geeksforgeeks.org

# ShellSort - GeeksforGeeks

4-5 minutes

ShellSort is mainly a variation of Insertion Sort. In insertion sort, we move elements only one position ahead. When an element has to be moved far ahead, many movements are involved. The idea of shellSort is to allow exchange of far items. In shellSort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1. An array is said to be h-sorted if all sublists of every h'th element is sorted.

Following is C++ implementation of ShellSort.

- C++
- Java
- Python

## C++

```
#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;

}
```

## Java

```
class ShellSort

{

    static void printArray(int arr[])

    {

        int n = arr.length;

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

            System.out.print(arr[i] + " ");

        System.out.println();

    }

    int sort(int arr[])

    {

        int n = arr.length;

        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;
    }
    public static void main(String args[])
    {
        int arr[] = {12, 34, 54, 2, 3};
        System.out.println("Array before
sorting");
        printArray(arr);
        ShellSort ob = new ShellSort();
        ob.sort(arr);
        System.out.println("Array after
sorting");
        printArray(arr);
    }
}
```

## Python

```
def shellSort(arr):
    n = len(arr)
    gap = n/2

    while gap > 0:

        for i in range(gap,n):

            temp = arr[i]

            j = i

            while j >= gap and arr[j-gap] >temp:

                arr[j] = arr[j-gap]

                j -= gap

            arr[j] = temp

        gap /= 2

arr = [ 12, 34, 54, 2, 3]

n = len(arr)

print ("Array before sorting:")

for i in range(n):

    print(arr[i]),

shellSort(arr)

print ("\nArray after sorting:")

for i in range(n):

    print(arr[i]),
```

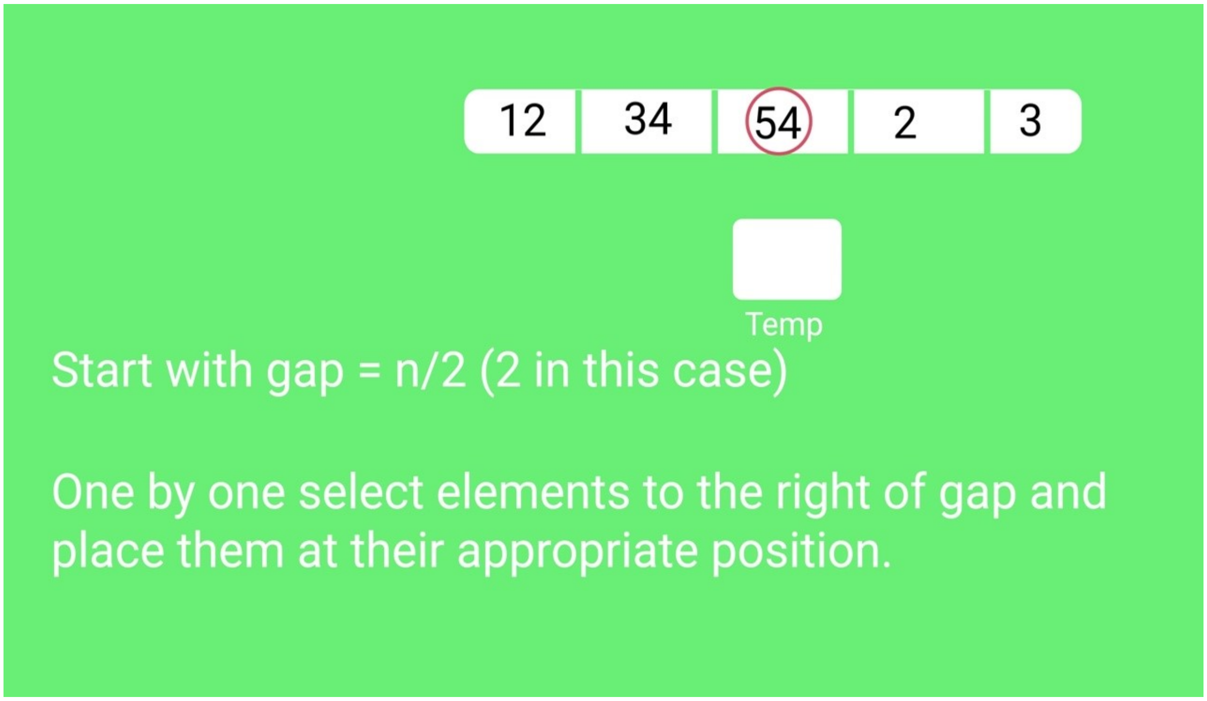
Output:

```
Array before sorting:
12 34 54 2 3
Array after sorting:
2 3 12 34 54
```

