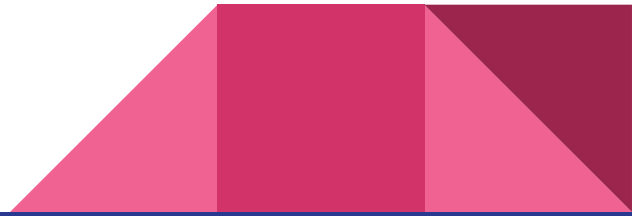
Sorting Algorithms

Do Now

Given the following list of numbers : –2, 45, 0, 11, -9

Describe a strategy to put these numbers in order from smallest to largest

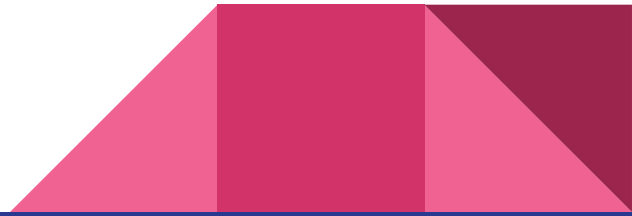
Note: Your strategy should work for any list of numbers, including a list that has some duplicate values



Bubble Sort

Bubble Sort is one of the most widely discussed algorithms, simply because of its lack of efficiency for sorting arrays.

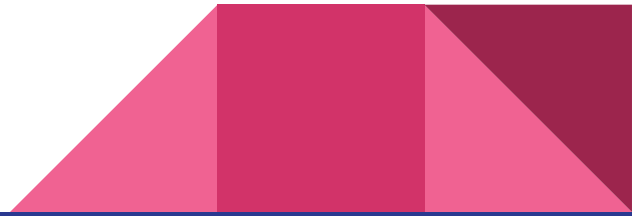
If an array is already sorted, Bubble Sort will only pass through the array once.



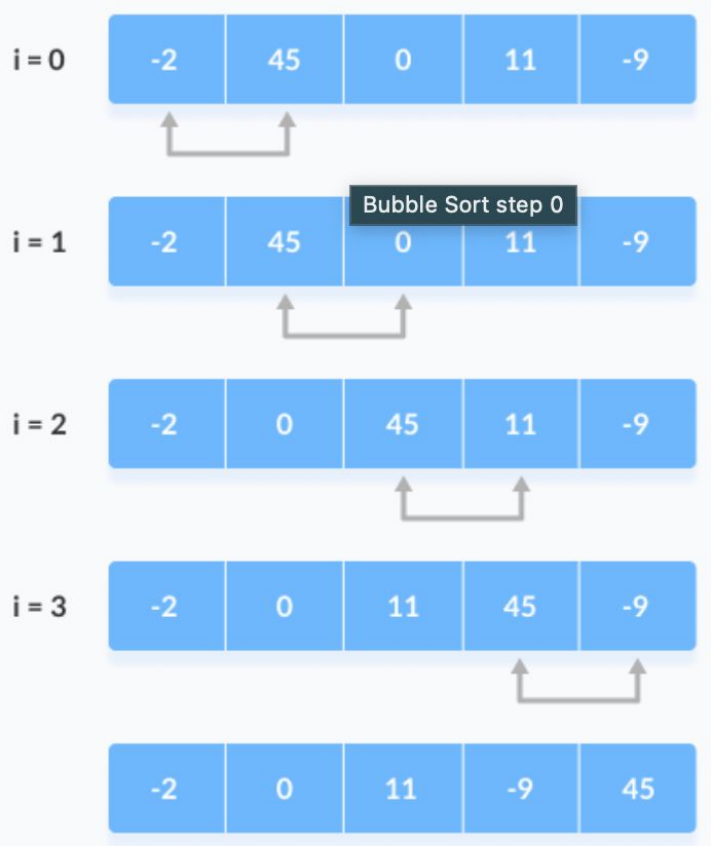
How does Bubble Sort work?

This algorithm compares two adjacent elements and swaps them until they are in the intended order.

As the algorithm progresses we have a sorted partition and an unsorted partition (This is a logical partition, we do not have two arrays)



Bubble Sort - Step 1 (Pass 1)

