GeeksforGeeks
A computer science portal for geeks

Google Custom Search

Practice GATE CS Placements Videos Contribute

Login/Register

ShellSort

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++

```
// C++ implementation of Shell Sort
#include  <iostream>
using namespace std;

/* function to sort arr using <a href="#">shellSort</a> */
int <a href="#">shellSort</a>(int arr[], int n)
{
    // Start with a big gap, then reduce the gap
    for (int gap = n/2; gap > 0; gap /= 2)
    {
        // Do a gapped insertion sort for this gap size.
        // The first gap elements a[0..gap-1] are already in gapped order
        // keep adding one more element until the entire array is
        // gap sorted
        for (int i = gap; i < n; i += 1)
        {
            // add a[i] to the elements that have been gap sorted
            // save a[i] in temp and make a hole at position i
            int temp = arr[i];

            // shift earlier gap-sorted elements up until the correct
            // location for a[i] is found
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                arr[j] = arr[j - gap];

            //  put temp (the original a[i]) in its correct location
            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);

    <a href="#">shellSort</a>(arr, n);

    cout << "\nArray after sorting: \n";
    printArray(arr, n);

    return 0;
}
```

Run on IDE

```
// Java implementation of <a href="#">ShellSort</a>
class <a href="#">ShellSort</a>
{
    /* An utility function to print array of size n*/
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    /* function to sort arr using <a href="#">shellSort</a> */
    int sort(int arr[])
    {
        int n = arr.length;

        // Start with a big gap, then reduce the gap
        for (int gap = n/2; gap > 0; gap /= 2)
        {
            // Do a gapped insertion sort for this gap size.
            // The first gap elements a[0..gap-1] are already
            // in gapped order keep adding one more element
            // until the entire array is gap sorted
            for (int i = gap; i < n; i += 1)
            {
                // add a[i] to the elements that have been gap
                // sorted save a[i] in temp and make a hole at
                // position i
                int temp = arr[i];

                // shift earlier gap-sorted elements up until
                // the correct location for a[i] is found
                int j;
                for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                    arr[j] = arr[j - gap];

                // put temp (the original a[i]) in its correct
                // location
                arr[j] = temp;
            }
        }
        return 0;
    }

    // Driver method
    public static void main(String args[])
    {
        int arr[] = {12, 34, 54, 2, 3};
        System.out.println("Array before sorting");
        printArray(arr);

        <a href="#">ShellSort</a> ob = new <a href="#">ShellSort</a>();
        ob.sort(arr);

        System.out.println("Array after sorting");
        printArray(arr);
    }
}
/*This code is contributed by Rajat Mishra */
```

Run on IDE

Python

```
# Python program for implementation of Shell Sort

def <a href="#">shellSort</a>(arr):

    # Start with a big gap, then reduce the gap
    n = len(arr)
    gap = n/2

    # Do a gapped insertion sort for this gap size.
    # The first gap elements a[0..gap-1] are already in gapped
    # order keep adding one more element until the entire array
    # is gap sorted
    while gap > 0:

        for i in range(gap,n):

            # add a[i] to the elements that have been gap sorted
            # save a[i] in temp and make a hole at position i
            temp = arr[i]

            # shift earlier gap-sorted elements up until the correct
            # location for a[i] is found
            j = i
            while  j >= gap and arr[j-gap] >temp:
                arr[j] = arr[j-gap]
                j -= gap

            # put temp (the original a[i]) in its correct location
            arr[j] = temp
        gap /= 2


# Driver code to test above
arr = [ 12, 34, 54, 2, 3]

n = len(arr)
print ("Array before sorting:")
for i in range(n):
    print(arr[i]),

<a href="#">shellSort</a>(arr)

print ("\nArray after sorting:")
for i in range(n):
    print(arr[i]),

# This code is contributed by Mohit Kumra
```

Run on ▲

Output:

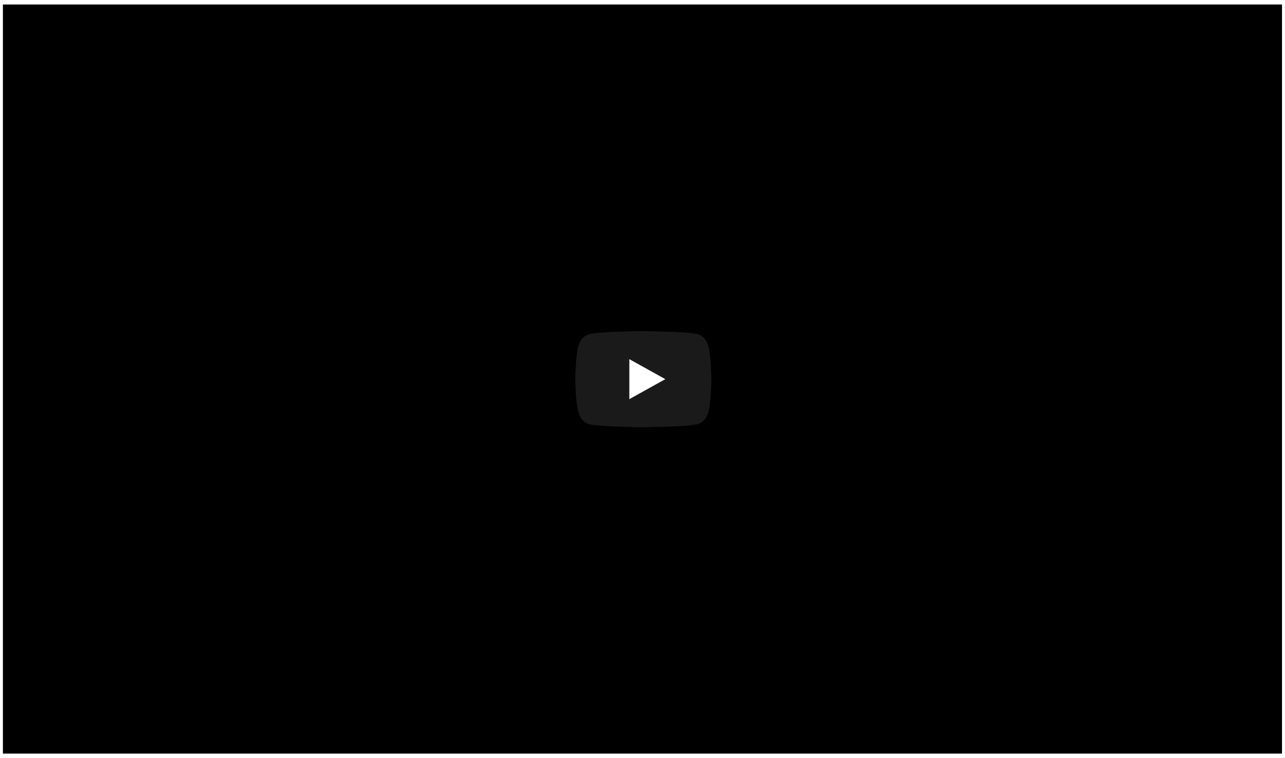
```
Array before sorting:
12 34 54 2 3
Array after sorting:
2 3 12 34 54
```

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.

