Insertion Sort

insertion sort →

arr → | 12 | 8 | 7 | 4 | 9 |

0 r 1 n 2 3 4

// Ascending order

```
for(int i=1; i<length; i++){
   int key = arr[i]; // Ascending order
   int j = i-1;
while(j>=0 && arr[j] > key){
        arr[j+1] = arr[j];
          j--;
   }
   arr[j+1] = key;
}
```

int i = 1 2

key = 8 7

j = 0 -1 x 0

j != 0

j	key			
12	8	7	4	9
12	12	7	4	9
8	12	7	4	9 (new arr)

and Pass (7)

8	12	7	4	9
8	12	12	4	9
7	8	12	4	9

3rd Pass key = 4 i = 4

7	8	12	4	9
7	8	12	12	9
7	8	8	12	9
7	7	8	12	9
4	7	8	12	9

j = 1 ; j--

4th Pass i = 5 key = 9 j = 4

4	7	8	12	9	8 > 9
4	7	8	12	12	arr[j+1] = key
4	7	8	9	12	

→ Sorted Array

Insertion Sort Descending order

```
for(int i=1; i<length; i++){
      int key = arr[i];
   int j = i-1;
while(j>=0 && arr[j] < key)
      arr[j+1] = arr[j];
        j--;
   }
   arr[j+1] = key;
}
```

Performance →

- Worst case time complexity = $O(n^2)$
- Best case time complexity = $O(n)$
- Average case time complexity = $O(n^2)$
- Worst case space complexity = $O(n^2)$ total, $O(1)$ auxiliary

Covered topic (29-09-2023)

1. Insertion Sort Ascending order
2. Insertion Sort Descending order

Tomorrow topic (30-09-2023)

1. Quick Sort