**Time Complexity:** Time complexity of above implementation of shellsort is O(n$^2$). In the above implementation gap is reduce by half in every iteration. There are many other ways to reduce gap which lead to better time complexity. See this for more details.

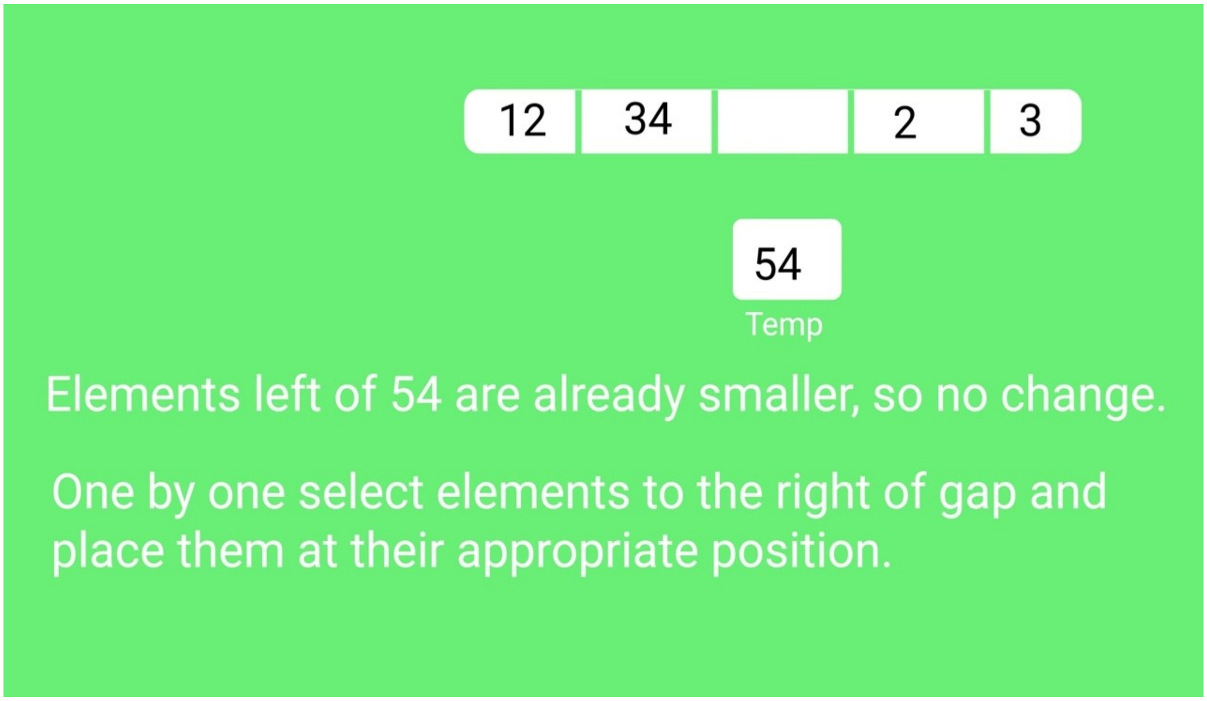
**References:**
https://www.youtube.com/watch?v=pGhazjsFW28
http://en.wikipedia.org/wiki/Shellsort

**Snapshots:**



Start with gap = n/2 (2 in this case)