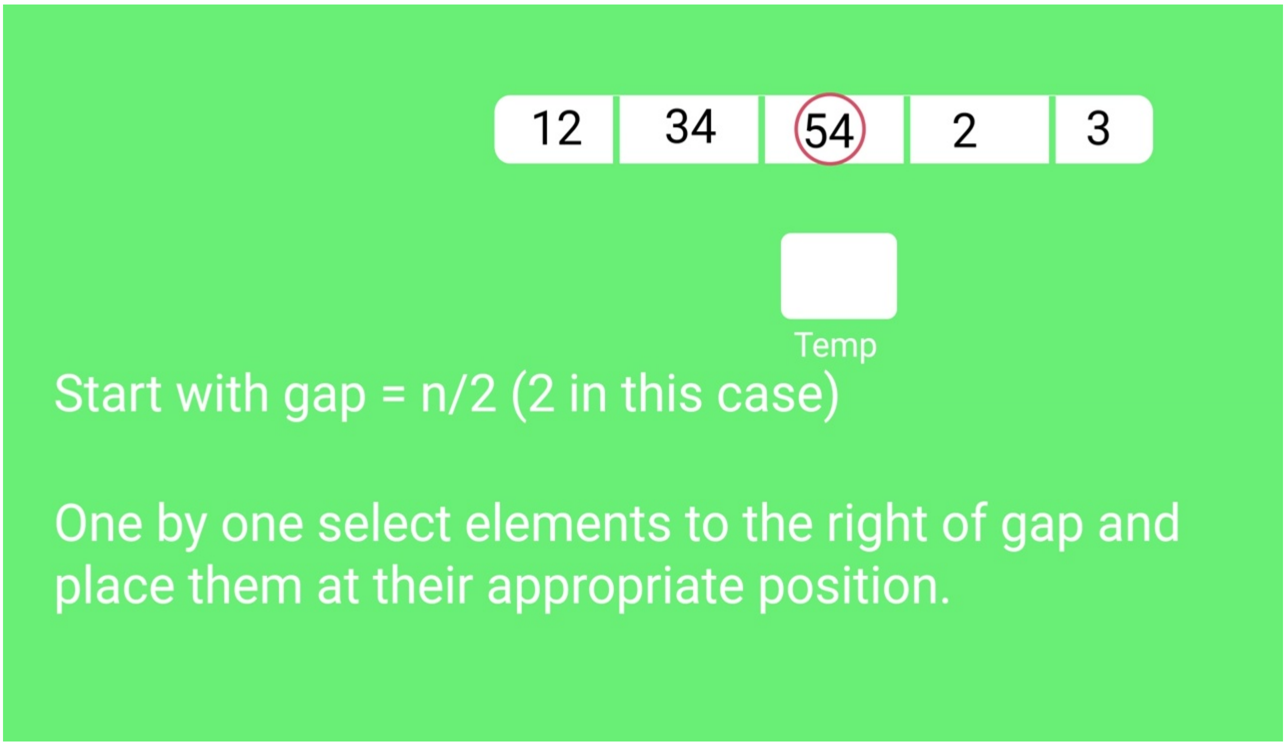
References:

https://www.youtube.com/watch?v=pGhazjsFW28

http://en.wikipedia.org/wiki/Shellsort

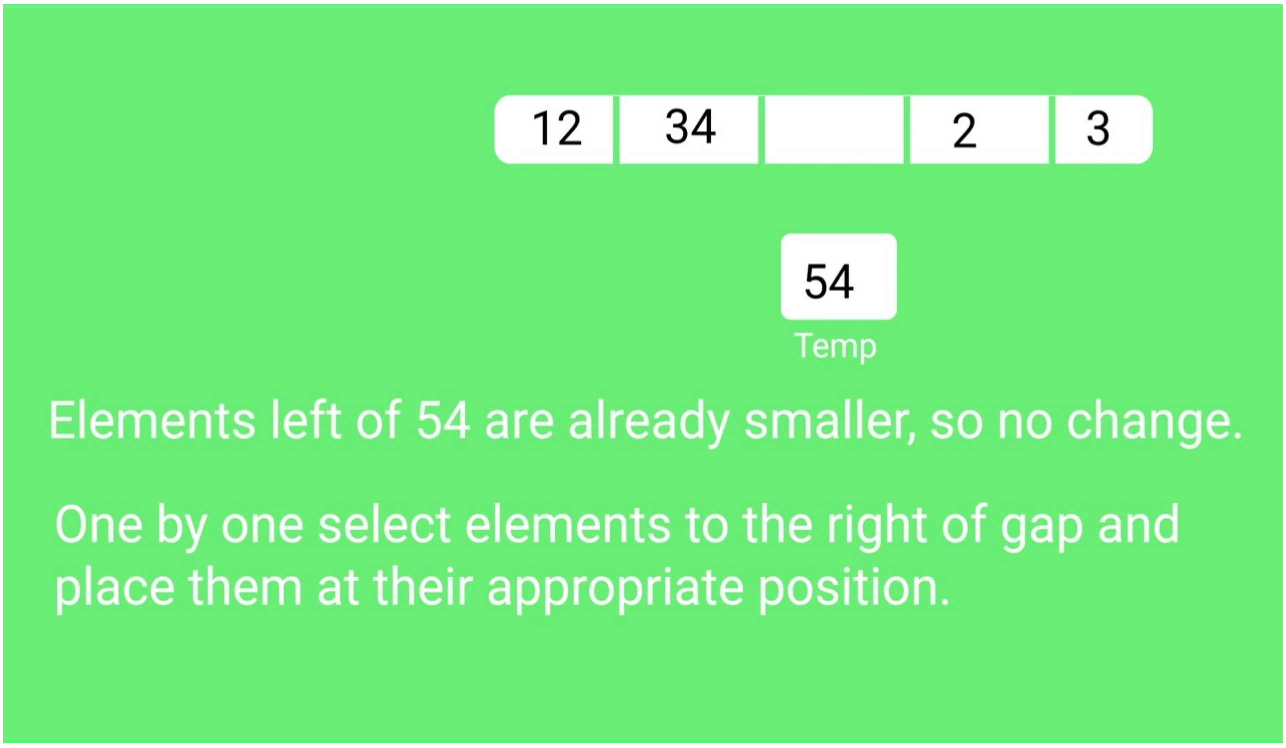


Snapshots:



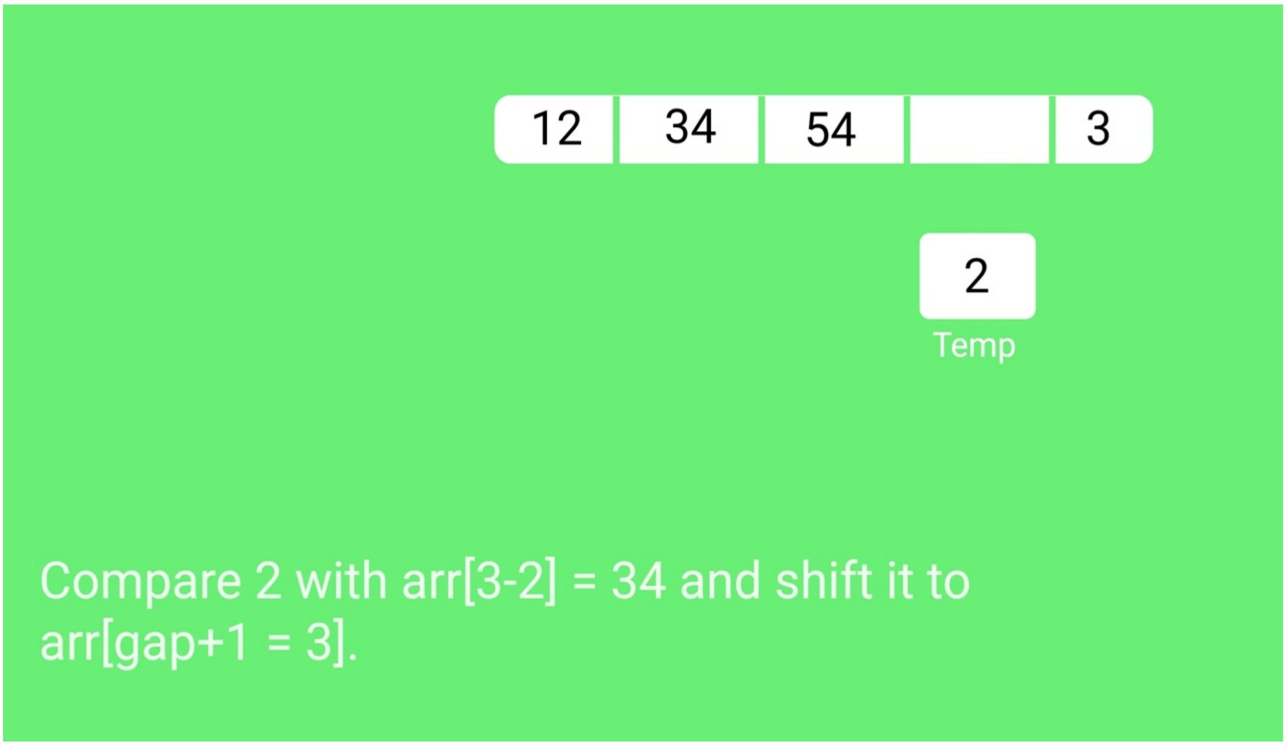
Start with gap = n/2 (2 in this case)

