NAME – MANSHU JAISWAL
2023IMT-051

Merge Sort – Analysis

Merge Sort has a time complexity of O(n log n) for all cases: best, average, and worst. This arises because it recursively divides the array into two halves (O(log n) levels) and merges each half in linear time (O(n)).

Space complexity is O(n) due to the additional space required for merging the divided arrays. The algorithm needs extra space proportional to the array size to store the merged result. This space is used for temporary arrays during the merge step and does not include the space used by the recursive stack.

CODE :

```
#include <bits/stdc++.h>

using namespace std;


// It will run for n complexity

void merge(int arr[],int l,int mid,int h){
   int b[h-l+1];  //Space Complexity N
   int k=0;
   int i=l;
   int j = mid+1;
   while(i<=mid and j<=h) {
        if (arr[i]<arr[j])b[k++] = arr[i++];
        else b[k++] = arr[j++];
   }
   while (j<=h) b[k++] = arr[j++];
```

```
    while (i<=mid) b[k++] = arr[i++];


    for(int i=0;i<h-l+1;i++) {
        arr[l+i] = b[i];
    }
}

// It will run for(n*logn)
void mergesort(int arr[],int l,int h){
    int mid = (l+h)/2;
    if (l<h){
        mergesort(arr,0,mid);
        mergesort(arr,mid+1,h);
        merge(arr,l,mid,h);
    }
}
int main(){
    int arr[] = {5,6,7,2};
    int n = sizeof(arr)/sizeof(int);
    mergesort(arr,0,n);
    for(int i: arr) cout<<i<<" ";
}
```

Time Complexity – O(n*logn)

Space Complexity – O(N)

Quicksort – Analysis

Quick Sort has a time complexity of O(n^2) in the worst case but O(n log n) on average and best cases. The worst case occurs when the pivot selection results in highly unbalanced partitions. In contrast, on average, Quick Sort efficiently partitions the array into balanced halves, leading to O(n log n) time complexity.

Space complexity is O(log n) for the average case, which is due to the stack space used for recursive calls. However, in the worst case, it can degrade to O(n) if the recursion depth becomes excessive.

CODE :

```
#include <bits/stdc++.h>

using namespace std;


int partition(int arr[], int low, int high) {
   int pivot = arr[high];
   int i = low - 1;
   for (int j = low; j < high; ++j) {
      if (arr[j] <= pivot) {
         ++i;
         swap(arr[i], arr[j]);
      }
   }
   swap(arr[i + 1], arr[high]);
   return i + 1;
}


void quicksort(int arr[],int l,int h){
   if (l<h){
      int p = partition(arr,l,h);
      quicksort(arr,l,p-1);
      quicksort(arr,p+1,h);
```

```
    }
}

int main(){
    int arr[] = {5,6,7,2};
    int n = sizeof(arr)/sizeof(int);
    quicksort(arr,0,n);
    for(int i: arr) cout<<i<<" ";
}
```

Time Complexity = Best Case O(N^2)

=Worst Case O(N^2)

Space Complexity = O(1)