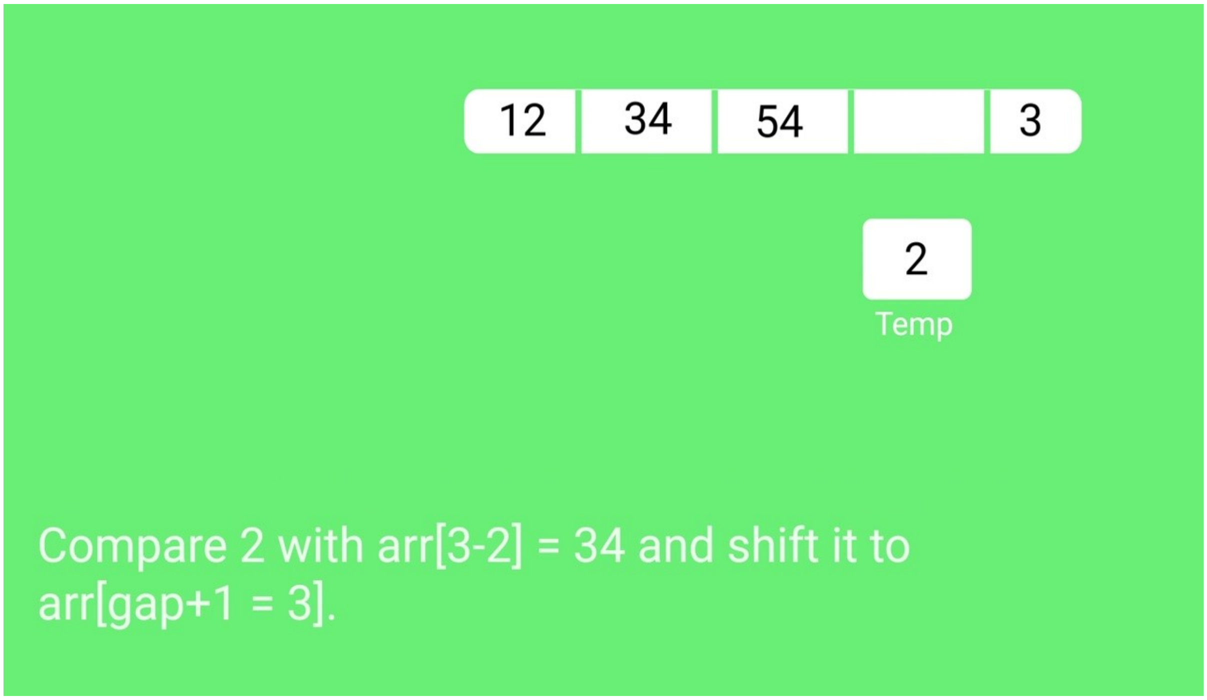
One by one select elements to the right of gap and place them at their appropriate position.



