Algorithms & Data Structures I: Sortings

Mikhail Anukhin

Today's Topics

- The sorting problem
- Why sorting?
- Insertion Sort
- Binary Insertion Sort
- Bubble Sort
- Number of comparisons required to sort an array

The problem of sorting

- Input:
  - array A[1…n] of numbers.
- Output:
  - permutation B[1…n] of A such that B[1] ≤ B[2] ≤ … ≤ B[n]
  - e.g. A = [7, 2, 5, 5, 9.6] → B = [2, 5, 5, 7, 9.6]

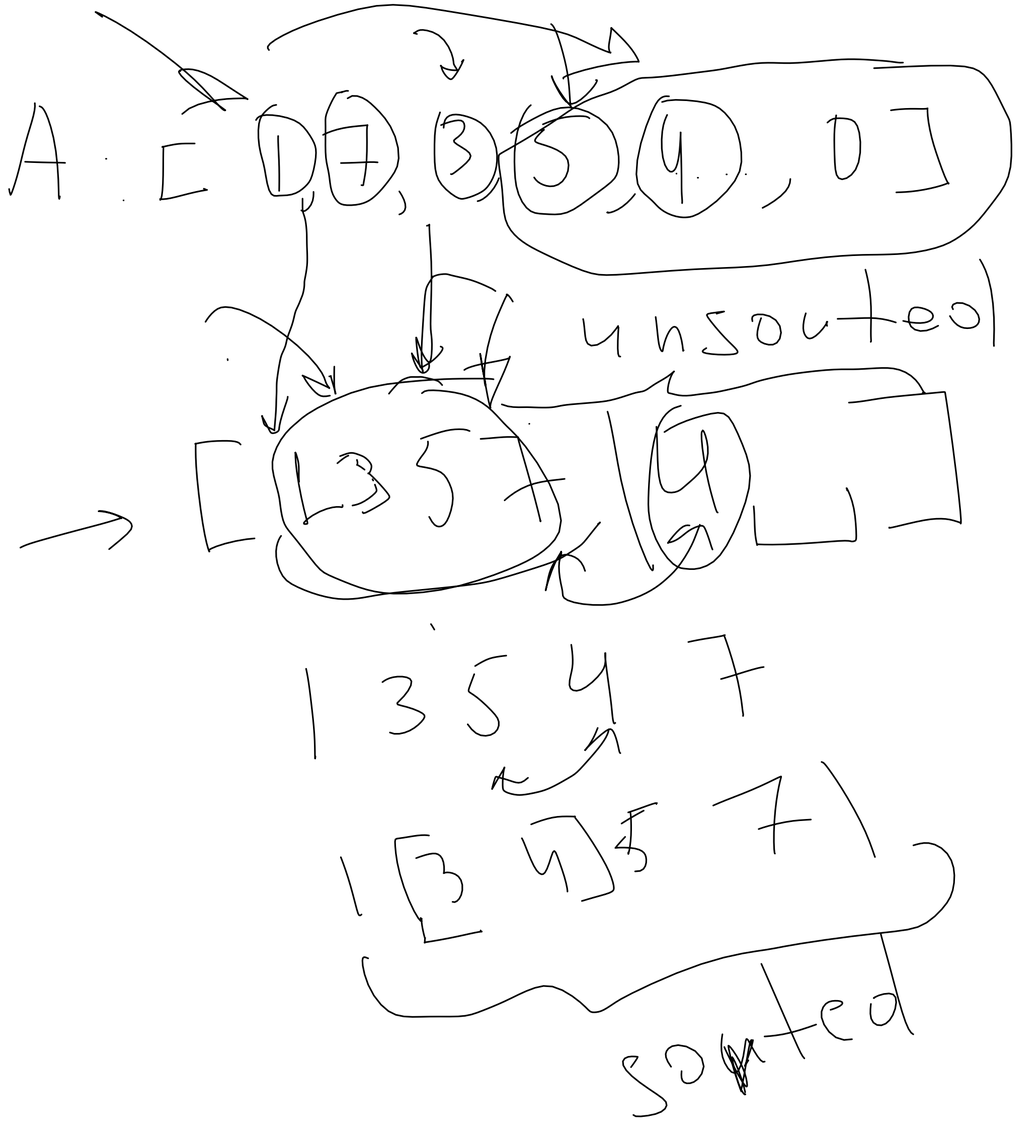
Applications

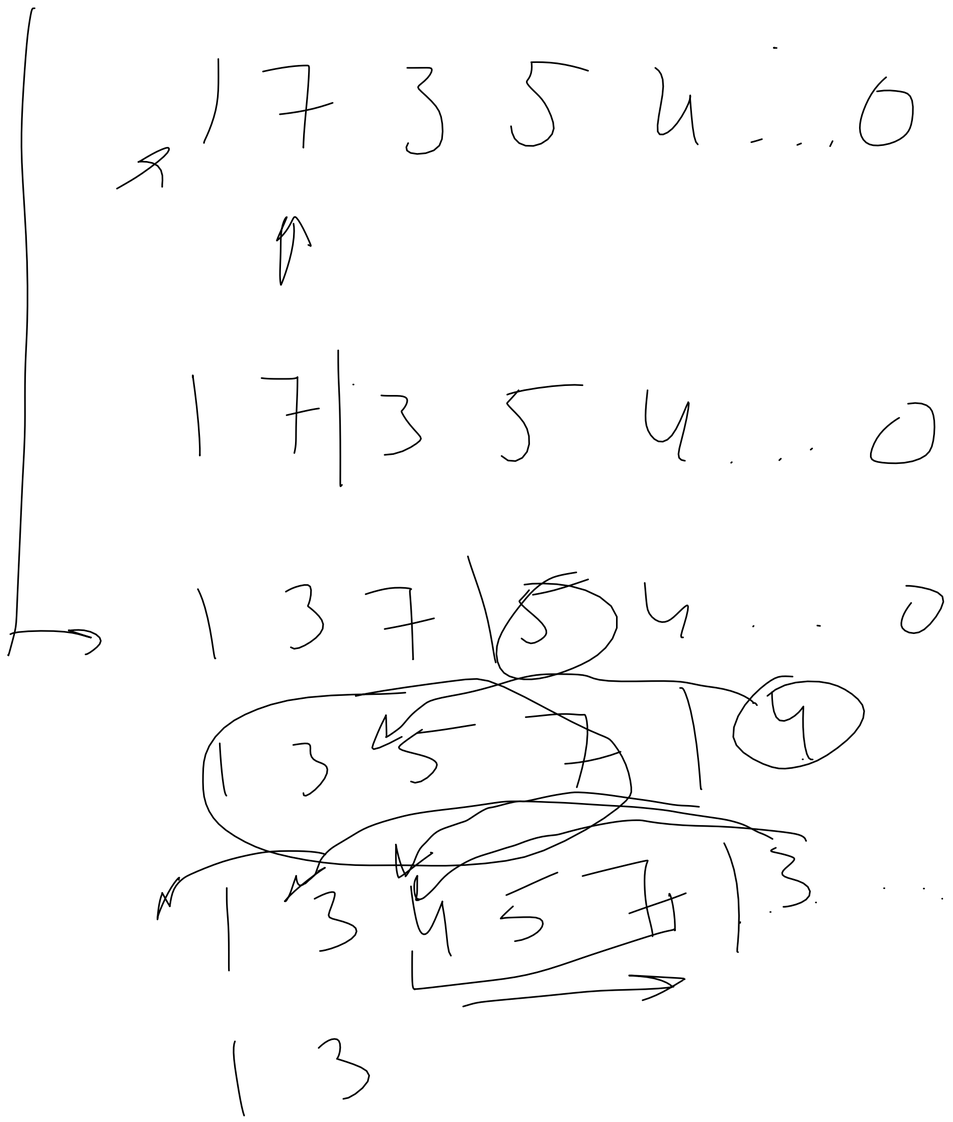
- Obvious applications
  - Organize an iTunes library
  - Maintain a telephone directory

