Welcome to "Little Bits to Big Ideas"

Lab 4: Sorting Algorithms and their Computational Complexity

TAs: Ashish & Shivani

CAs: Abhinav, Harry, & Sherry

ILLINOIS

Big ideas about Algorithms

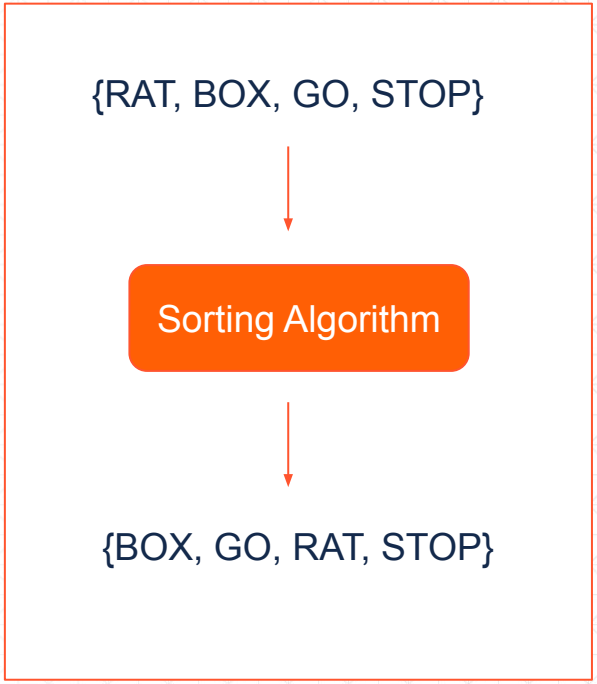
- The same functionality can be performed by different algorithms
- Some of those algorithms could be better than others for various reasons
  a. Faster
  b. More correct
  c. Easier for humans to understand

What is a Sorting Algorithm?

Start with a **list**

Re-arrange the contents of the list (This part varies between algorithms)

End with a **sorted list**



Why do we care about sorting algorithms?

- They are a common way to explore big ideas about algorithms and computational complexity.

- Knowing a little bit about them might give you some credibility with computer scientists (see an example by President Obama)

Candidate Obama interviewed at Google in 2007

See how he charmed his audience with a reference to a sorting algorithm.

(Turn on captions)

Explaining BubbleSort

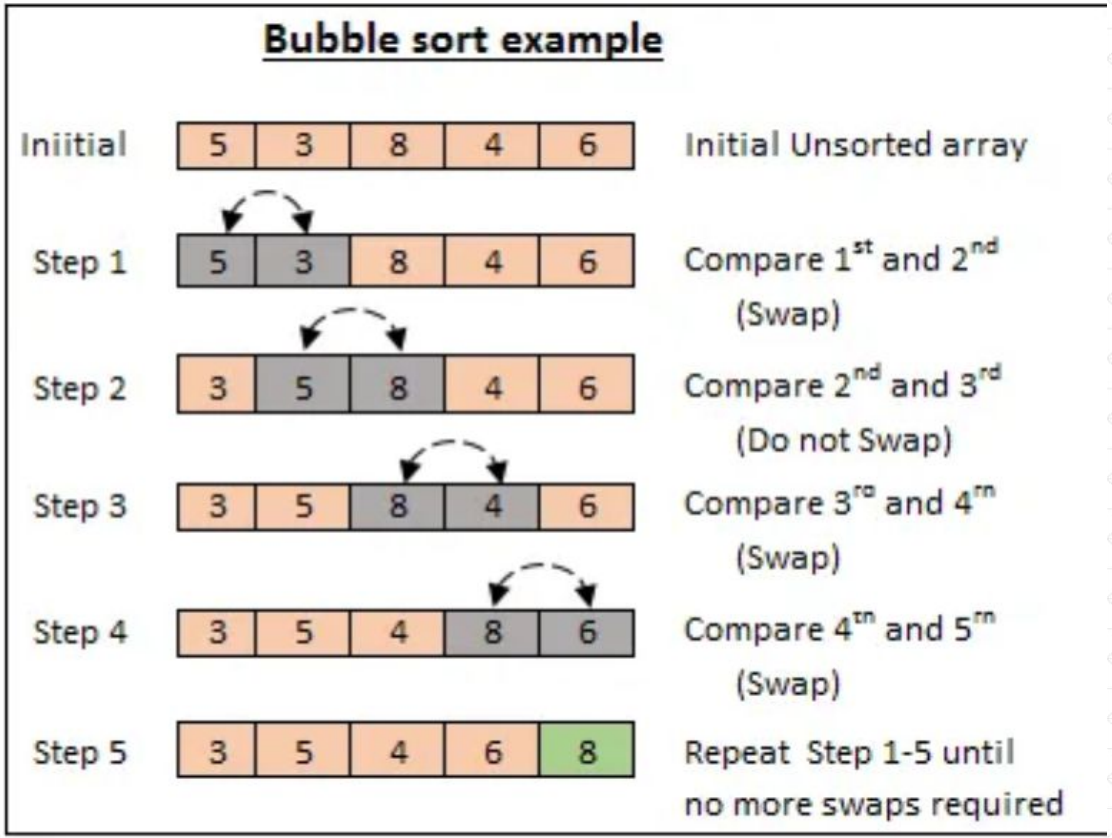
English description:

- Repeatedly swap adjacent elements if they are in the wrong order.
- Keep doing this until the list is sorted.

Pseudocode description:

```
for i from 0 to n-1:
    for j from 0 to n-i-1:
        if array[j] > array[j+1]:
            swap(array[j], array[j+1])
```

Explaining BubbleSort



BubbleSort Demo with Hungarian Dance

Explaining MergeSort

English description:

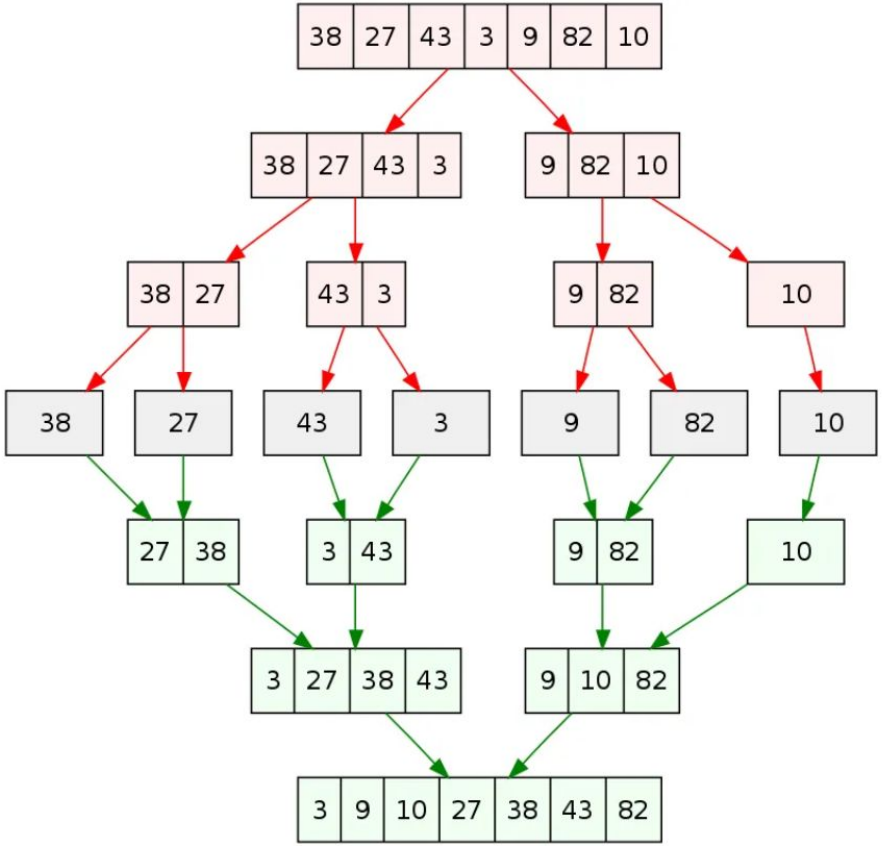
- Divide the list into halves until single elements remain.
- Merge sorted halves back together in order.

Pseudocode description:

```
function mergeSort(array):
    if length of array <= 1:
        return array
    mid = length of array / 2
    left = mergeSort(array[:mid])
    right = mergeSort(array[mid:])
    return merge(left, right)
```

Explaining MergeSort

Demo with handing back exams or with cards



Today you're using Google Co-Lab to run Python Code

This is a type of Notebook

See a demo before you get started

Warning: Answers after this point

Answers to complexity

BubbleSort - O(n^2)

SelectionSort - O(n^2)

MergeSort - O(nlog(n))

QuickSort - O(nlog(n))

Python built-in sorting algorithm - (nlog(n))

- But it’s written in C which helps it be faster!