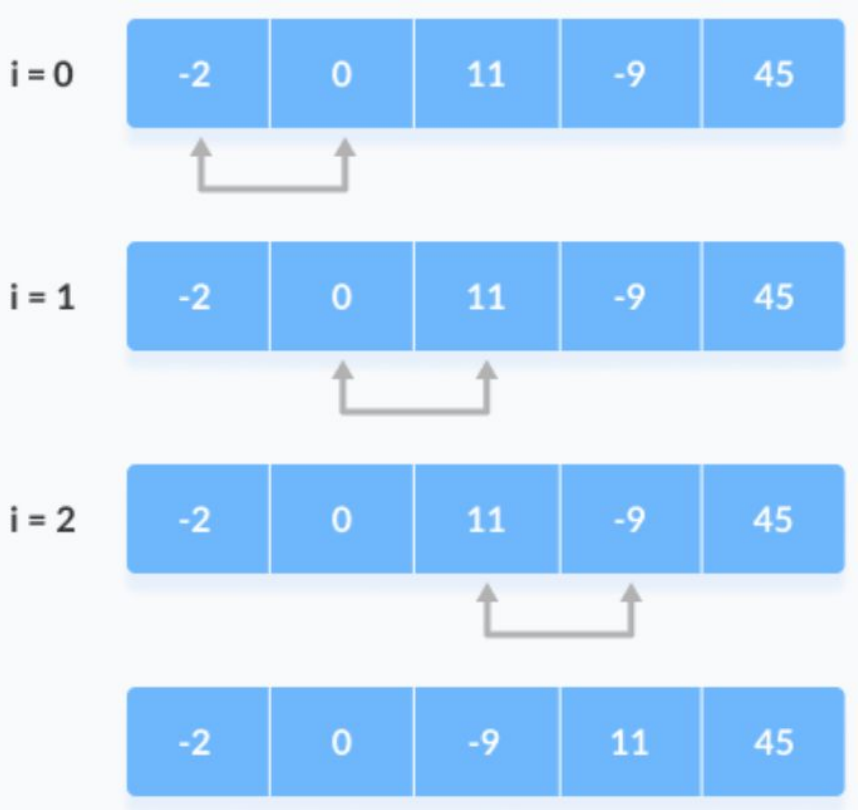


1. Starting at index 0, compare the first and the second elements.
2. Swap the element if the first one is greater than the second one.
3. Next, compare the second and the third elements and swap them if the order is not correct.
4. Keep doing the process until your algorithm reaches the last element.

What do you notice? Is any element at its correct position?

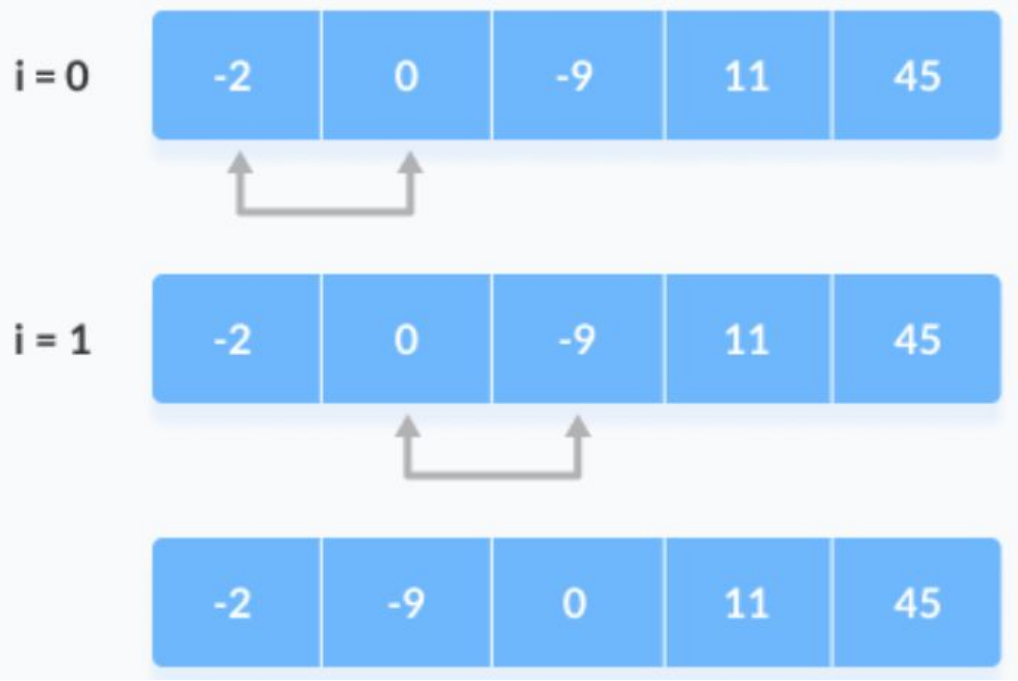
A logical sorted partition starts forming with the largest element placed at the end.

Bubble Sort - Step 2 (Pass 2)