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

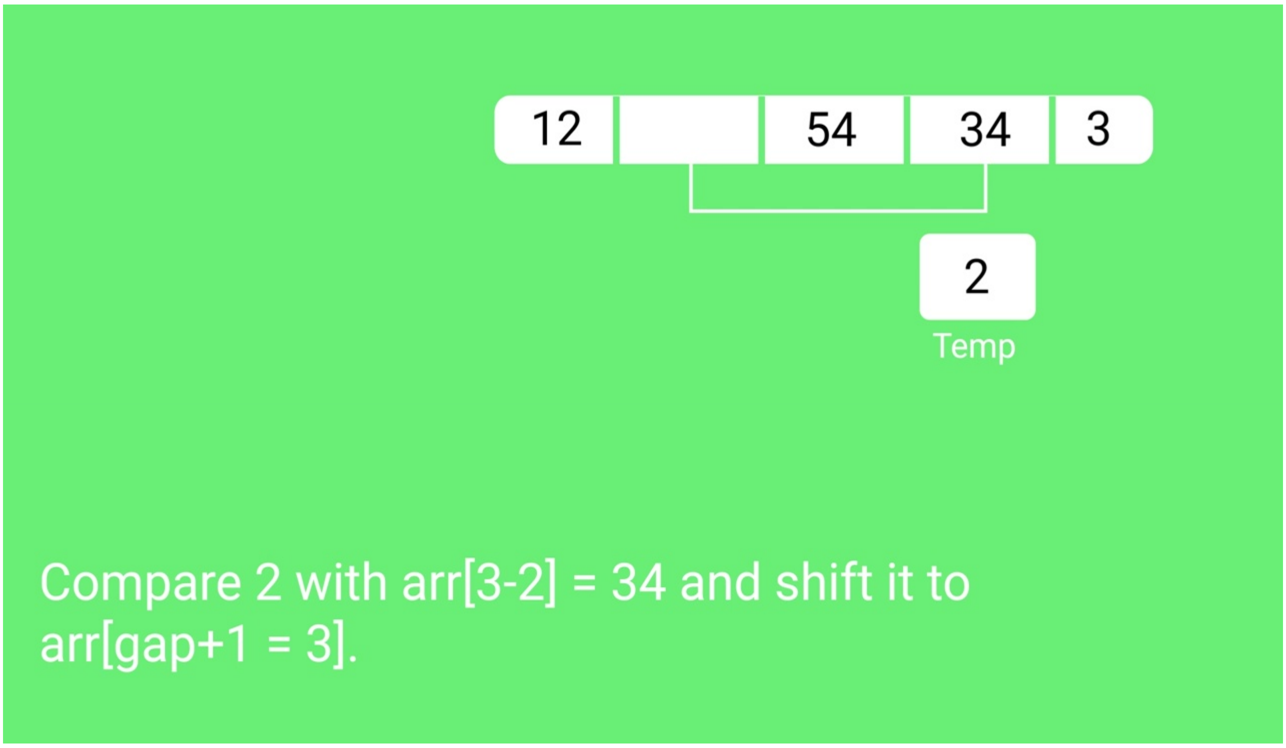


Elements left of 54 are already smaller, so no change.