- Problems that become easy once items are in sorted order
  - Find a median, or find closest pairs
  - Binary search, identify statistical outliers

- Non-obvious applications
  - Data compression: sorting finds duplicates
  - Computer graphics: rendering scenes front to back

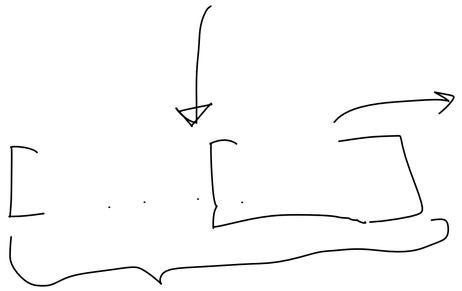
Insertion Sort Idea





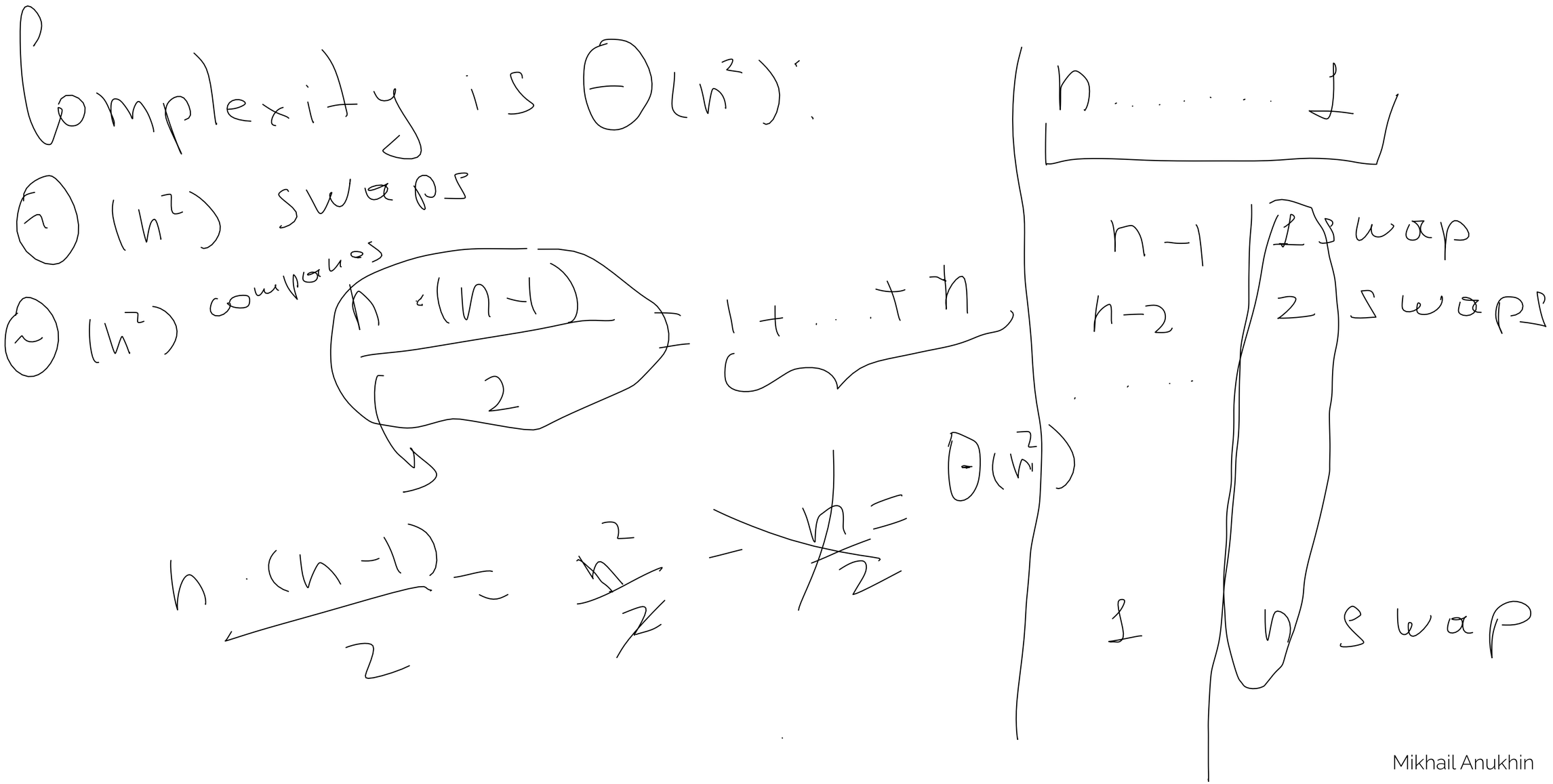
Insertion Sort Algorithm

```
InsertionSort(array A, n):
  for j = 2 to n
    insert key A[j] into the (already sorted) sub-array
    A[1..j-1] by pairwise key-swaps down to its right
position
```



