Sri SAI RAM ENGINEERING COLLEGE

INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8391

DATA STRUCTURES (COMMON TO CSE & IT)

UNIT No. 5

SEARCHING,SORTING AND HASHING TECHNIQUES

5.3 Insertion Sort-Shell sort

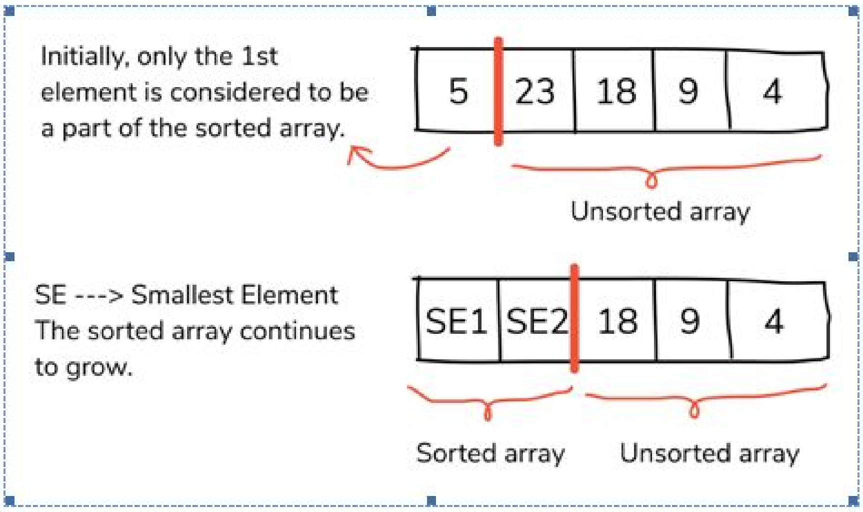
Version: 1.XX

Insertion Sort

What is Insertion Sort Algorithm?

Insertion sort is slightly different from the other sorting algorithms. It is based on the idea that each element in the array is consumed in each iteration to find its right position in the sorted array such that the entire array is sorted at the end of all the iterations.

In other words,it compares the current element with the elements on the left-hand side (sorted array).If the current element is greater than all the elements on its left hand side, then it leaves the element in its place and moves on to the next element. Else it finds its correct position and moves it to that position by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead.

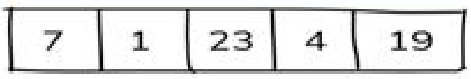

Initially, only the 1st element is considered to be a part of the sorted array.
5
23
18
9
4
Unsorted array
SE ---> Smallest Element
The sorted array continues to grow.
SE1
SE2
18
9
4
Sorted array
Unsorted array

Conclusion: Each iteration of insertion sort causes the sorted subset to grow, and the unsorted subset to shrink.

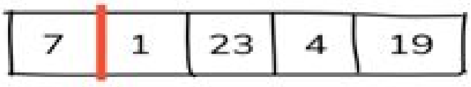
Insertion Sort Example & Working

Let us consider an array with 5 elements, A = [7, 1, 23, 4, 19]. Below is how insertion sort works for this array.

Given unsorted array

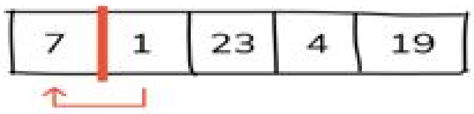


Consider the first element as sorted as there are no other elements on its left-hand side

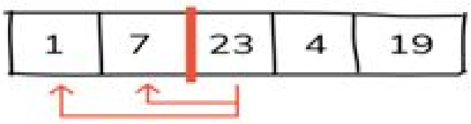


Look at the next unsorted element (7) & compare it with the sorted elements (1)

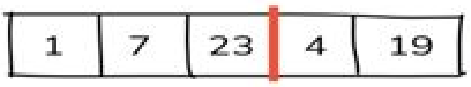
1 < 7 ?
Yes, so swap



23 < 7 ? ---> No
23 < 1 ? ---> No
So no swaps in this iteration

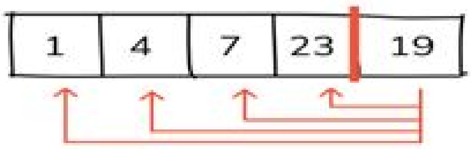


4 < 23 ? ---> Yes, so check the next number
4 < 7 ? ---> Yes, so check the next number
4 < 1 ? ---> No

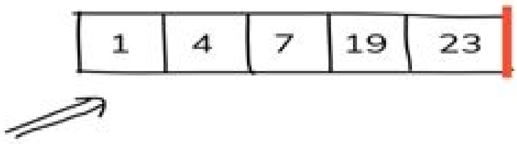


So the correct position of 4 is the current position of 7. **So shift all the elements from 7 to one position ahead.**

19 < 23 ? ---> Yes, so check the next number
19 < 7 ? ---> No



So the correct position of 19 is the current position of 23. **So shift the element 23 one position ahead.**



The entire array is now sorted.