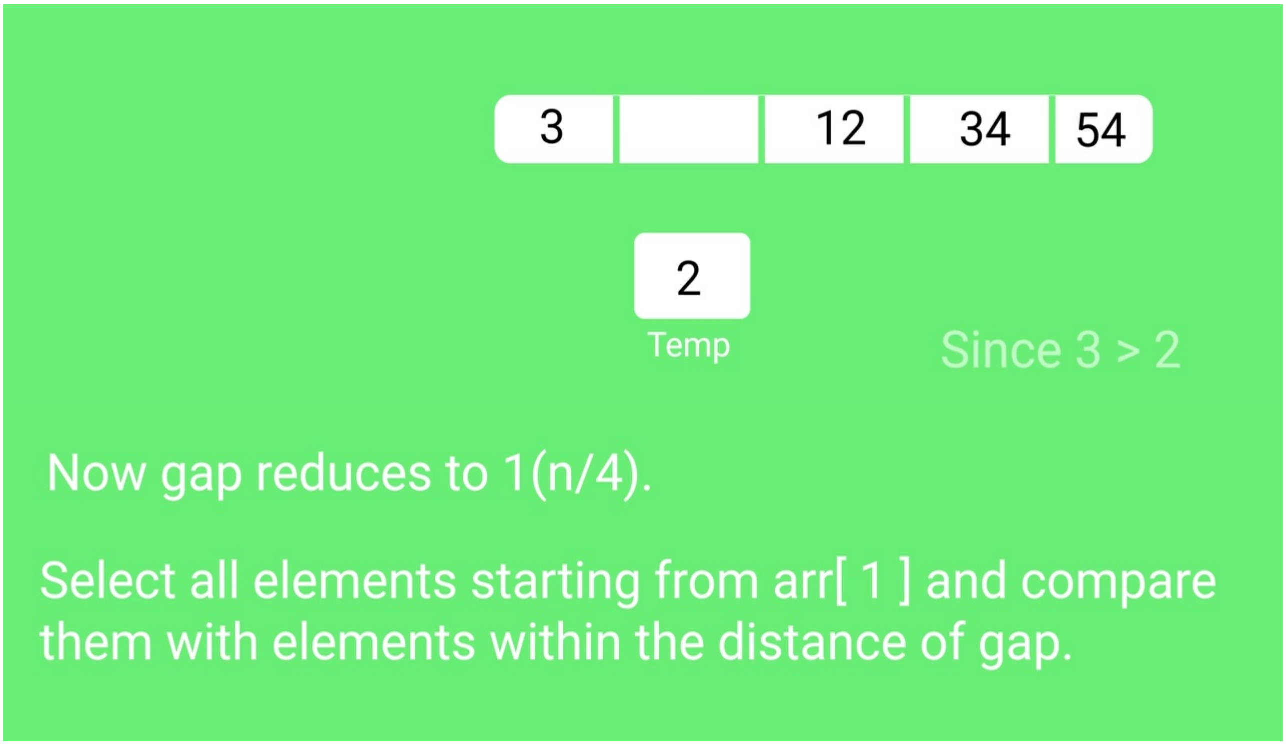
One by one select elements to the right of gap and place them at their appropriate position.