| | | | | |
|---|---|---|---|---|
| 12 | 34 | | 2 | 3 |

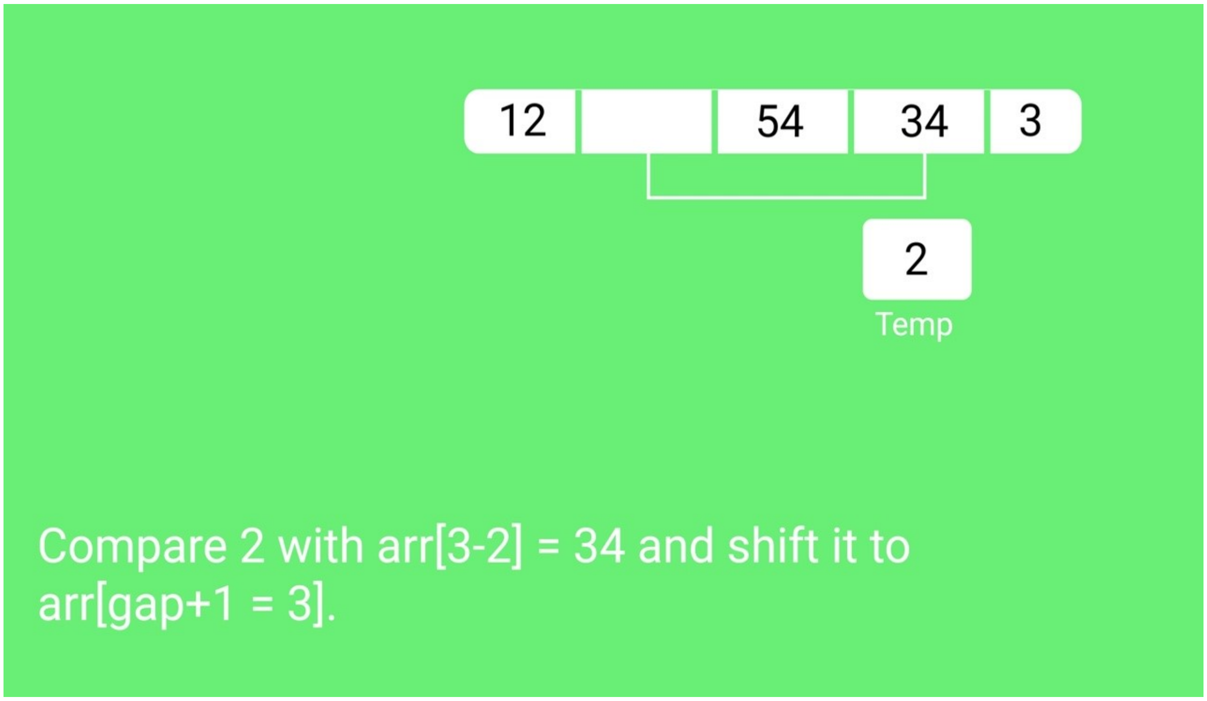
Elements left of 54 are already smaller, so no change.

One by one select elements to the right of gap and place them at their appropriate position.

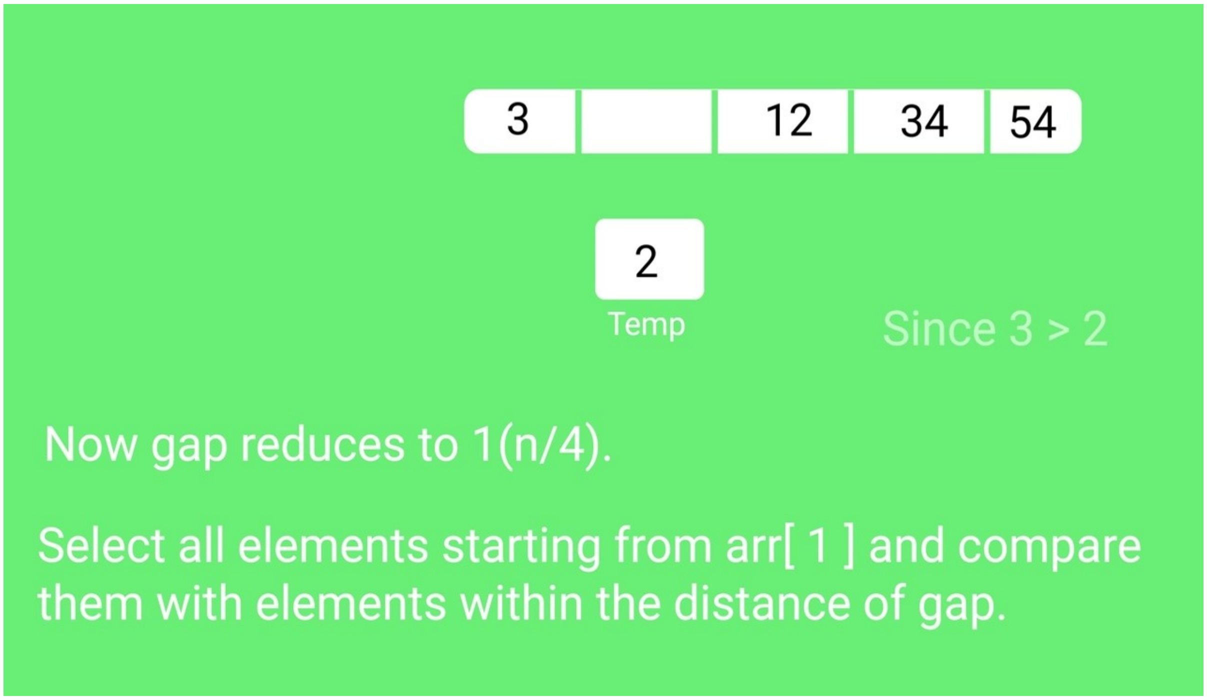


| | | | | |
|---|---|---|---|---|
| 12 | 34 | 54 | | 3 |

Compare 2 with arr[3-2] = 34 and shift it to arr[gap+1 = 3].