Insertion Sort Pseudo Code

Here is the pseudocode implementation of Insertion sort.

```
for i = 1 to n
key = arr[i]
j = i - 1
// comparing whether the first element is greater than the second element
// if yes, then store the largest element to the next position
while j >= 0 and arr[j] > key
 arr[j + 1] = arr[j]
 j = j - 1
 end while
 // storing the smallest element in the correct position
 arr[j + 1] = key
end for
```

Program

```
#include<stdio.h>
int main()
{
int i, n, j, temp, min, key;
scanf("%d", &n);
int arr[n];
for(i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
for (i = 1; i < n; i++)
{
```

```
key = arr[i];
j = i - 1;
// comparing whether the first element is greater than the second element
// if yes, then store the largest element to the next position
while (j >= 0 && arr[j] > key)
{
arr[j + 1] = arr[j];
j = j - 1;
}.
// storing the smallest element in the correct position
arr[j + 1] = key;
}
for(i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
}
```

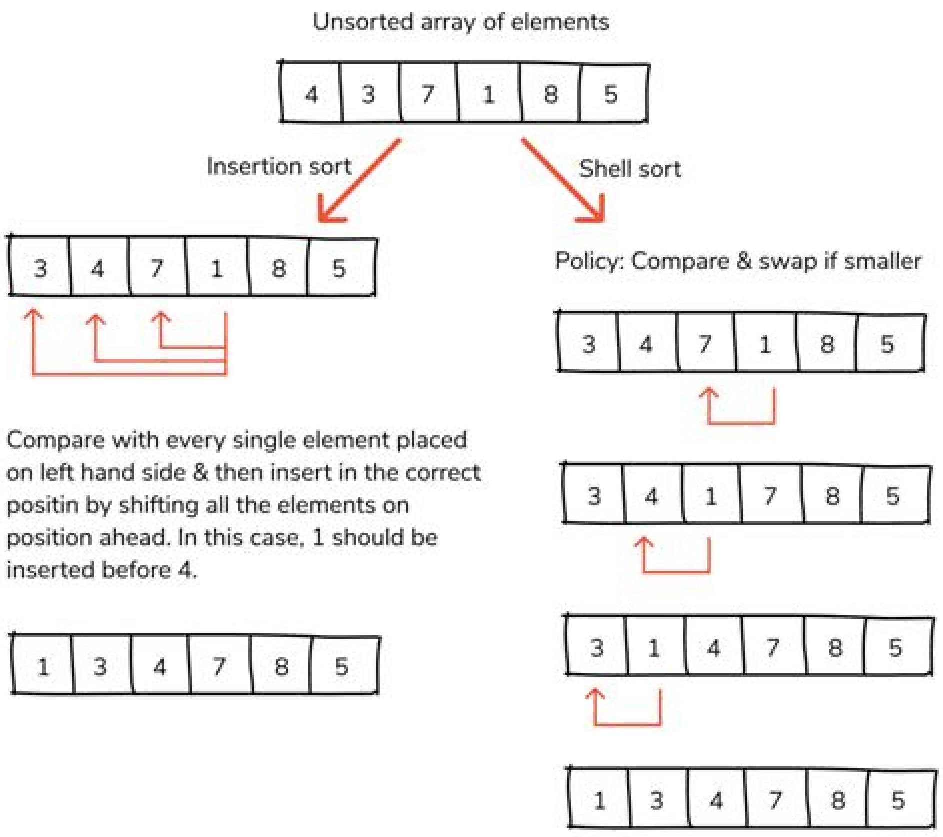
Shell Sort

What is Shell Sort Algorithm?

Shellsort, also known as Shell sort or Shell's method, is an in-place comparison sort. It can either be seen as a generalization of sorting by exchange (bubble sort) or sorting by insertion (insertion sort). Worst case time complexity is O(n2) and best case complexity is O(nlog(n)).

