1-merge sort :

#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {

int i = left, j = mid + 1, k = 0;

int temp[right - left + 1];

while (i <= mid && j <= right) {

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

temp[k++] = arr[i++];

} else {

temp[k++] = arr[j++];

}

}

while (i <= mid) {

temp[k++] = arr[i++];

}

while (j <= right) {

temp[k++] = arr[j++];

}

for (i = left, k = 0; i <= right; i++, k++) {

arr[i] = temp[k];

}

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = (left + right) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

int main() {

int n, i;

printf(" number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

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

}

mergeSort(arr, 0, n - 1);

printf("Sorted array: ");

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

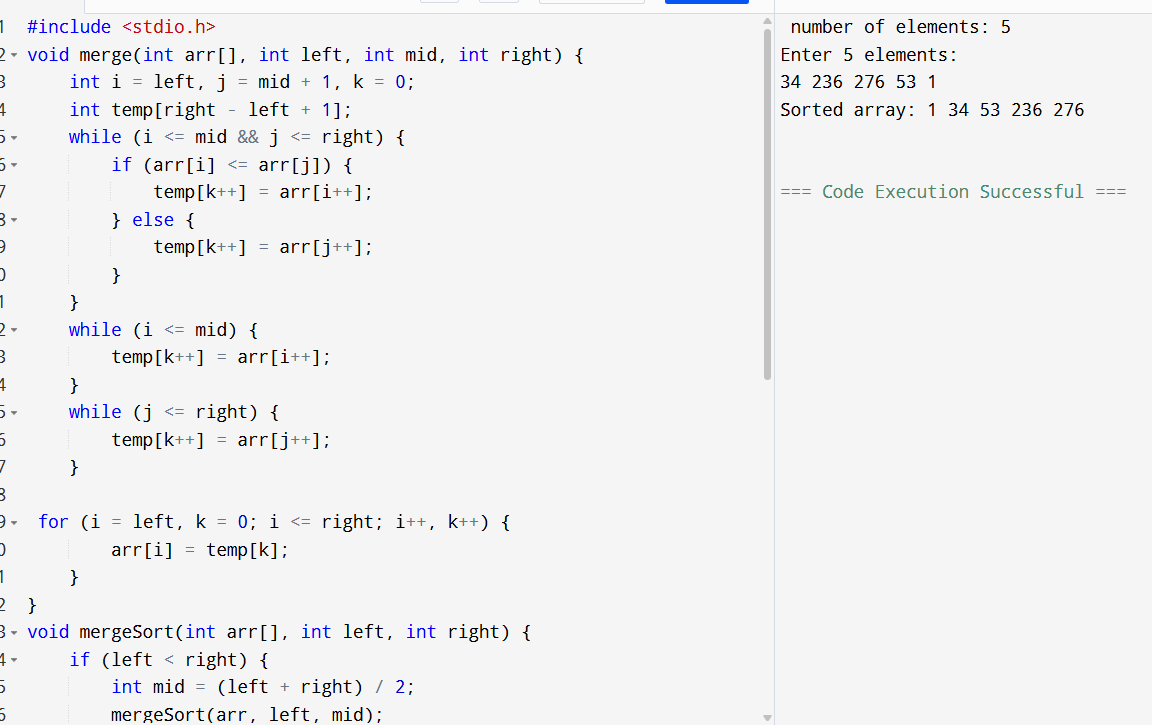
printf("%d ", arr[i]);

}

printf("\n");

return 0;

}



2.#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 p = arr[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j] <= p) {

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

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

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void print(int arr[], int size) {

for (int i = 0; i < size; i++)

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

printf("\n");

}

int main() {

int n;

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

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

printf("Original array:\n");

print(arr, n);

quickSort(arr, 0, n - 1);

printf("Sorted array:\n");

print(arr, n);

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.

