MERGE SORT

#include <iostream>

using namespace std;

// Function to merge two sorted halves

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

int n1 = mid - left + 1; // size of left subarray

int n2 = right - mid; // size of right subarray

// Temporary arrays

int L[n1], R[n2];

// Copy data into temporary arrays

for (int i = 0; i < n1; i++)

L[i] = arr[left + i];

for (int j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

// Merge temp arrays back into arr[left..right]

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (L[i] <= R[j])

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

else

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

}

// Copy remaining elements (if any)

while (i < n1) arr[k++] = L[i++];

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

}

// Recursive function to divide array

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

if (left < right) {

int mid = (left + right) / 2;

// Sort first and second halves

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

// Merge sorted halves

merge(arr, left, mid, right);

}

}

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

int arr[n];

cout << "Enter elements:\n";

for (int i = 0; i < n; i++) cin >> arr[i];

mergeSort(arr, 0, n - 1);

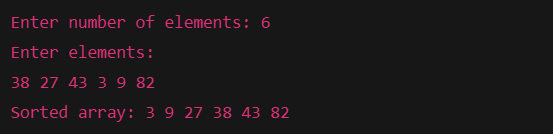
cout << "Sorted array: ";

for (int i = 0; i < n; i++) cout << arr[i] << " ";

cout << endl;

return 0;

}



BUCKET SORT

#include <iostream>

#include <vector>

#include <algorithm> // for sort()

using namespace std;

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

// Create n empty buckets

vector<float> buckets[n];

// Put array elements into different buckets

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

int bi = n \* arr[i]; // index in bucket

buckets[bi].push\_back(arr[i]);

}

// Sort individual buckets

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

sort(buckets[i].begin(), buckets[i].end());

}

// Concatenate all buckets back into arr[]

int index = 0;

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

for (float x : buckets[i]) {

arr[index++] = x;

}

}

}

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

float arr[n];

cout << "Enter elements between 0 and 1:\n";

for (int i = 0; i < n; i++) cin >> arr[i];

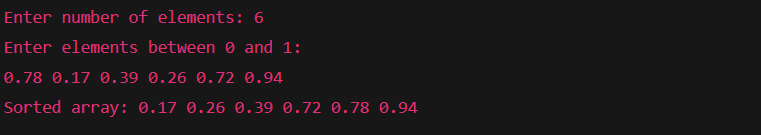
bucketSort(arr, n);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) cout << arr[i] << " ";

cout << endl;

return 0;

}

QUICK SORT

#include <iostream>

using namespace std;

// Partition function (places pivot at right position)

int partition(int arr[], int low, int high) {

int pivot = arr[high]; // pivot element

int i = low - 1; // index of smaller element

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

if (arr[j] <= pivot) {

i++;

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

}

}

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

return (i + 1);

}

// QuickSort function

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

if (low < high) {

// Partition index

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

// Sort left and right subarrays

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

int arr[n];

cout << "Enter elements:\n";

for (int i = 0; i < n; i++) cin >> arr[i];

quickSort(arr, 0, n - 1);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) cout << arr[i] << " ";

cout << endl;

return 0;

}