

insertion sort

Builds up the final sorted list by transferring one element at a time.

Maintain a partially sorted array.

Keep elements in that partially sorted array sorted.

More efficient than selection sort.

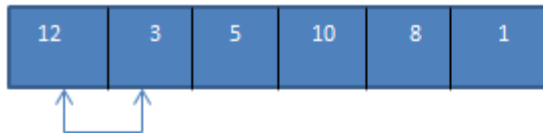
complex then selection sort.

The array to be sorted is as follows:

12	45	8	5	16
----	----	---	---	----

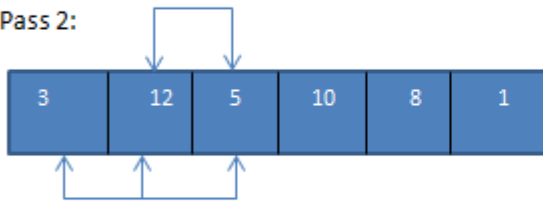
Now for each pass, we compare the current element to all its previous elements. Thus in the first pass, we start with the second element.

Pass 1:



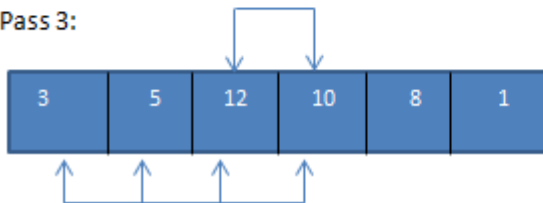
=>compare 2nd element to 1st element before it and sort

Pass 2:



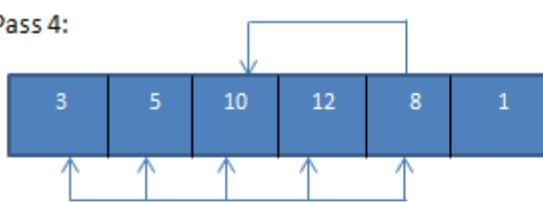
=>compare 3rd element to all elements before it and sort

Pass 3:



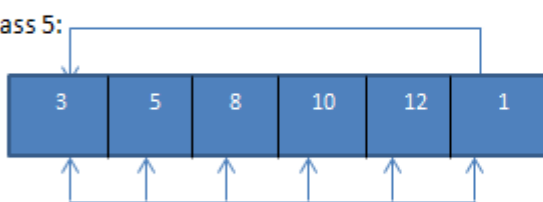
=>compare 4th element to all elements before it and sort

Pass 4:



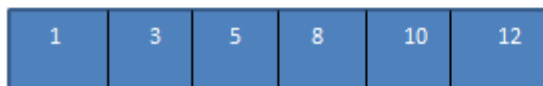
=>compare 5th element to all elements before it and sort

Pass 5:



=>compare 5th element to all elements before it and sort

Pass 6:



ted array

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

Time Complexity: $O(N^2)$

Auxiliary Space: $O(1)$

Already sorted array: $O(N)$

