Bubble Sort Performance Analysis

In this project, I tested four versions of the bubble sort algorithm to see how fast they can sort arrays of different sizes and types. I worked with three kinds of arrays:

- **Random elements**: Numbers in no particular order.
- **Already sorted elements**: Numbers already arranged in ascending order.
- **Reverse-sorted elements**: Numbers arranged in descending order.

I tested the algorithms with arrays ranging from 5,000 to 60,000 elements and measured how long it took to sort them.I recorded my outputs and filled in the tables below.

Classic Bubble Sort

Number of Elements	Random Elements	Already Sorted	Reverse Sorted
5000	**0.226486**	**0.123599**	**0.255183**
10000	**0.905823**	**0.512495**	**1.024800**
20000	**3.598270**	**2.008290**	**4.142300**
30000	**8.223190**	**4.576100**	**9.441760**
40000	**14.890800**	**8.402280**	**18.086800**
50000	**29.065700**	**13.485200**	**26.792300**
60000	**34.200600**	**19.134600**	**39.491300**

Reduced Bubble Sort

Number of Elements	Random Elements	Already Sorted	Reverse Sorted
5000	**0.098231**	**0.099605**	**0.100824**
10000	**0.396547**	**0.397634**	**0.392999**
20000	**1.611050**	**1.586400**	**1.588650**
30000	**3.710500**	**3.904310**	**3.703710**
40000	**6.797520**	**7.255650**	**6.619870**
50000	**10.318500**	**10.484600**	**10.791900**

60000	15.193500	15.276800	14.998700

Bubble Sort With Flag

Number of Elements	Random Elements	Already Sorted	Reverse Sorted
5000	0.000048	0.000050	0.000051
10000	0.000111	0.000190	0.000186
20000	0.000222	0.000243	0.000246
30000	0.000317	0.000333	0.000654
40000	0.000461	0.000442	0.000429
50000	0.000539	0.000558	0.000543
60000	0.000686	0.000710	0.000716

Two-Way Bubble Sort

Number of Elements	Random Elements	Already Sorted	Reverse Sorted
5000	0.000037	0.000035	0.000037
10000	0.000082	0.000082	0.000077
20000	0.000181	0.000248	0.000179
30000	0.000233	0.000231	0.000245
40000	0.000340	0.000342	0.000376
50000	0.000414	0.000385	0.000388
60000	0.000491	0.001052	0.000462

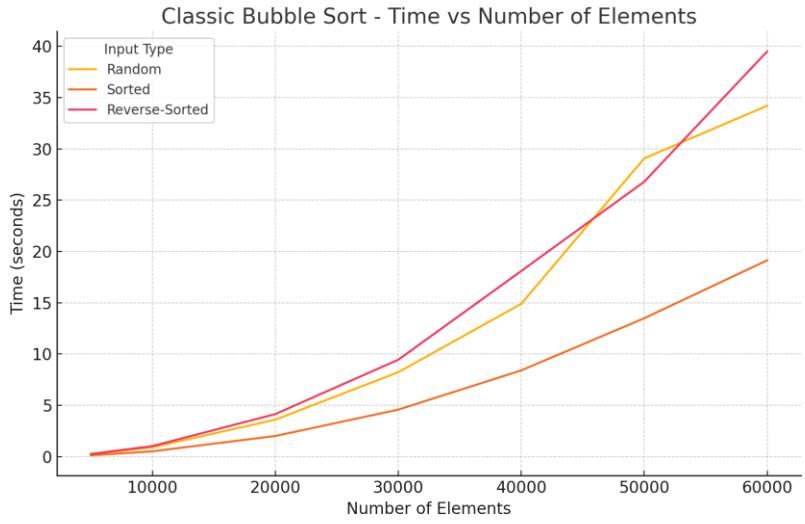
My Observations

1. **Classic Bubble Sort**:

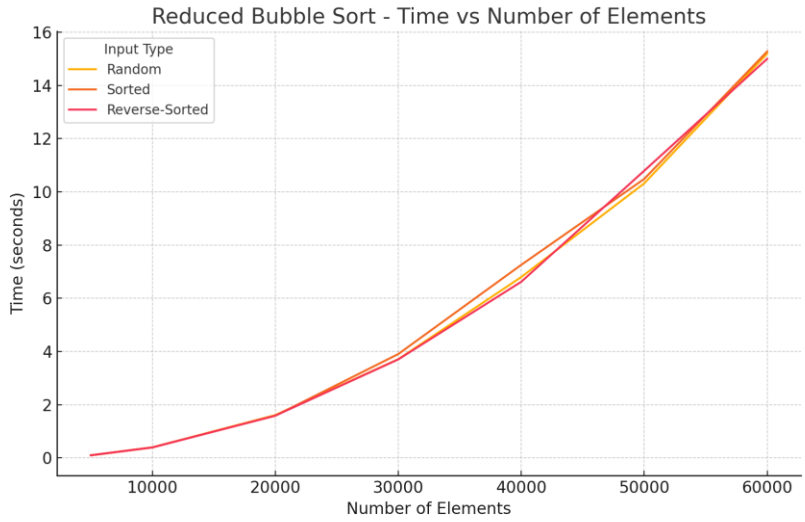
- This version is the slowest because it always compares and swaps elements, even when the array is already sorted.
- Sorting time increases significantly as the number of elements grows. It's very slow for large arrays.

2. **Reduced Bubble Sort**:
   - This version is slightly faster than the classic one because it stops checking the already sorted portion of the array.
   - The performance improved for all input types, but it's still not fast enough for large datasets.
3. **Bubble Sort with Flag**:
   - This version is much faster when the array is already sorted or becomes sorted early in the process. It stops immediately if no swaps are needed.
   - It performs the best with sorted arrays and is quick even for reverse-sorted ones.
4. **Two-Way Bubble Sort (Cocktail Sort)**:
   - This version checks the array in both directions (left to right and right to left), which helps balance the workload.
   - It performs well for all input types and is more efficient than the classic and reduced versions.

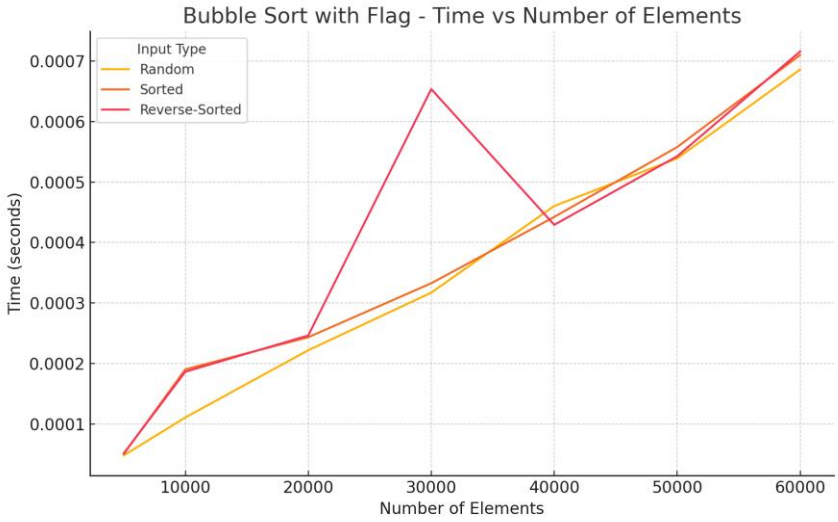
Classic Bubble Sort - Time vs Number of Elements