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

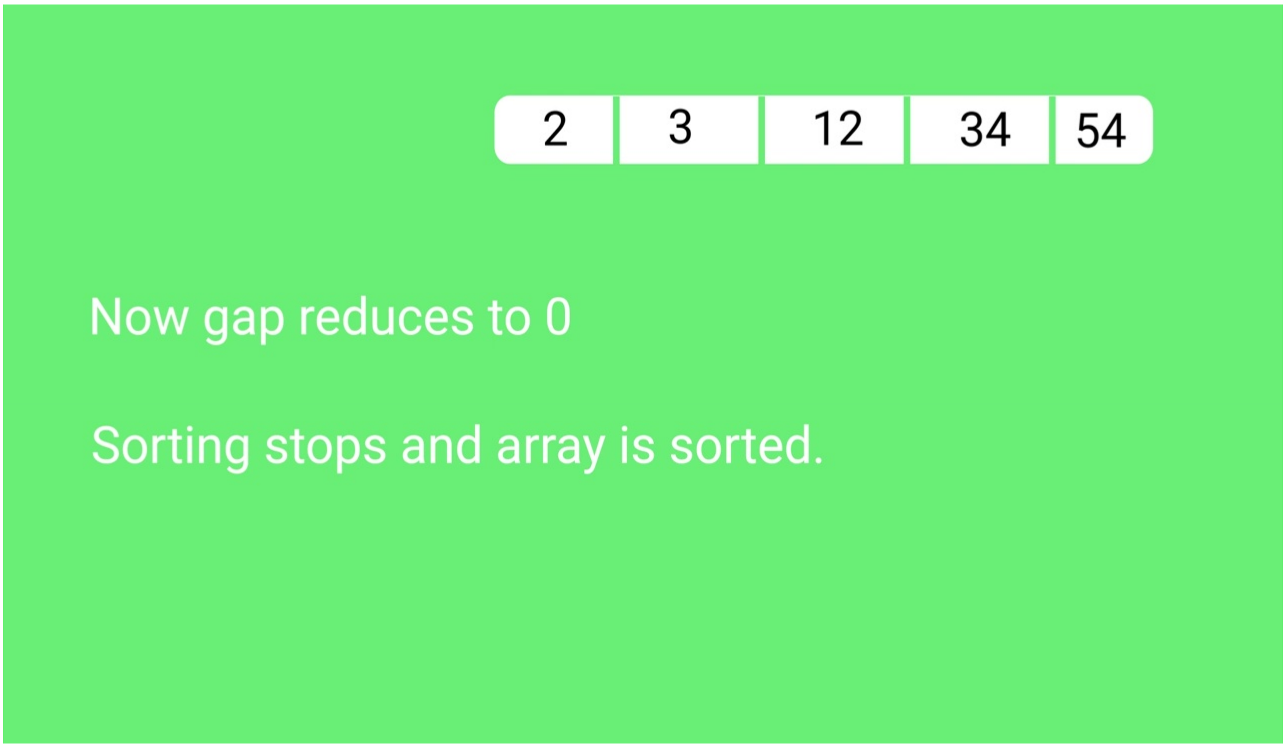


Compare 2 with arr[3-2] = 34 and shift it to arr[gap+1 = 3].



Now gap reduces to 1(n/4).

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



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

GATE CS Corner   Company Wise Coding Practice

Sorting   Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Comb Sort
Bucket Sort
Insertion Sort
Radix Sort
Merge Sort
Minimum number of subsets with distinct elements
Sorting array of strings (or words) using Trie | Set-2 (Handling Duplicates)
Python | Sort Tuples in Increasing Order by any key
Sum of Manhattan distances between all pairs of points
Maximum triplet sum in array

(Login to Rate)

3 Average Difficulty : **3/5.0**
Based on **11** vote(s)

Basic Easy Medium Hard Expert

🟩 Add to TODO List
🟩 Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!

▲

4 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Rajat Agrawal • 2 months ago

error : snapshots are repeated and one is missing

Reply • Share ›

roottraveller • 5 months ago

seriously?? do we need this?

Reply • Share ›

Harsh Sharma • 7 months ago

" arr (j - gap) = temp " in place of arr (j) = temp. As we need to swap the smaller arr (j) at the (j - gap) index.

Reply • Share ›

Venkatesh Ellaboina → Harsh Sharma • 5 months ago

No!! let j is at 4 and j-gap at 2 and the while condition fails.so where the temp variable should be inserted. Obviously it is j(4)

Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy DISQUS