Mikhail Anukhin

Quick Sort Worst Case

- Reverse order
- Partition around min element
- One side of partition always has no elements

$$\boxed{1}\quad \overbrace{\boxed{2}\;\; 3 \;\; 4 \;\; 5}^{n-1}$$

$\le x$ $\quad =: x \quad$ $\ge x$

$$\boxed{\text{Time Complexity: } O(n^2)}$$

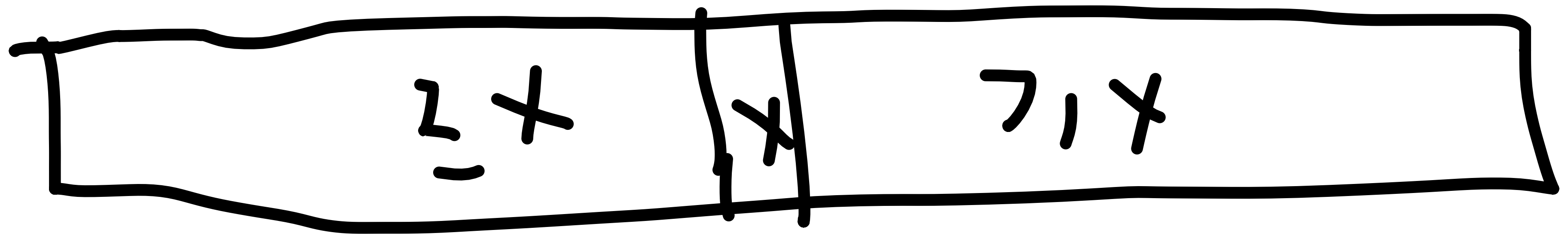
$$T(n) = T(0) + T(n-1) + \Theta(n) =$$

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

$$\sim n \cdot c + (n-1) \cdot c + (n-2) \cdot c + \ldots 1 \cdot c = c \cdot \frac{n \cdot (n-1)}{2} = O(n^2)$$

Quick Sort Best Case

- What if we always splits the array evenly?



Time Complexity:
$O(n \cdot \log n)$

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

$$= 2 \cdot T\left(\frac{n}{2}\right) + c \cdot n = 4 \cdot T\left(\frac{n}{4}\right) + c \cdot n + c \cdot \frac{n}{2} =$$

$$= n \cdot T(0) + \ldots + \left(c \cdot n + \frac{n}{2} \cdot c + \frac{n}{4} \cdot c + \ldots\right) \leq c \cdot n \cdot \log n + n$$

Quick Sort Average

St. In average case Quicksort performs in $O(n \cdot \log n)$.

- Choose a pivot as a random element -> then Partition splits the array evenly in the average case.

Master Theorem Motivation

- $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$
- $T(n) = T\left(\frac{n}{2}\right) + O(1)$

Master Theorem Statement

- **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 $T(n) = O(n^{\log_b a})$

Mikhail Anukhin

Master Theorem Examples

- Binary Search

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

$$T(n) = 1 \cdot T\left(\frac{n}{2}\right) + O(n^0)$$

$a = 1, b = 2, c = 0$

$$0 = \log_2 1 = 0$$

$$T(n) = O(n^0 \cdot \log n) = O(\log n)$$

Your questions!