Data Structures Odyssey: Exploring the Foundations of Computing

Ex. No.:15	Sorting	Date:23/05/2024

Write a C program to take n numbers and sort the numbers in ascending order. Try to implement the same using the following sorting techniques.

1. **Quick Sort**
2. **Merge Sort**

Algorithm:

1) Start
2) If the array has fewer than two elements, return it as it is already sorted.
3) Divide the array into two halves.
4) Recursively sort the two halves using Merge Sort.
5) Merge the sorted halves into a single sorted array.
6) Choose a sorting algorithm (e.g., Bubble Sort, Merge Sort, Quick Sort).
7) Implement the selected sorting algorithm.
8) Pass the unsorted array to the sorting algorithm.
9) Execute the sorting algorithm to sort the array. Obtain the sorted array as output. 10) stop

PROGRAM:

QUICK SORT:

```
#include
<stdio.h>
void
swap(int* a, int* b)
{
int temp = *a; *a
= *b;
*b = temp;
}
int partition(int arr[], int low, int high)
{ int pivot =
arr[low]; int i =
low; int j = high;

while (i < j) { while (arr[i] <= pivot &&
i<= high - 1) { i++;
}
while (arr[j] > pivot && j >= low + 1) {
j--; } if (i < j) {
swap(&arr[i], &arr[j]);
}
}
swap(&arr[low], &arr[j]); return
j;
}

void quickSort(int arr[], int low, int high)
{ if (low < high)
{
```

```
int partitionIndex = partition(arr, low, high);
quickSort(arr, low, partitionIndex - 1); quickSort(arr,
partitionIndex + 1, high);
}
}
int main()
{
int arr[] = { 19, 17, 15, 12, 16, 18, 4, 11, 13 };
int n = sizeof(arr) / sizeof(arr[0]);
printf("Original array:'); for (int i = 0; i<n; i++)
{ printf("&%d", arr[i]);
}
quickSort(arr, 0, n 0;
printf("\nSorted array:"); for
(int i = 0; i &lt; n; i++) {
printf("%d", arr[i]);
}

return 0;
}
```

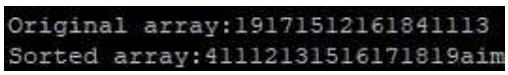
MERGE SORT

```
#include <stdio.h> #include
<stdlib.h> void merge(int arr[], int l,
int m, int r)
{ int i, j,
k;
int n1 = m - l + 1; int n2
= r - m; int L[n1], R[n2];
```

```
for (i = 0; i < n1; i++) L[i]
= arr[l + i]; for (j = 0; j <
n2; j++) R[j] = arr[m + 1
+ j]; i = 0; j = 0; k = l;
while (i < n1 && j < n2) {
if (L[i] <= R[j]) { arr[k] =
L[i]; i++;
} else {
arr[k] = R[j];
j++;
}
k++;
} while (i < n1)
{ arr[k] = L[i];
i++; k++;
}

while (j < n2) {
arr[k] = R[j]; j++;

k++;
}
}
void mergeSort(int arr[], int l, int r)
{ if (l < r) { int m = l + (r -
l) / 2; mergeSort(arr, l,
m); mergeSort(arr, m +
1, r); merge(arr, l, m, r);
}
}
void printArray(int A[], int size)
```

```
{ int i; for (i = 0;
i<size; i++)
printf("%d ", A[i]);


printf("\n");
}
int main()
{
int arr[] = { 12, 11, 13, 5, 6, 7 }; int
arr_size = sizeof(arr) / sizeof(arr[0]);
printf("Given array is \n");
printArray(arr, arr_size);
mergeSort(arr, 0, arr_size - 1);
printf("\nSorted array is \n");
printArray(arr, arr_size); return 0;
}
```

OUTPUT:



RESULT: Thus, the program was successfully executed.