Bubble Sort

In this lesson, we'll learn how bubble sort works and see the implementation.

We'll cover the following

Bubble sort

For each element starting from the first, compare with the adjacent element and swap the two if they are in incorrect order, i.e., `arr[i] > arr[i+1]`. The first iteration places the largest element at `arr[N-1]`. The second iteration places the second largest at `arr[N-2]` and so on...

- Initially:[6 3 5 4 1 2]
- Iteration 1: [**6 3** 5 4 1 2] -> [3 **6 5** 4 1 2] -> [3 5 **6 4** 1 2] -> [3 5 4 **6 1** 2] -> [3 5 4 1 **6 2**] -> [3 5 4 1 2 6]
- Iteration 2: [**3 5** 4 1 2 6] -> [3 **5 4** 1 2 6] -> [3 4 **5 1** 2 6] -> [3 4 1 **5 2** 6] -> [3 4 1 2 5 6]
- Iteration 3: [**3 4** 1 2 5 6] -> [3 **4 1** 2 5 6] -> [3 1 **4 2** 5 6] -> [3 1 2 4 5 6]

For example, in the third iteration, we can stop after the first four elements because the previous two iterations will make sure the last two elements are larger than the first four elements.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int N = 6;
  int arr[6] = {6, 3, 5, 4, 1, 2};

  for (int i = 0; i < N; i++) {
    for(int j = 1; j < N - i; j++) { // skip comparing with last i elements
      if (arr[j] < arr[j-1])
        swap(arr[j], arr[j-1]);
    }
  }

  for (int i = 0;i < N; i++) cout << arr[i] << " "; cout << "\n";
```

```
  return 0;
}
```




In the next lesson we'll see insertion sort, another $O(N^2)$ sorting algorithm.