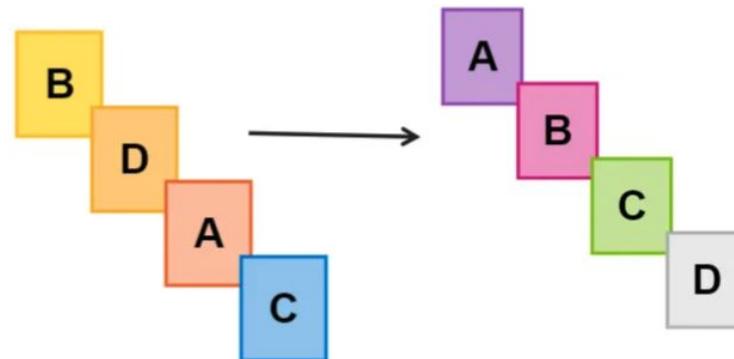


Bogo Sort

Kian Ahrabian

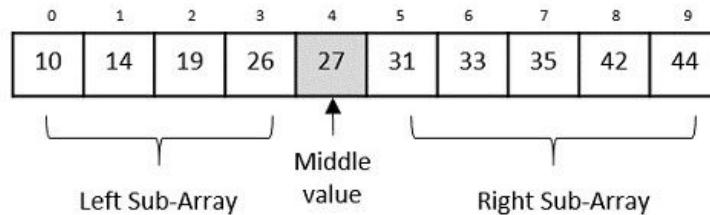
What is sorting?

- **Input:** We are given an array of **objects** without any particular order.
- **Output:** We want an ordered array based on a specific **criteria**.
- **Example:** We are given an array containing descriptions of various **cars** (i.e., objects) and we want to sort them based on **ascending horsepower** (i.e., criteria).



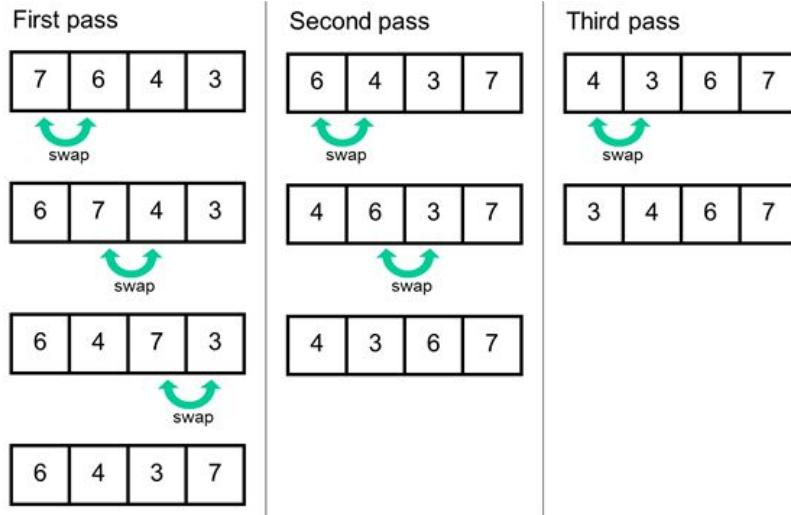
Why do we need sorting?

- Enables efficient searching and processing of data.
- **Example:** Given an ordered array with N elements, we can search for an object in $O(\log N)$ instead of $O(N)$ using binary search. While there is an overhead computation for sorting the array, as the number of searches increase, we save more time.