3-

#include <stdio.h>

#include <stdlib.h>

#define BUCKET\_COUNT 10

typedef struct Node {

int data;

struct Node\* next;

} Node;

void insertSorted(Node\*\* bucket, int value) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = value;

newNode->next = NULL;

if (\*bucket == NULL || value < (\*bucket)->data) {

newNode->next = \*bucket;

\*bucket = newNode;

} else {

Node\* current = \*bucket;

while (current->next != NULL && current->next->data <= value)

current = current->next;

newNode->next = current->next;

current->next = newNode;

}

}

void bucketSort(int arr[], int n) {

Node\* buckets[BUCKET\_COUNT] = { NULL };

int max = arr[0];

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

if (arr[i] > max)

max = arr[i];

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

int index = (arr[i] \* BUCKET\_COUNT) / (max + 1);

insertSorted(&buckets[index], arr[i]);

}

int idx = 0;

for (int i = 0; i < BUCKET\_COUNT; i++) {

Node\* current = buckets[i];

while (current != NULL) {

arr[idx++] = current->data;

Node\* temp = current;

current = current->next;

free(temp);

}

}

}

void printAr(int arr[], int n) {

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

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

printf("\n");

}

int main() {

int n;

printf(" number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d non-negative integers:\n", n);

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

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

printf("Original array:\n");

printAr(arr, n);

bucketSort(arr, n);

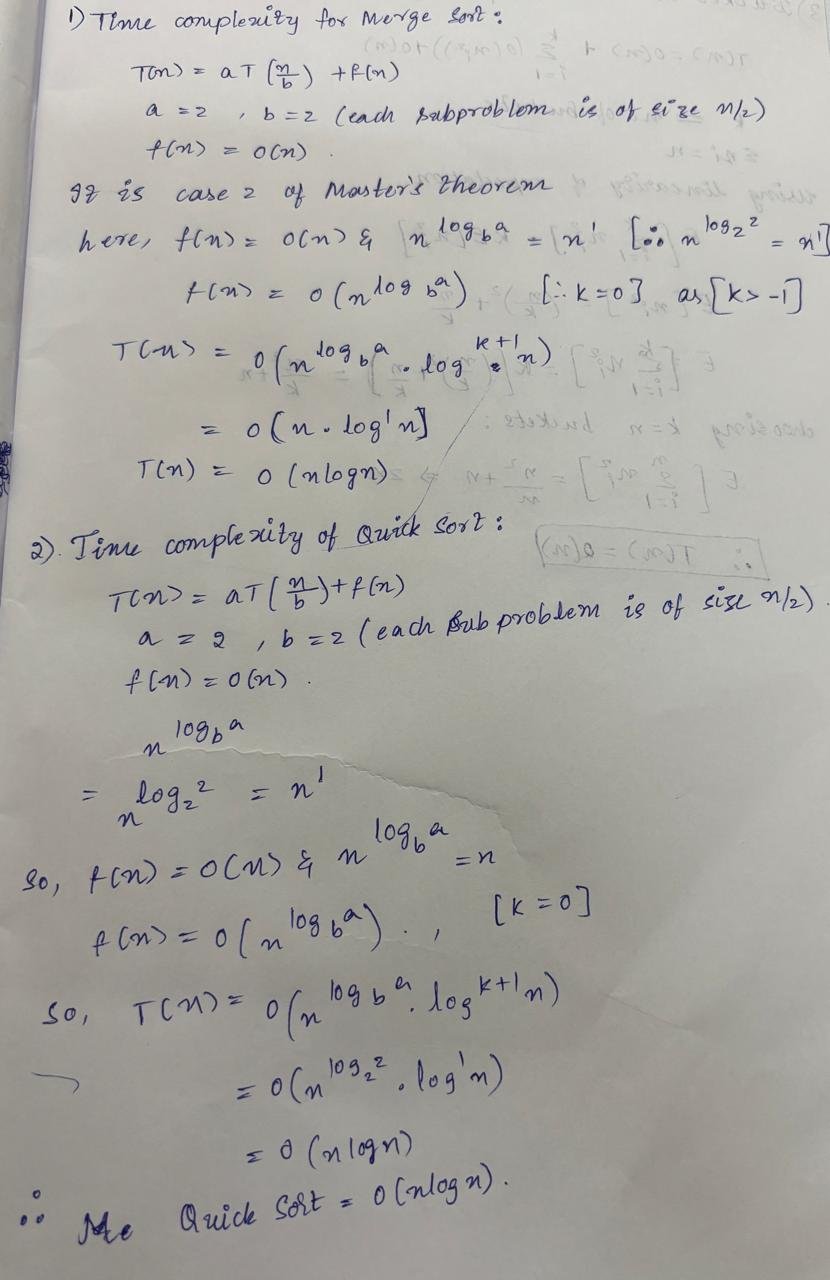
printf("Sorted array:\n");

printAr(arr, n);

return 0;

}





A piece of paper with math equations

AI-generated content may be incorrect.