Shell Sort

Shell Sort is an in-place comparison-based sorting algorithm. It is a generalization of insertion sort that allows the exchange of items that are far apart. The idea is to arrange the list of elements so that, starting anywhere, taking every (h)-th element produces a sorted list. Such a list is said to be (h)-sorted. The algorithm uses a sequence of gaps (or intervals) to determine which elements to compare and move.

How Shell Sort Works:

1. **Initialize the gap**: Start with a large gap and reduce it gradually.
2. **Sort sublists**: For each gap, sort the sublists created by the elements at that gap distance apart using insertion sort.
3. **Reduce the gap**: Continue the process until the gap is reduced to 1, at which point the list should be sorted.

Example Code in C:

```c
#include <stdio.h>

void shellSort(int arr[], int n) {
    // Start with a big gap, then reduce the gap
    for (int gap = n/2; gap > 0; gap /= 2) {
        // Perform a gapped insertion sort for this gap size
        for (int i = gap; i < n; i++) {
            // Save arr[i] in temp and make a hole at position i
            int temp = arr[i];
            int j;
            // 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 -= gap) {
                arr[j] = arr[j - gap];
            }
            // Put temp (the original arr[i]) in its correct location
            arr[j] = temp;
        }
    }
}

int main() {
    int arr[] = {12, 34, 54, 2, 3};
    int n = sizeof(arr)/sizeof(arr[0]);

    shellSort(arr, n);

    printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Explanation:

- **Initialization**: The gap is initialized to half the size of the array and is reduced by half in each iteration.
- **Gapped Insertion Sort**: For each gap, the elements are sorted using insertion sort. Elements that are `gap` distance apart are compared and swapped if necessary.
- **Reduction of Gap**: The process continues until the gap is reduced to 1, ensuring that the array is fully sorted.