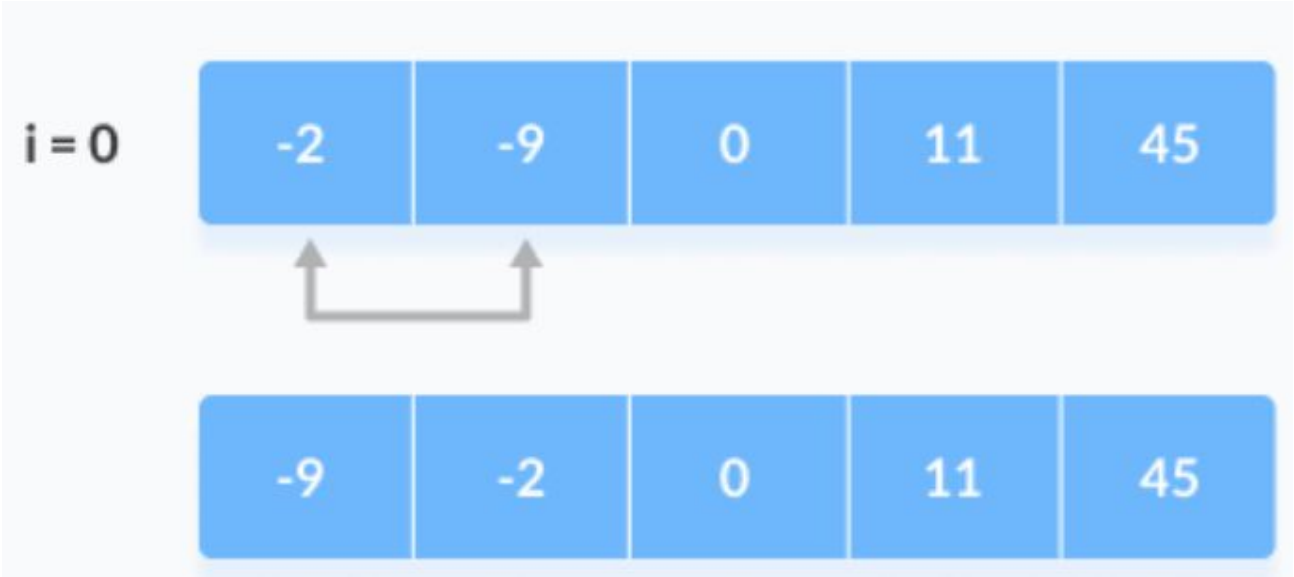


Repeat the comparing and swapping for the remaining iterations (unsorted partition).

Bubble Sort - Step 3 (Pass 3)



Bubble Sort - Step 4 (Pass 4)



Discussion Time

Read the Bubble Sort algorithm again. Then answer the following:

1. If the array is filled with all 0's, what should happen during sorting?
2. How many swaps should occur in this case, and why?
3. What could cause this algorithm to run much slower than it should?
4. How could you improve the algorithm so it runs faster in this scenario?

Bubble Sort Algorithm

1. Starting at index 0, compare the first and the second elements.
2. Swap the element if the first one is greater than the second one.
3. Next, compare the second and the third elements and swap them if the order is not correct.
4. Keep doing the process until your algorithm reaches the last element.

Optimized Bubble Sort

This is a **faster version of Bubble Sort** that stops early when the list is already sorted.

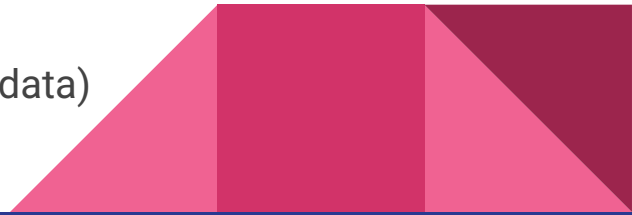
Strategy

Use a **swap flag** to detect whether any swap happened during a pass.

- If a swap occurs => keep sorting
- If **no** swaps occur => array is sorted => **stop immediately**

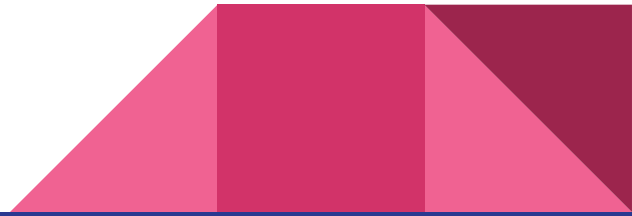
Why is it better?

- Avoids unnecessary passes
- Improves best-case time complexity
- Makes Bubble Sort **adaptive** (performs better on nearly sorted data)



Optimized Bubble Sort - Steps

1. Start each pass with **swappedFlag = false**
2. Compare adjacent elements; swap if needed
3. If any swap occurs, set **swappedFlag = true**
4. After the pass, if **swappedFlag == false**, **break** out of the loop

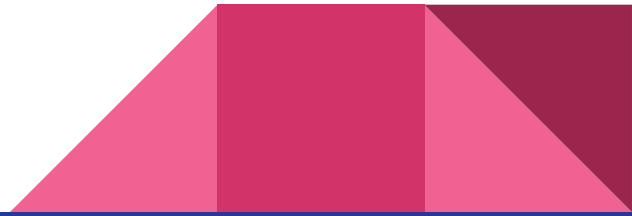


Bubble Sort Advantages

- Bubble sort is easy to understand and implement.
- It’s an adaptive sorting algorithm. The order of elements affects the time complexity of the sorting.

Bubble Sort Disadvantages

- Time complexity of **?** which makes it very slow for large data sets.
- It is not efficient for large data sets, because it requires multiple passes through the data.



Bubble Sort Applications

- It is often used to introduce the concept of a sorting algorithm because it is very simple.
- This algorithm is not efficient for real life applications unless:
    - Complexity does not matter
    - Short and simple code is preferred

Bubble Sort Visualizer

https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/visualize/