4. Problem Statement: Merge Sort

**Problem Analysis:**

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The idea of breaking down a list into several sub-lists until each sub-list consists of a single element and merging those sub-lists in a manner that results into a sorted list.

*1.* Divide the unsorted list into N sub-lists, each containing 1element.

*2.* Take adjacent pairs of two singleton lists and merge them to form a list of 2 elements. N will now convert into N/2lists of size 2.

*3.* Repeat the process till a single sorted list of obtained.

While comparing two sub-lists for merging, the first element of both lists is taken into consideration. While sorting in ascending order, the element that is of a lesser value becomes a new element of the sorted list. This procedure is repeated until both the smaller sub-lists are empty and the new combined sub-list comprises all the elements of both the sub-lists.

**Algorithm:**

Algorithm Merge(low, mid, high)

*// a[low : high] is a global array containing two sorted*

*// subsets in a[low : mid] and in a[mid+1 : high]. The goal*

*// is to merge these two sets into a single set residing*

*// in a[low : high]. b[] is an auxiliary global array.*

{

h := low; i := low; j := mid+1;

while ((h<=mid) and (j<=high)) do

{

if(a[h]<=a[j]) then

b[i] := a[h]; h := h+1;

else

{

b[i] := a[j];

j := j+1;

}

i := i+1;

}

if (h>mid) then

{

for k := j to high do

b[i] := a[k]; i := i+1;

}

else

{

for k := h to mid do

b[i] := a[k]; i := i+1;

}

for k := low to high do a[k] := b[k];

}

Algorithm MergeSort(low, high)

*// a[low : high] is a global array to be sorted.*

*// Small (P) is true if there is only one element*

*// to sort. In this case the list is already sorted.*

{

if (low<high) then *// If there are more than one elemant*

{

*// Divide P into subproblems.*

*// Find where to split the set.*

mid: = [(low + high)/2];

*// Solve the subproblems.*

MergeSort(low,mid);

MergeSort(mid+1,high);

*// Combine the solutions.*

Merge(low,mid,high);

}

}

**Source Code:**

#include<stdio.h>

void merge(int a[], int low, int mid, int high)

{

int b[100000],k;

int h=low, i=low, j=mid+1;

while(h<=mid && j<=high)

{

if(a[h]<=a[j])

{

b[i]=a[h];

h=h+1;

}

else

{

b[i]=a[j];

j=j+1;

}

i=i+1;

}

if(h>mid)

{

for(k=j;k<=high;k++)

{

b[i]=a[k];

i=i+1;

}

}

else

{

for(k=h; k<=mid; k++)

{

b[i]=a[k];

i=i+1;

}

}

for(k=low; k<=high; k++)

a[k]=b[k];

}

void mergesort(int a[], int low, int high)

{

if(low < high)

{

int m = (high + low)/2;

mergesort(a, low, m);

mergesort(a, m + 1, high);

merge(a, low, m, high);

}

}

int main()

{

int n,i, arr[100000];

printf("Enter array size: \n");

scanf("%d",&n);

printf("Enter array elements:\n");

for(i=1; i<=n; i++)

scanf("%d",&arr[i]);

mergesort(arr,1,n);

printf("Sorted array elements: \n");

for(i=1; i<=n; i++)

printf("%d ",arr[i]);

return 0;

}

**Sample Input:**

Enter array size:

10

Enter array elements:

310 285 179 652 351 423 861 254 450 520

**Sample Output:**

Sorted array elements:

179 254 285 310 351 423 450 520 652 861