***Task:-*** Implement 2-way merge sort for the given input array of elements < 20, 47, 15, 8, 9, 4, 40, 30, 12, 17 >. Demonstrate the following —-

1. Time complexity of both standard and 2-way merge sort.
2. Number of comparisons of both standard and 2-way merge sort.



**Pseudocode-1**

Mergesort( int list[ ],int low ,int high)

Int mid1,mid2

Mid1=(low +high)/2

Mid2=mid1+1

mergesort(a,l,mid1);

mergesort(a,mid2,r);

merge(a,low,mid1,mid2,high);

void merge(int \*a,int l,int mid1,int mid2,int r)

{

int i,j,k,c[50];

i=l; // low index

k=0;

j=mid1+1;

while(i<=mid1 && j<=mid2) // Sorting left sub-arrays

{

if(a[i]<a[j])

c[k++]=a[i++];

else

c[k++]=a[j++];

}

while(i<=mid1)

c[k++]=a[i++];

while(j<=mid2)

c[k++]=a[j++];

for(i=l;i<=mid2;i++)

a[i]=c[i-l];

i=l;

j=mid2+1;

k=0;

while(i<=mid2 && j<=r)

{

if(a[i]<a[j])

{

c[k++]=a[i++];

}

else

c[k++]=a[j++];

}

while(i<=mid2)

c[k++]=a[i++];

while(j<=r)

c[k++]=a[j++];

for(i=l;i<=r;i++)

a[i]=c[i-l];

**Code:**

#include<stdio.h>

void merge(int \*a,int l,int mid1,int mid2,int r)

{

int i,j,k,c[50];

i=l; // low index

k=0;

j=mid1+1;

while(i<=mid1 && j<=mid2) // Sorting left sub-arrays

{

if(a[i]<a[j])

c[k++]=a[i++];

else

c[k++]=a[j++];

}

while(i<=mid1) // left sub-array

c[k++]=a[i++];

while(j<=mid2) // right sub-array

c[k++]=a[j++];

for(i=l;i<=mid2;i++)

a[i]=c[i-l];

// \*\* \*\* \* \*\* \*\*\* \*\* \*\* \*\* \*\* \*

i=l;

j=mid2+1;

k=0;

while(i<=mid2 && j<=r) // sorting right sub-arrays

{

if(a[i]<a[j])

{

c[k++]=a[i++];

}

else

c[k++]=a[j++];

}

while(i<=mid2)

c[k++]=a[i++];

while(j<=r)

c[k++]=a[j++];

for(i=l;i<=r;i++)

a[i]=c[i-l];

printf("\n iterations: ");

for(i=l;i<=r;i++)

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

printf("\n");

}

void mergesort(int \*a,int l,int r)

{

if(l<r)

{

int mid1=(l+r)/2;

int mid2=mid1 + 1;

mergesort(a,l,mid1);

mergesort(a,mid2,r);

printf("range l= %d mid1=%d mid2=%d r=%d \n",l,mid1,mid2,r);

merge(a,l,mid1,mid2,r);

}

}

int main()

{

int a[20];

int i,n;

printf("enter no of elements in the array : \n");

scanf("%d",&n);

printf("enter the elements of the array : \n");

for(i=0;i<n;i++)

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

//printf("hi \n");

mergesort(a,0,n-1);

printf("the sorted array is : \n");

for(i=0;i<n;i++)

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

return 0;

}

**Pseudocode-2(single mergesort):**

void merge\_sort(int arr[], int low, int high)

int mid;

if (low < high)

mid=(low+high)/2;

merge\_sort(arr,low,mid);

merge\_sort(arr,mid+1,high);

//merge or conquer sorted arrays

merge(arr,low,high,mid)

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

int i, j, k, array2[50];

i = low;

k = low;

j = mid + 1;

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

if (arr[i] < arr[j])

array2[k] = arr[i];

k++;

i++;

else

array2[k] = arr[j];

k++;

j++;

while (i <= mid)

array2[k] = arr[i];

k++;

i++;

while (j <= high)

array2[k] = arr[j];

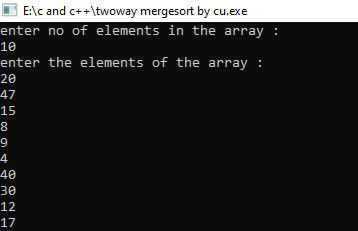
k++;

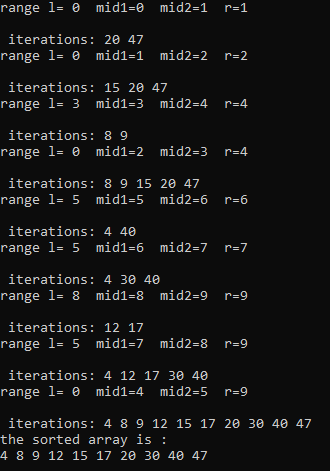
j++;

for i low; to k; i++)

cout<<array2[i]<<" ";

**Output:**

****

****

**Data type and Data structure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Datatype | Typical Value | Minimum Value | Maximum Value |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Input size | Expected Output | Actual Output | Comments |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |