Design and Analysis of Algorithms

Week-4

Name: Musali Reddy Manish Reddy

Roll.no :CH.SC.U4CSE24129

Merge sort:

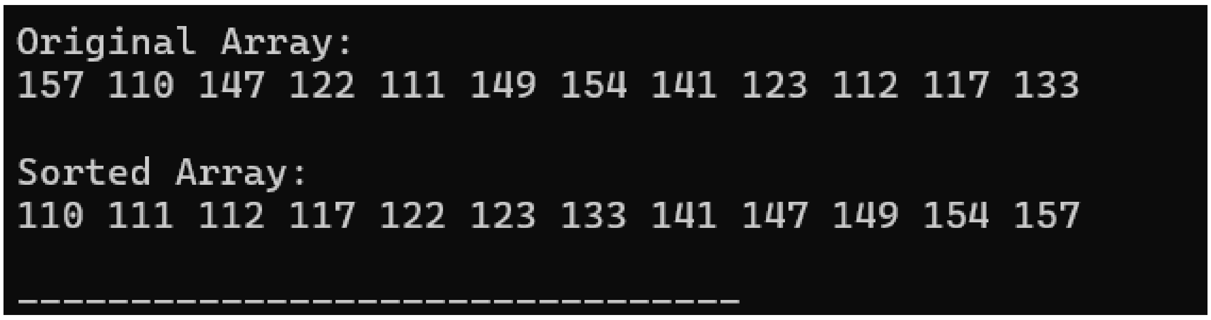
Code:

```
//CH.SC.U4CSE24129
#include <stdio.h>
#define MAX 50
void merge(int arr[], int left, int mid, int right) {
   int i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;
   int L[MAX], R[MAX];
   for (i = 0; i < n1; i++)
      L[i] = arr[left + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[mid + 1 + j];
   i = 0;
   j = 0;
   k = left;

   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 left, int right) {
    int mid;
    if (left < right) {
        mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
void printArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    int arr[] = {157, 110, 147, 122, 111, 149, 154, 141, 123, 112, 117, 133};
    int n = 12;

    printf("Original Array:\n");
    printArray(arr, n);
    mergeSort(arr, 0, n - 1);
    printf("\nSorted Array:\n");
    printArray(arr, n);
    return 0;
}
```

Output:



```
Original Array:
157 110 147 122 111 149 154 141 123 112 117 133

Sorted Array:
110 111 112 117 122 123 133 141 147 149 154 157

--------------------------------
```

Time Complexity: O(n log n)

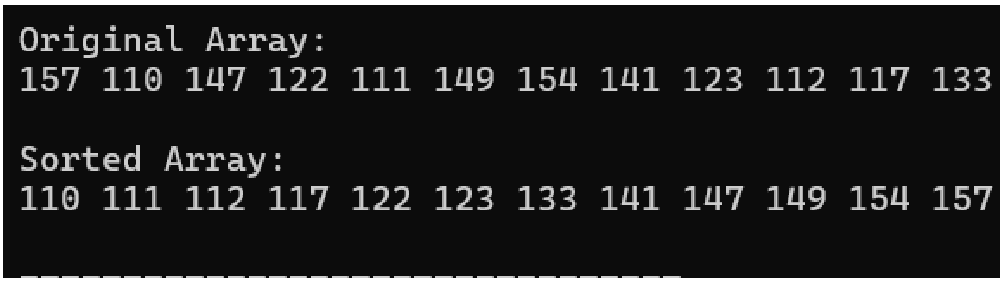
Space Complexity: O(n)

Quick sort:

Code:

```
#include <stdio.h>
void exchange(int *x, int *y) {
    int holder;
    holder = *x;
    *x = *y;
    *y = holder;
}
int split(int data[], int start, int end) {
    int reference;
    int left, right;
    reference = data[end];
    left = start - 1;
    for (right = start; right < end; right++) {
        if (data[right] <= reference) {
            left++;
            exchange(&data[left], &data[right]);
        }
    }
    exchange(&data[left + 1], &data[end]);
    return (left + 1);
}
void quickArrange(int data[], int start, int end) {
    int position;
    if (start < end) {
        position = split(data, start, end);
        quickArrange(data, start, position - 1);
       quickArrange(data, position + 1, end);
    }
}
void show(int data[], int size) {
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", data[i]);
    printf("\n");
}
int main() {
    int data[] = {157, 110, 147, 122, 111, 149, 154, 141, 123, 112, 117, 133};
    int size = 12;
    printf("Original Array:\n");
    show(data, size);
    quickArrange(data, 0, size - 1);
    printf("\nSorted Array:\n");
    show(data, size);
    return 0;
}
```

Output:



```
Original Array:
157 110 147 122 111 149 154 141 123 112 117 133

Sorted Array:
110 111 112 117 122 123 133 141 147 149 154 157
```

Time Complexity:

Best case: O(n log n)

Worst case: O(n^2)

Space Complexity: O(log n)