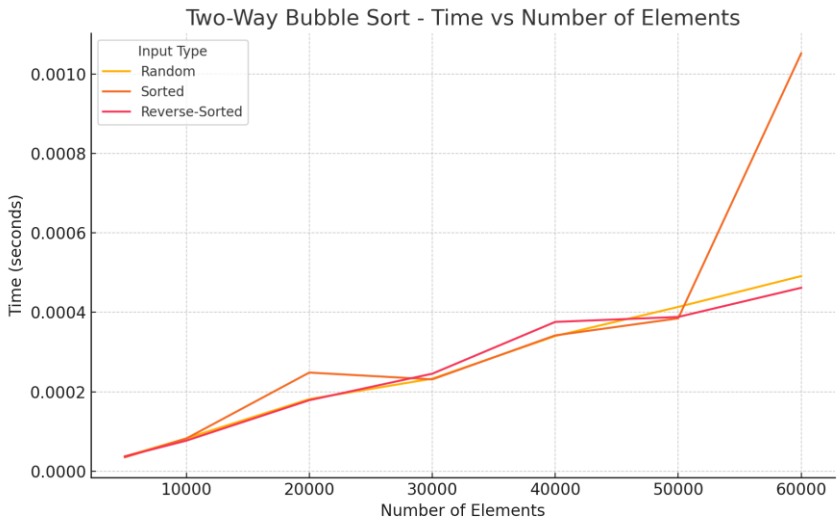
Reduced Bubble Sort - Time vs Number of Elements



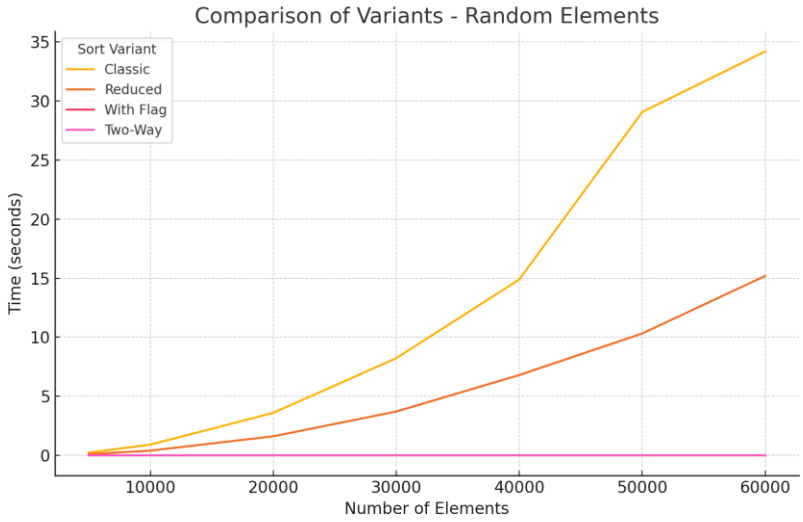
Bubble Sort with Flag - Time vs Number of Elements



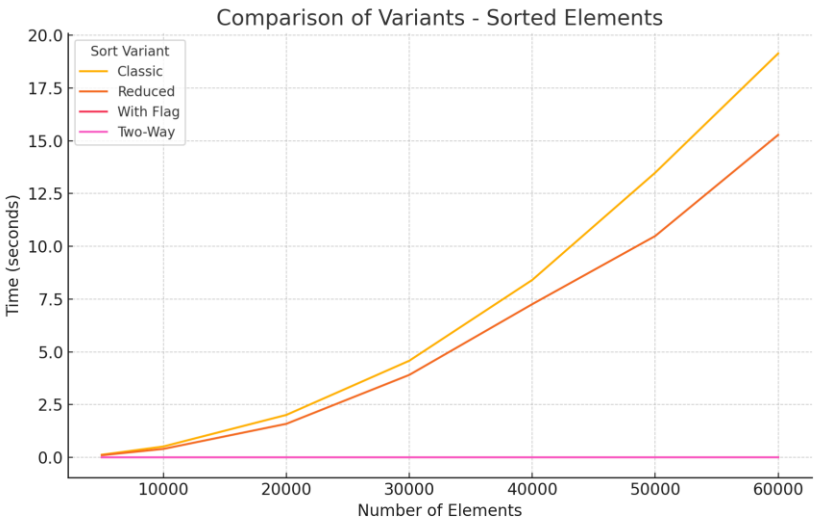
Two-Way Bubble Sort - Time vs Number of Elements