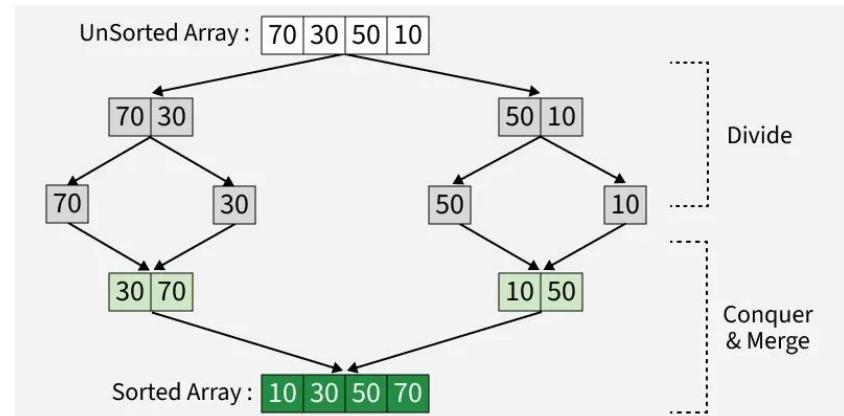


Examples of efficient sorting algorithms

- Bubble Sort → $O(N^2)$

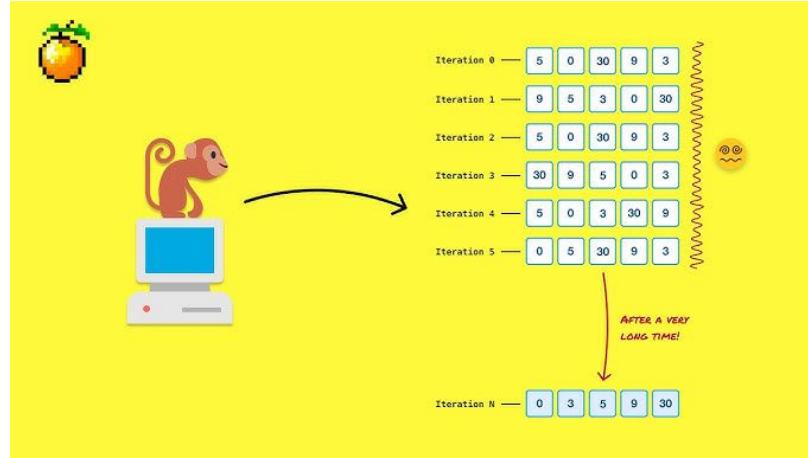


- Merge Sort → $O(N \log N)$



What about the worst sorting algorithms?

- Any algorithm requiring more than N^2 comparisons is considered inefficient.
- Today we talk about **Bogo Sort** (also called *Permutation Sort* or *Stupid Sort*).

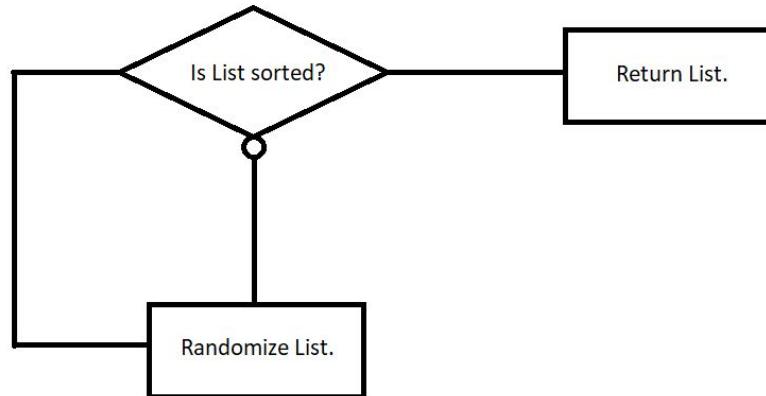


Bogosort: Sorting in the Slow Lane!



How does Bogo Sort works?

- Given an array \mathcal{A} , we do the following steps:
 - Shuffle \mathcal{A}
 - Check whether \mathcal{A} is sorted or not
 - Stop if sorted, otherwise repeat



Example of Bogo Sort

- $\mathcal{A} = \{6, 4, 3, 5\}$
- Step 1: $\mathcal{A} \rightarrow \{4, 6, 5, 3\}$ ✗
- Step 2: $\mathcal{A} \rightarrow \{5, 6, 4, 3\}$ ✗
- Step 3: $\mathcal{A} \rightarrow \{3, 4, 5, 6\}$ ✓

How fast is Bogo Sort?

- **Best Case:** The first permutation gives us the sorted array, which requires $2N$ operations (shuffling + verifying) → $O(N)$
- **Average Case:** Since we are randomly shuffling the array, the probability of getting the correct ordering is $(1 / N!)$. Hence, which means that we need $N!$ repetitions to have an expected value of 1 → $O(N \cdot N!)$
- **Worst Case:** There is no limit on the number of permutations → **Unbounded**

Thank you for listening!