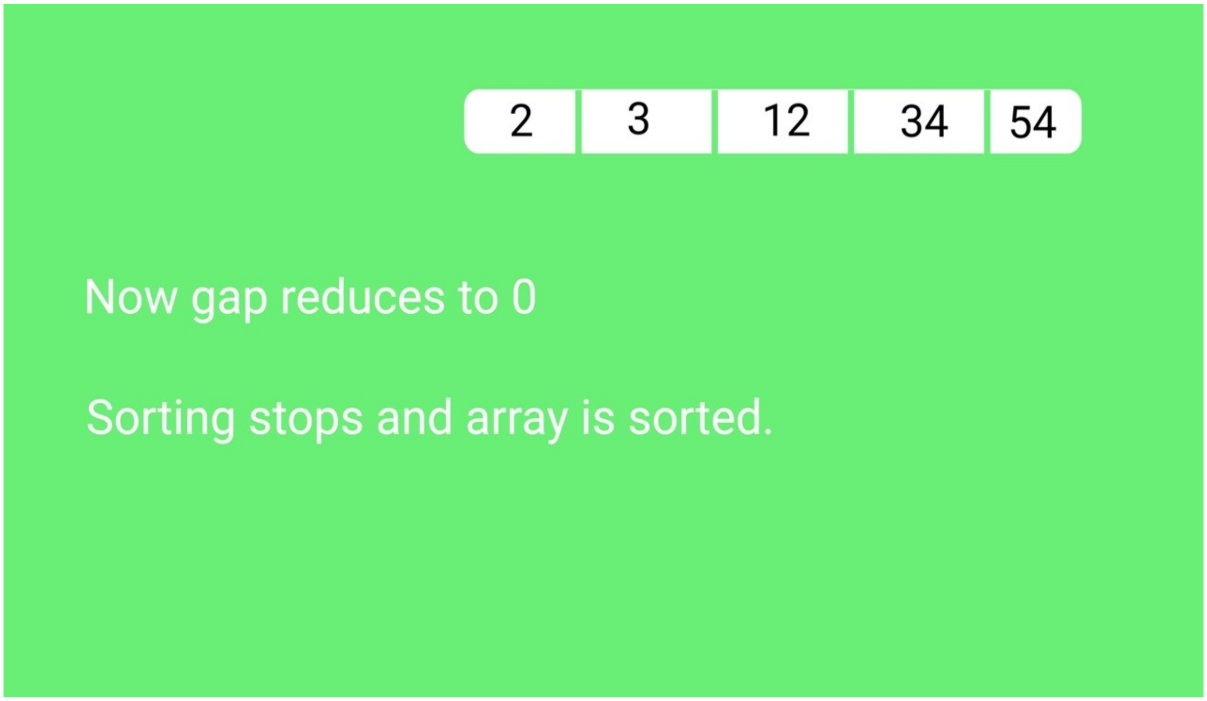


Compare 2 with arr[3-2] = 34 and shift it to arr[gap+1 = 3].



Now gap reduces to 1(n/4).

Select all elements starting from arr[ 1 ] and compare them with elements within the distance of gap.



| | | | | |
|---|---|---|---|---|
| 2 | 3 | 12 | 34 | 54 |

Now gap reduces to 0

Sorting stops and array is sorted.

## Quiz on Shell Sort

**Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz:**

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Heap Sort
- QuickSort
- Radix Sort
- Counting Sort
- Bucket Sort

## Coding practice for sorting.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above