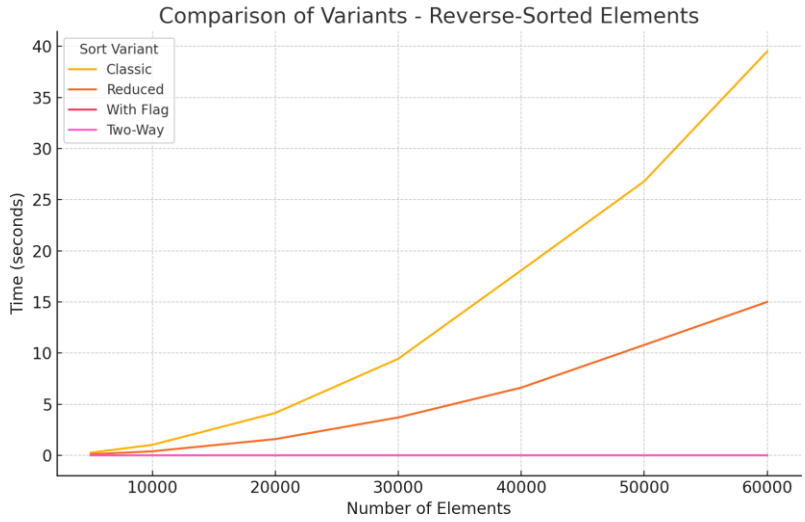
Comparison of Variants - Random Elements



Comparison of Variants - Sorted Elements



Comparison of Variants - Reverse-Sorted Elements



General Findings

- **Sorted arrays**: The "With Flag" version is the fastest because it can quickly determine that no swaps are needed.
- **Random and reverse-sorted arrays**: The two-way version performs consistently better than others because it processes the array in both directions.
- **Growth in sorting time**: Sorting time increases significantly with the number of elements, especially for the classic and reduced versions. The algorithms still

follow a quadratic time complexity, meaning their time grows very fast as the number of elements increases.

Conclusion

The "Bubble Sort with Flag" is the best option for small or already sorted datasets because of its ability to exit early. The "Two-Way Bubble Sort" is more balanced and works better for random or reverse-sorted data. However, all these algorithms are inefficient for large datasets, and better sorting algorithms like quicksort or mergesort should be used instead.

My output (just the first part)

```
Testing for array size: 5000
Bubble Sort Times:
          Random Elements : 0.226486 seconds
           Already Sorted : 0.123599 seconds
           Reverse Sorted : 0.255183 seconds

Bubble Sort Reduced Times:
          Random Elements : 0.0982306 seconds
           Already Sorted : 0.0996048 seconds
           Reverse Sorted : 0.100824 seconds

Bubble Sort With Flag Times:
          Random Elements : 4.8e-05 seconds
           Already Sorted : 5.03e-05 seconds
           Reverse Sorted : 5.13e-05 seconds

Bubble Sort Two Way Times:
          Random Elements : 3.71e-05 seconds
           Already Sorted : 3.47e-05 seconds
           Reverse Sorted : 3.71e-05 seconds

Testing for array size: 10000
Bubble Sort Times:
          Random Elements : 0.905823 seconds
           Already Sorted : 0.512495 seconds
           Reverse Sorted : 1.0248 seconds

Bubble Sort Reduced Times:
          Random Elements : 0.396547 seconds
           Already Sorted : 0.397634 seconds
           Reverse Sorted : 0.392999 seconds

Bubble Sort With Flag Times:
          Random Elements : 0.0001106 seconds
           Already Sorted : 0.0001902 seconds
           Reverse Sorted : 0.0001863 seconds

Bubble Sort Two Way Times:
          Random Elements : 8.24e-05 seconds
           Already Sorted : 8.21e-05 seconds
           Reverse Sorted : 7.67e-05 seconds

Testing for array size: 20000
Bubble Sort Times:
          Random Elements : 3.59827 seconds
           Already Sorted : 2.00829 seconds
           Reverse Sorted : 4.1423 seconds
```