Insertion Sort Example

Insertion Sort Analysis



Mikhail Anukhin

Binary Insertion Sort Algorithm

```
BinaryInsertionSort(array A, n):
  for j = 2 to n
    insert key key A[j] into the (already sorted) sub-array
    A[1 .. j-1]. Use binary search to find the right position
```

Complexity: $\Theta(n \log n)$ Comparisons + $\Theta(n^2)$ swaps / $\Theta(n^2)$

Binary Insertion Sort Example

Worst

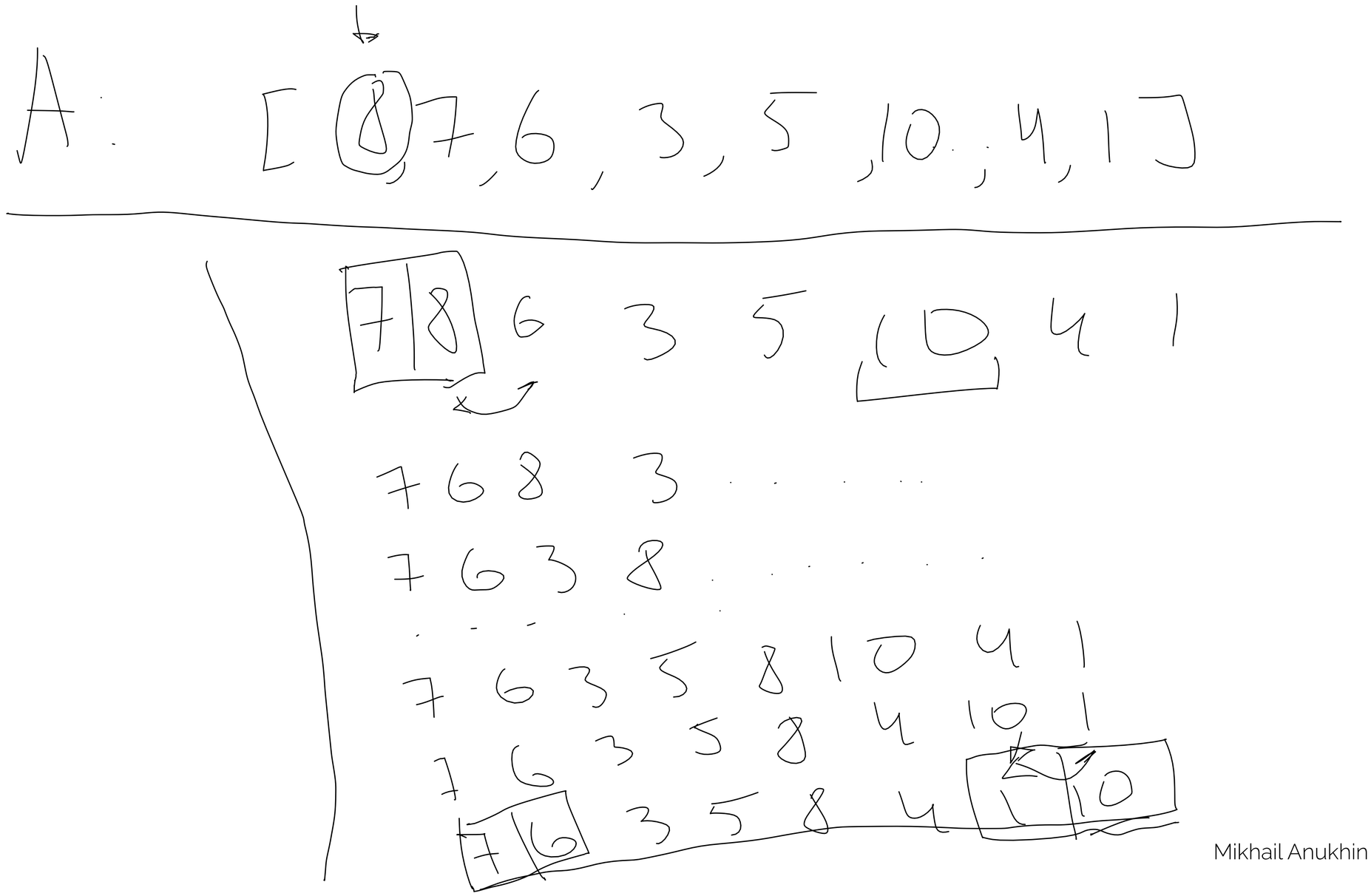
$(n-1) \cdot \log(n-1) \sim O(n \cdot \log n)$

