Insertion Sort

Tags	Algorithm Elementary-Sorting_Algorithms Sorting
Companies	
Created time	@November 3, 2022 1:39 PM
Last edited time	@November 5, 2022 6:00 AM
Status	Not started
URL	
URL 1	

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Characteristics of Insertion Sort:

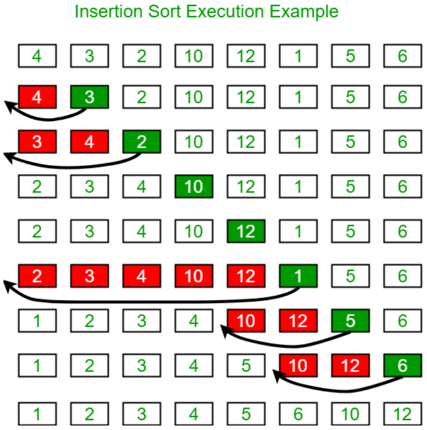
- This algorithm is one of the simplest algorithm with simple implementation
- Basically, Insertion sort is efficient for small data values
- Insertion sort is adaptive in nature, i.e. it is appropriate for data sets which are already partially sorted.

Working of Insertion Sort algorithm:

> Consider an example: arr[]: {12, 11, 13, 5, 6}
>
> **First Pass:**
>
> - Initially, the first two elements of the array are compared in insertion sort.
> - Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position. Thus, swap 11 and 12.
> - So, for now 11 is stored in a sorted sub-array.
>
> **Second Pass:**
>
> - Now, move to the next two elements and compare them
> - Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11
>
> **Third Pass:**
>
> - Now, two elements are present in the sorted sub-array which are **11** and **12**
> - Moving forward to the next two elements which are 13 and 5
> - Both 5 and 13 are not present at their correct place so swap them
> - After swapping, elements 12 and 5 are not sorted, thus swap again
> - Here, again 11 and 5 are not sorted, hence swap again
> - here, it is at its correct position
>
> **Fourth Pass:**

> - Now, the elements which are present in the sorted sub-array are **5, 11** and **12**
> - Moving to the next two elements 13 and 6
> - Clearly, they are not sorted, thus perform swap between both
> - Now, 6 is smaller than 12, hence, swap again
> - Here, also swapping makes 11 and 6 unsorted hence, swap again
> - Finally, the array is completely sorted.
>
> ### Illustrations:

Insertion Sort Algorithm



To sort an array of size N in ascending order:

- Iterate from arr[1] to arr[N] over the array.
- Compare the current element (key) to its predecessor.
- If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

```
void insertionSortII(vector<int> &arr)
{
  for (size_t i = 1; i < arr.size(); i++)
  {
    int current = i;
    int currentElement = arr[current];
    while (currentElement < arr[current - 1] && current > 0)
    {
      arr[current] = arr[current - 1];
      current--;
    }
    arr[current] = currentElement;
  }
}
```