CSE 246: Algorithms

Insertion Sort, Bubble Sort, Selection Sort

Redwan Ahmed Rizvee

Insertion Sort: Main Intuition

- I want to sort i^{th} element, I will check with the sorted previous [1, $(i-1)^{th}$] elements (**Trying to insert the elements in sorted order**)
- I will try to move the element, as much as I can.
    - If I can, I will move forward
    - Otherwise stop, no need to

```
for( i = 0; i < N; i++) {
        for(j = i; j-1 >= 0; j=j-1) {
                If (array[j] < array[j-1]) {
                        swap(array[j], array[j-1])
                }
                Else break
        }
}
```

Insertion Sort: Example

5	8	6	1	7	9

i=1

5	8	6	1	7	9

i=2

5	8	6	1	7	9

5	6	8	1	7	9

Insertion Sort: Example

i=3

5	6	8	1	7	9

5	6	1	8	7	9

5	1	6	8	7	9

1	5	6	8	7	9

Insertion Sort: Example

i=4

1	5	6	8	7	9

1	5	6	7	8	9

Insertion Sort: Example

i=5

1	5	6	7	8	9

Insertion Sort: Complexity Analysis

- Memory Complexity, O(N)
- Worst Case Complexity, O(N^2): [9, 8, 7, 6, 5, 1] ~ 0+1+2+3+4+5 = (5^* 6)/2 ~ c * 6^2
- Best Case Complexity, O(N): Already sorted during insertion

Bubble Sort

Bubble sort: Main Intuition

- Concept: Heavier bubble stays below, lighter bubble stays above
- Will iterate n times
  - In each iteration, will sort each consecutive elements: (i, i+1) -> (i+1, i+2) -> (i+2, i+3) …..
  - So, after each iteration, one element will be sorted/reach its actual position
  - After n iteration, n elements will be sorted

```
for( i = 0; i < N; i++) {
        for(j = 0; j+1<N; j=j+1) {
                If (array[j+1] < array[j]) {
                        swap(array[j], array[j+1])
                }
        }
}
```