Binary Insertion Sort Example

Binary Insertion Sort Analysis

Mikhail Anukhin

Bubble Sort Idea



Mikhail Anukhin

Bubble Sort Algorithm

```
bubbleSort(array list):
    for i = 0 to list.size()-1 do:
     swapped = false

     for j = 0 to list.size()-1 do:
        if list[j] > list[j+1] then
           swap(list[j], list[j+1])
           swapped = true
     if(not swapped) then
        break
     return list
```

Bubble Sort Example

6 5 3 1 8 7 2 4

Bubble Sort Analysis

We need to find maximum n times.

$$[n + (n-1) + (n-2) \dots + 1] = \frac{n \cdot (n-1)}{2} \sim \Theta(n^2)$$

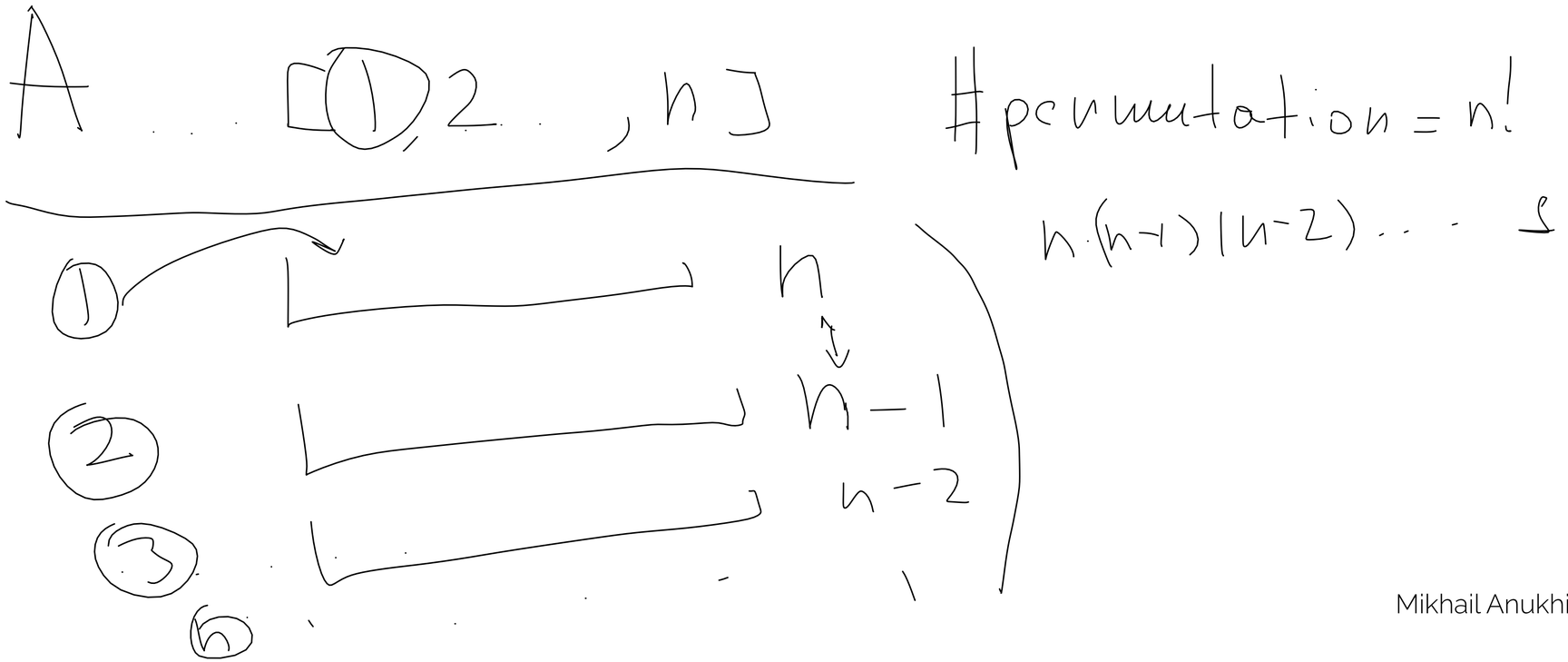
Number of comparisons required to sort an array

- We observed a few algorithms. Can we do better?