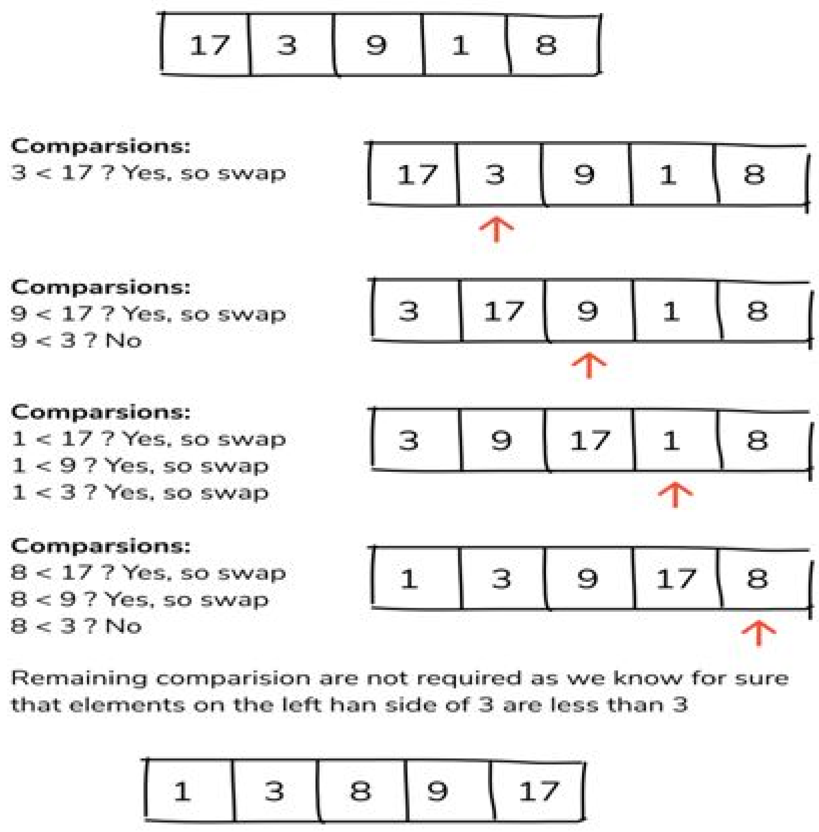
Shell sort algorithm is very similar to that of the Insertion sort algorithm. In case of Insertion sort, we move elements one position ahead to insert an element at its correct position. Whereas here, Shell sort starts by sorting pairs of elements far apart from each other, then progressively reducing the gap between elements to be compared. Starting with far apart elements, it can move some out-of-place elements into the position faster than a simple nearest-neighbor exchange.

A simple look at Shell Sort Vs Insertion Sort


Unsorted array of elements
4 3 7 1 8 5
Insertion sort
Shell sort
3 4 7 1 8 5
Policy: Compare & swap if smaller
3 4 7 1 8 5
Compare with every single element placed on left hand side & then insert in the correct positin by shifting all the elements on position ahead. In this case, 1 should be inserted before 4.
3 4 1 7 8 5
1 3 4 7 8 5
3 1 4 7 8 5
1 3 4 7 8 5

Shelling out the Shell Sort Algorithm with Examples

Here is an example to help you understand the working of Shell sort on array of elements name A = {17, 3, 9, 1, 8}



Program

```
#include<stdio.h>
int main()
{
int n, i, j, temp, gap;
scanf("%d", &n);
int arr[n];
for(i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
for (gap = n/2; gap > 0; gap = gap / 2)
{
 // Do a gapped insertion sort
// The first gap elements arr[0..gap-1] are already in gapped order
// keep adding one more element until the entire array is gap sorted
for (i = gap; i < n; i = i + 1)
{
 // add arr[i] to the elements that have been gap sorted
// save arr[i] in temp and make a empty space at index i
int temp = arr[i];
 // shift earlier gap-sorted elements up until the correct location for arr[i] is found
for (j = i; j >= gap && arr[j - gap] > temp; j = j - gap)
arr[j] = arr[j - gap];
 // put temp (the original arr[i]) in its correct position
arr[j] = temp;
}
}
for(i = 0; i < n; i++)
{
printf("%d ", arr[i]);
}
}
```