Bubble Sort

Ascending Order => Swap elements a[j] and a[j+1] if a[j] > a[j+1]

Descending Order => Swap elements a[j] and a[j+1] if a[j] < a[j+1]

https://upload.wikimedia.org/wikipedia/commons/c/c8/Bubble-sort-example-300px.gif

https://www.youtube.com/watch?v=k4RRi_ntQc8

Eric Schmidt asked "the most efficient way to sort a million 32-bit integers"

Obama replied, "the bubble sort would be the wrong way to go."

Runtime Complexity of Bubble Sort

Worst Case

Input Array => 10, 9, 8, 7, 6, 5, 4, 3, 2 ,1

1st Iteration => 9 Comparisons (9, 8, 7, 6, 5, 4, 3, 2, 1, **10**)

2nd Iteration => 8 Comparisons (8, 7, 6, 5, 4, 3, 2, **9, 10**)

3rd Iteration => 7 Comparisons (7, 6, 5, 4, 3, 2, 1, **8, 9, 10**)

4th Iteration => 6 Comparisons (6, 5, 4, 3, 2, 1, **7, 8, 9, 10**)

5th Iteration => 5 Comparisons (5, 4, 3, 2, 1, **6, 7, 8, 9, 10**)

6th Iteration => 4 Comparisons (4, 3, 2, 1, **5, 6, 7, 8, 9, 10**)

7th Iteration => 3 Comparisons (3, 2, 1, **4, 5, 6, 7, 8, 9, 10**)

8th Iteration => 2 Comparisons (2, 1, **3, 4, 5, 6, 7, 8, 9, 10**)

9th Iteration => 1 Comparison (1**, 2, 3, 4, 5, 6, 7, 8, 9, 10**)

f(n) = (n-1) + (n-2) + (n-3) + … + 3 + 2 + 1 = n(n-1)/2

Worst Case Complexity = O(n^2)

Best Case

Input Array => 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

1st Iteration = 9 Comparisons (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) (no elements were swapped)

Since in one complete iteration, no elements were swapped it means the array is sorted and no further sorting is required.

Best Case Complexity = O(n)

Note that If we would have use used non-optimized Bubble Sort algorithm the best case complexity would have been O(n^2)

Average Case

There are n! permutations possible for array containing n elements.

[7, 8, 2, 4, 6, 1, 10, 3, 9, 5]

[8, 7, 2, 4, 6, 1, 10, 3, 9, 5]

[2, 8, 7, 4, 6, 1, 10, 3, 9, 5]

[4, 8, 2, 7, 6, 1, 10, 3, 9, 5]

…

10! = 3628800

Average Case Complexity = O(n^2)

Space Complexity = O(1)