- Where are the limits?
- What is the most efficient sorting?

Theorem

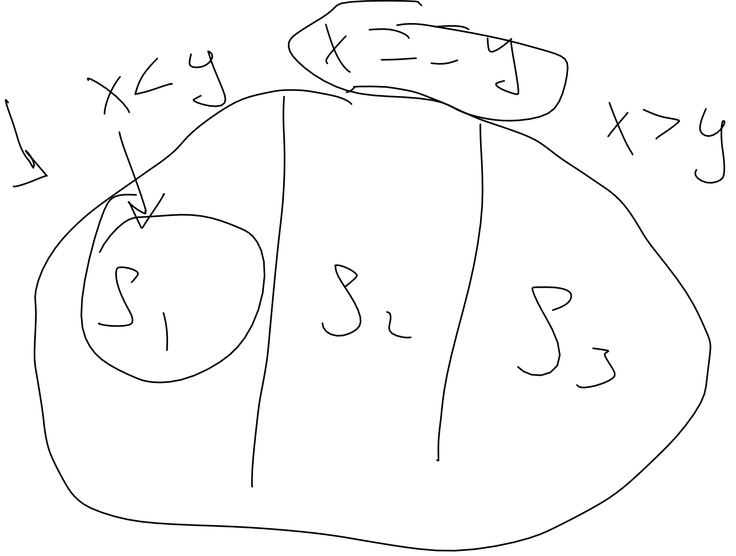
- **Theorem statement:**
  - The number of comparisons that a comparison sort algorithm requires increases in proportion to n * log(n), where n is the number of elements to sort. → $\Omega(n \cdot \log n)$

Proof:



Theorem

Proof:



$\log(n!) = (\log n!) \leq \Omega(n \cdot \log n)$

$\geq$

$\log n + \log(n-1) + \log(n-2) \cdots + \log \frac{n}{2} > \frac{n}{2} \log\left(\frac{n}{2}\right) =$

$= \frac{n}{2}(\log n - \log 2) = \frac{n}{2} \log n - \frac{n}{2} \log 2 = \Omega(n \cdot \log n)$

Conslusion

We need at least n*log n operations to sort an array using comparisons!

Your questions!