C++ Sorting Algorithms

Implementation and Visualization

Mr. Gullo

July 2, 2025

Learning Objectives

After this presentation, you will:

- Understand five different sorting algorithms
- Be able to implement each sorting algorithm in C++
- Know the advantages and disadvantages of each method
- Recognize the time complexity of different algorithms

Initial Array

Numbers to Sort

Our example will use these 9 numbers: [7, 2, 9, 4, 5, 8, 3, 6, 10]

Bubble Sort

Algorithm Description

- Repeatedly steps through the list
- Compares adjacent elements and swaps them if needed
- Continues until no swaps are needed

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
                swap(arr[j], arr[j+1]);
}
```

Bubble Sort Steps

7	2	9	4	5	8	3	6	10
2	7	4	5	8	3	6	9	10
2	4	5	7	3	6	8	9	10
2	4	5	3	6	7	8	9	10
2	4	3	5	6	7	8	9	10
2	3	4	5	6	7	8	9	10

Selection Sort

Algorithm Description

- Divides array into sorted and unsorted regions
- Finds minimum element in unsorted region
- Swaps it with first element of unsorted region

```
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int min_idx = i;
        for (int j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;
        swap(arr[min_idx], arr[i]);
    }
}
```

Time Complexity Comparison

Time Complexity

- Bubble Sort: $O(n^2)$
- Selection Sort: $O(n^2)$
- Insertion Sort: $O(n^2)$
- Quick Sort: $O(n \log n)$ average, $O(n^2)$ worst case
- Merge Sort: $O(n \log n)$

Summary

Key Points

- Bubble Sort: Simple but inefficient
- Selection Sort: Simple and performs well on small lists
- Insertion Sort: Efficient for small data sets
- Quick Sort: Generally the fastest in practice
- Merge Sort: Consistent performance but requires extra space