**WORKSHEET 2.2**

**1. Aim:**

Write a program to implement shell sort and find their complexities.

**2. Problem Description:**

To implement shell sort and find their complexities

**3. 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

**4. Computational Complexity:-**

**Worst Case Complexity – O(n2)**  
The worst-case complexity for shell sort is  O(n2)

**Best Case Complexity - O(n log(n))**  
When the given array list is already sorted the total count of comparisons of each interval is equal to the size of the given array.  
So best case complexity is O(n log(n))

**Average Case Complexity – O(n\*log(n))**

It's somewhere around O. (n1.25).

The degree of complexity is determined by the interval picked. The above complexity varies depending on the increment sequences used. The best increment sequence has yet to be discovered.

**5. 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

**6. Source Code:**

#include <iostream>

using namespace std;

int shellSort(int arr[], int n)

{

for(int gap = n/2; gap > 0; gap /= 2)

{

for(int i = gap; i< n; i += 1)

{

int temp = arr[i];

int j;

for(j = i; j >= gap &&arr[j - gap] > temp; j -= gap)

arr[j] = arr[j - gap];

arr[j] = temp;

}

}

return 0;

}

void printArray(int arr[], int n)

{

for(int i=0; i<n; i++)

cout<<arr[i] << " ";

}

int main()

{

int arr[] = {12, 34, 54, 2, 3}, i;

int n = sizeof(arr)/sizeof(arr[0]);

cout<< "Array before sorting: \n";

printArray(arr, n);

shellSort(arr, n);

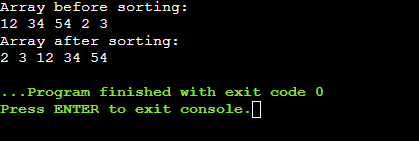
cout<< "\nArray after sorting: \n";

printArray(arr, n);

return 0;

}

1. **Screenshot of Output:**



1. **Learning & Outcomes:**

* Learned about the shell sort Algorithm, how it works, How much is the time complexity.
* Learned to create dynamic array using pointers.
* Learnt basic similarity and variations between shell sort and insertion sort.