## Project – Sorting

|  |  |  |  |
| --- | --- | --- | --- |
| Performance Analysis Results | | | |
| **Data Size** | **Bubble Sort** | **Quick Sort** | **STL Sort** |
| 100 | 0 | 0 | 0 |
| 1000 | 2 | 0 | 0 |
| 10000 | 301 | 1 | 1 |
| 100000 | 33859 | 12 | 15 |

## Performance Differences

The performance differences between these sorting algorithms are significant and stem from their fundamental approaches to sorting. Bubble Sort performs poorly on large datasets because it makes multiple passes through the array, comparing and swapping elements until the list is sorted. This means that for each element, it potentially needs to traverse the entire array multiple times, leading to quadratic growth in execution time as shown in the test results (33,859ms for 100,000 elements). In contrast, Quick Sort and the STL Sort achieve much better performance with O(n log n) complexity by using divide-and-conquer strategies. These algorithms break down the problem into smaller sub-problems and sort them independently, resulting in a lot faster execution times (12-14ms for 100,000 elements).

## Ideal Use Cases

Each algorithm has its ideal use cases based on these characteristics. Bubble Sort, despite poor performance heree, can be useful for educational purposes or for very small datasets (less than 50 elements) due to its simplicity and minimal space requirements (O(1) extra space). It's also stable, meaning it preserves the relative order of equal elements. Quick Sort excels in general-purpose sorting scenarios with large datasets due to its excellent average-case performance and good cache utilization, though it's not stable and requires O(log n) extra space for recursion. The STL Sort (typically IntroSort) is the most practical choice for production code as it combines the benefits of multiple algorithms - using QuickSort's speed for large partitions, HeapSort's consistency to avoid QuickSort's worst-case scenarios, and InsertionSort's efficiency for small sub-arrays, making it efficient across many types of input data.

## Resources

* <https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/>