Bubble Sort: Example

5	8	6	1	7	9

i=0

5	**8**	6	1	7	9

5	**8**	**6**	1	7	9

5	6	**8**	**1**	7	9

5	6	1	**8**	**7**	9

5	6	1	7	**8**	**9**

Bubble Sort: Example

5	6	1	7	8	9

i=1

5	6	1	7	8	9

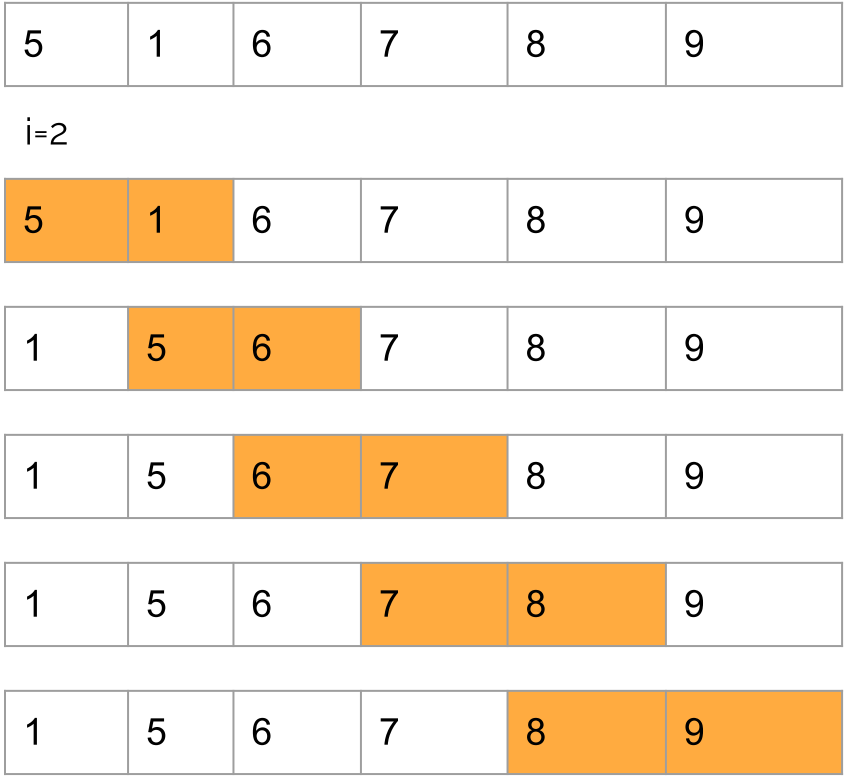
5	6	1	7	8	9

5	1	6	7	8	9

5	1	6	7	8	9

5	1	6	7	8	9

Bubble Sort: Example



5	1	6	7	8	9

i=2

5	1	6	7	8	9

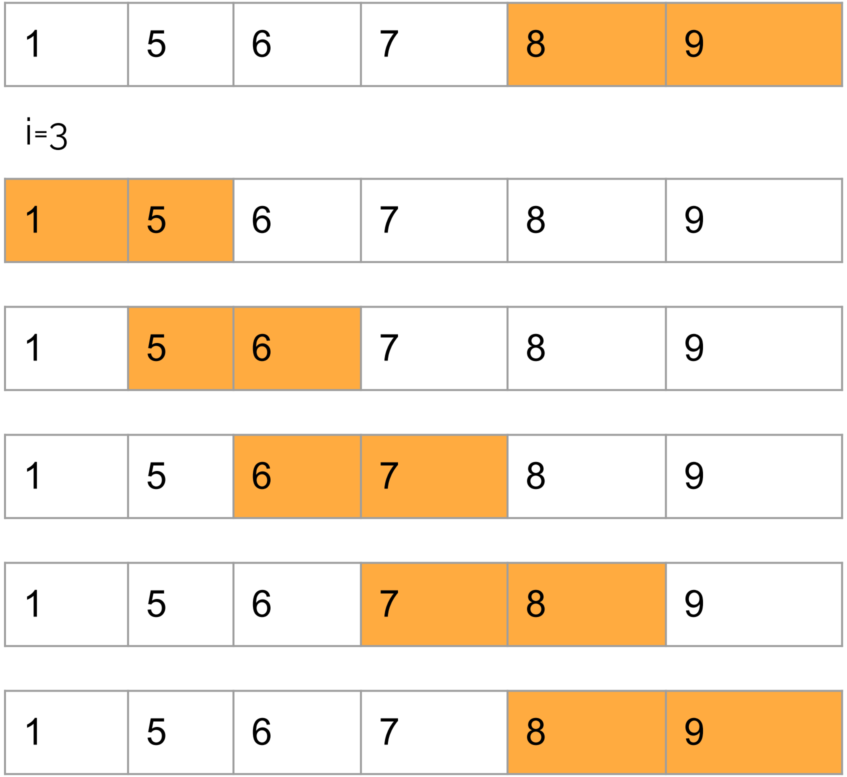
1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

Bubble Sort: Example



1	5	6	7	8	9

i=3

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

Bubble Sort: Example

1	5	6	7	8	9

i=4

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

1	5	6	7	8	9

Bubble Sort: Example

1	5	6	7	**8**	**9**

i=5

1	**5**	6	7	8	9

1	**5**	**6**	7	8	9

1	5	**6**	**7**	8	9

1	5	6	**7**	**8**	9

1	5	6	7	**8**	**9**

Bubble Sort: Optimizations

- Already sorted -> No shifting in some iteration
- If in each iteration, one element gets sorted, do we need to check all the pairs in each iteration.

Bubble Sort: Complexity Analysis

- Worst Case/ Big O Complexity: O(N^2) -> all the elements are reverse sorted
- Best Case Complexity: O(N) -> all are sorted, Using optimizations
- Memory Complexity, O(N)

Selection Sort

Selection Sort: Main Intuition

- In each iteration, we fix a position and then select the most suitable element for that position
- For i^{th} position,
    - Compare with all the values in [$(i+1)^{th}$, N^{th}] positions
    - Choose the best value suitable for that position [the smallest value]

```
for( i = 0; i < N; i++) {
        Best_idx = i
        for(j = i+1; j+1<N; j=j+1) {
                If (array[j] < array[Best_idx]) {
                        Best_idx = j
                }
        }
        swap(array[i], array[Best_idx])
}
```

Selection Sort: Example

5	8	6	1	7	9

i=0

5	8	6	1	7	9

1	8	6	5	7	9

Selection Sort: Example

1	8	6	5	7	9

i=1

1	8	6	5	7	9

1	8	6	5	7	9

1	5	6	8	7	9

Selection Sort: Example

1	5	6	8	7	9

i=2

1	5	6	8	7	9

Selection Sort: Example

1	5	6	8	7	9

i=3

1	5	6	8	7	9

1	5	6	7	8	9

Selection Sort: Example

1	5	6	7	8	9

i=4

1	5	6	7	8	9

Selection Sort: Complexity

- Worst Case Complexity: $O(N^2)$
- Best Case Complexity: $O(N^2)$
- Memory Complexity: O(N)
- Can we make optimizations ?
    - In i^{th} selection, no swapping occurred: No [1, 2, 3, 8, 7]