INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHOPAL

Name: Tanav Singh Bajaj

Scholar No. : 20U03037

Semester: Third

Branch: IT

Algorithm

1. If there is only one element in the list it is already sorted, return.
2. Divide the list recursively into two halves until it can no longer be divided.
3. Merge the smaller lists into new lists in sorted order.

Merging Process :

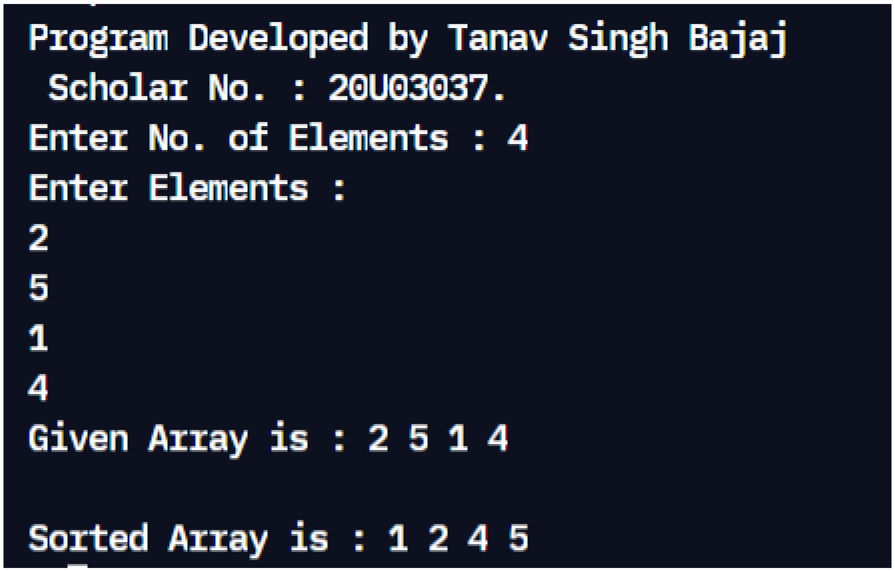
1. Divide the unsorted array into subarray, each containing a single element.
2. Take adjacent pairs of two single-element arrays and merge them to form an array of 2 elements.
3. Repeat the process till a single sorted array is obtained.

Implementation of Merge Sort

```
#include <iostream>
using namespace std;

void printArray(int *Arr, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << Arr[i] << " ";
    }
}
```

```
void merge(int A[], int mid, int low, int high)
{
    int i, j, k;
    int B[high + 1];
    i = low;
    j = mid + 1;
    k = low;

    while (i <= mid && j <= high)
    {
        if (A[i] < A[j])
        {
            B[k] = A[i];
            i++;
            k++;
        }
        else
        {
            B[k] = A[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        B[k] = A[i];
        k++;
        i++;
    }
    while (j <= high)
    {
        B[k] = A[j];
        k++;
        j++;
    }
    for (int i = low; i <= high; i++)
    {
        A[i] = B[i];
    }
}

void MS(int A[], int low, int high)
{
    int mid;
    if (low < high)
    {
        int mid = (low + high) / 2;
        MS(A, low, mid);
        MS(A, mid + 1, high);
        merge(A, mid, low, high);
    }
}

int main()
```

```
{
    cout << "Program Developed by Tanav Singh Bajaj /n Scholar No. : 20U03037.\n";
    int n;
    cout << "Enter No. of Elements : ";
    cin >> n;
    int A[n];
    cout << "Enter Elements : " << endl;
    for (int i = 0; i < n; i++)
    {
        cin >> A[i];
    }

    cout << "Given Array is : ";
    printArray(A, n);
    cout << endl;

    MS(A, 0, n - 1);
    cout << endl;
    cout << "Sorted Array is : ";

    printArray(A, n);
    cout << endl;
    return 0;
}
```



```
Program Developed by Tanav Singh Bajaj
 Scholar No. : 20U03037.
Enter No. of Elements : 4
Enter Elements :
2
5
1
4
Given Array is : 2 5 1 4

Sorted Array is : 1 2 4 5
```

Time Complexity:

As we have already learned in Binary Search that whenever we divide a number into half in every step, it can be represented using a logarithmic function, which is log n and the number of steps can be represented by log n + 1(at most) Also, we perform a single step operation to find out the middle of any subarray, i.e. O(1).

And to merge the subarrays, made by dividing the original array of n elements, a running time of O(n) will be required.

Hence the total time for mergeSort function will become n(log n + 1), which gives us a time complexity of O(n*log n).

Best Case : θ(nLogn)

Worst Case :θ(nLogn)

Average Case : θ(nLogn)