

# **Experiment-2.2**

## Aim of the Experiment:

Write a program to implement the Shell sort along with its complexity analysis.

## 1. Problem Description:

<u>Shell Sort:</u> Shell sort is an optimization of insertion sort that allows the exchange of items that are far apart. To move an element to a far-away position, many movements are required that increase the algorithm's execution time. But shell sort overcomes this drawback of insertion sort. It allows the movement and swapping of far-away elements as well. It is a comparison-based and in-place sorting algorithm. Shell sort is efficient for medium-sized data sets.

## 2. Algorithm:

```
ShellSort(a, n) // 'a' is the given array, 'n' is the size of array for (interval = n/2; interval > 0; interval /= 2) for ( i = interval; i < n; i += 1) temp = a[i]; for (j = i; j >= interval && a[j - interval] > temp; j -= interval) a[j] = a[j - interval]; a[j] = temp;
```

**End ShellSort** 



## 3. Complexity Analysis:

**Time Complexity:** Time complexity of the Shell sort when gap is reduced by half in every iteration is  $O(n^2)$ .

1) Best Case: O(nlog(n)). When the given array is already sorted.

2) <u>Worst Case:</u> O(n^2).

3) Average Case:  $O(n^{1.25})$ . It depends on the interval selected by the programmer.

**Space Complexity:** The space complexity of Shell Sort is O(1).

#### 4. Pseudo Code:

```
procedure shell sort(array, n)
 while gap < length(array) /3:
            gap = (interval * 3) + 1
 end while loop
  while gap > 0:
    for (outer = gap; outer < length(array); outer++):
        insertion value = array[outer]
            inner = outer;
        while inner > gap-1 and array[inner - gap] >= insertion value:
            array[inner] = array[inner - gap]
            inner = inner - gap
        end while loop
           array[inner] = insertion_value
    end for loop
    gap = (gap - 1)/3;
  end while loop
end shell sort
```



#### 5. Source Code for Experiment:

```
#include <iostream>
using namespace std;
void printArray(int arr[], int n) {
  for(int i=0; i<n; i++) {
     cout<<arr[i]<<" ";
  cout << endl;
}
void shellSort(int arr[], int n) {
  // Start with a big gap, then reduce the gap by half
  for(int gap=n/2; gap>0; gap=gap/2) {
     //Sort elements in each sublist
     for(int i= gap; i<n; i++) {
       int temp=arr[i];
       int j;
       //Swap elements in sublist if not in order
       for(j=i; j>=gap && arr[j-gap]>temp; j=j-gap) {
          arr[j]=arr[j-gap];
       arr[j]=temp;
     cout<<"Gap "<<gap<<" Sorting :";
     printArray(arr,n);
```

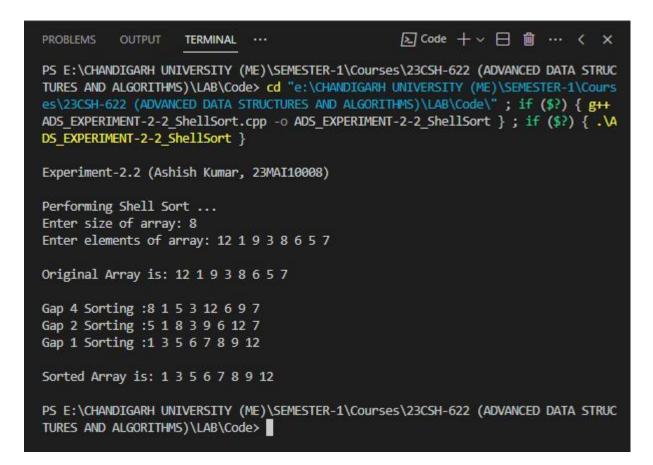


```
int main() {
  cout<<"\nExperiment-2.2 (Ashish Kumar, 23MAI10008)"<<endl<<endl;
  cout<<"Performing Shell Sort ..."<<endl;</pre>
  int n;
  int arr[100];
  cout<<"Enter size of array: ";</pre>
  cin>>n;
  cout<<"Enter elements of array: ";</pre>
  for(int i=0; i< n; i++){
     cin>>arr[i];
   }
  cout<<"\nOriginal Array is: ";</pre>
  printArray(arr,n);
  cout << endl;
  // Shell Sort Function
  shellSort(arr,n);
  cout<<"\nSorted Array is: ";</pre>
  printArray(arr,n);
  cout << endl;
  return 0;
```

}



#### 6. Result/Output:



#### Learning outcomes (What I have learnt):

- **1.** I learnt about how to input elements in an array.
- **2.** I learnt about how to perform shell sort in an array.
- **3.** I learnt about different applications of shell sort.
- **4.** I learnt about comparison between shell sort and insertion sort.
- **5.** I learnt about time and space complexity of shell sort.