Sorting Algorithms

Prepared for DSC 314 course at IISER-TVM by Dr. Dhanyamol Antony.

1. Selection Sort

- Selection sort is one of the easiest approaches to sorting.
- It is inspired by the way we sort things in day-to-day life.
- It is an in-place sorting algorithm because it uses no auxiliary data structures while sorting.

How Selection Sort Works

Consider the following elements to be sorted in ascending order using selection sort:

6, 2, 11, 7, 5

Selection sort works as follows:

- It finds the smallest element, which is 2.
- It swaps it with the first element of the unordered list.
- It finds the second smallest element, which is 5.
- It swaps it with the second element of the unordered list.
- Similarly, it continues this process until all elements are sorted.

As a result, the sorted elements in ascending order are:

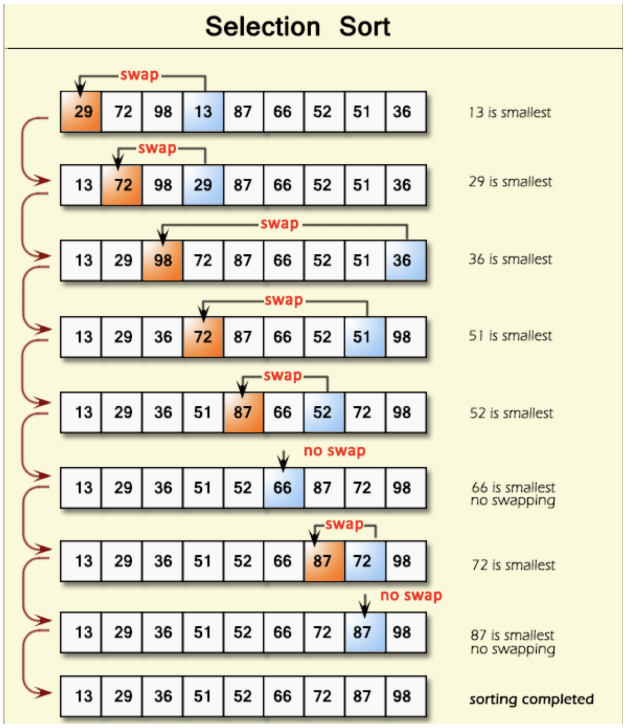
2, 5, 6, 7, 11

Selection Sort Algorithm

```
SelectionSort(A, n)

for i = 0 to n-2 do

    // assume the first element of the unsorted part is minimum
    minimum = i
```




```
    // find the minimum element in the remaining unsorted array
    for j = i + 1 to n-1 do
        if A[j] < A[minimum] then
            minimum = j
        end if
    end for

    // place the minimum element at its correct position
    swap(A[minimum], A[i])

end for
```

Time Complexity analysis

The number of comparisons in Selection Sort can be calculated as the sum of

$$(n-1) + (n-2) + (n-3) + \cdots + 1$$

$$= \frac{n(n-1)}{2}$$

Therefore, the time complexity of Selection Sort is:

$$O(n^2)$$

2. Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping adjacent elements if they are in the wrong order, causing larger elements to "bubble up" to their correct positions.

Pseudocode for Bubble Sort

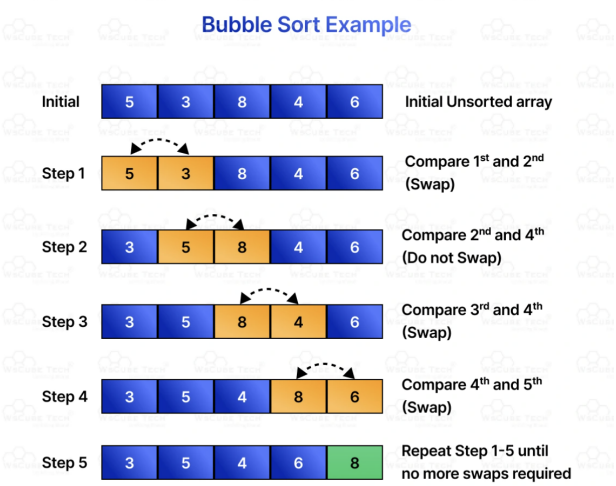
```
BubbleSort(A, n)
for i = 0 to n-2 do
    swapped = false
    for j = 0 to n-i-2 do
        if A[j] > A[j+1] then
            swap(A[j], A[j+1])
            swapped = true
        end if
    end for
    if swapped == false then
        break
    end if
end for
```

- Bubble Sort works by repeatedly comparing adjacent elements.
- If two adjacent elements are in the wrong order, they are swapped.
- After the first pass, the largest element moves to the last position.
- After the second pass, the second largest element moves to the second last position.
- After k passes, the largest k elements are placed in their correct positions.
- In the optimized version, a flag variable is used to check whether any swap occurs.
- If no swaps occur in a pass, the array is already sorted and the algorithm terminates early.

Time Complexity of Bubble Sort

In Bubble Sort, the number of comparisons in each pass decreases as:

$$(n-1) + (n-2) + (n-3) + \cdots + 1$$



$$= \frac{n(n-1)}{2}$$

Hence, the time complexity of Bubble Sort in the worst and average cases is:

$$O(n^2)$$

In the best case (already sorted array), the optimized bubble sort completes in :

$$O(n)$$