Quick Sort

#include <iostream>

using namespace std;

// Partition function

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

int pivot = arr[high];

int i = low - 1;

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) {

while (low < high) {

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

if (pi - low < high - pi) { // Optimize to reduce stack depth

quickSort(arr, low, pi - 1);

low = pi + 1;

} else {

quickSort(arr, pi + 1, high);

high = pi - 1;

}

}

}

Merge Sort

#include <iostream>

using namespace std;

// Merge two subarrays

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

int n1 = mid - left + 1;

int n2 = right - mid;

int L[n1], R[n2];

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];

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

while (i < n1 && j < n2) arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];

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

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

}

// MergeSort function

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

while (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

break; // Add loop functionality by breaking recursion

}

}

Insertion Sort

#include <iostream>

using namespace std;

// InsertionSort function

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

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j--;

}

arr[j + 1] = key;

}

}

Selection Sort

#include <iostream>

using namespace std;

// SelectionSort function

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

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

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

}

}

Main

int main() {

int arr[] = {64, 34, 25, 12, 22, 11, 90};

int n = sizeof(arr) / sizeof(arr[0]);

// Uncomment one sort at a time to test

// quickSort(arr, 0, n - 1);

// mergeSort(arr, 0, n - 1);

// insertionSort(arr, n);

// selectionSort(arr, n);

cout << "Sorted array: ";

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

cout << endl;

return 0;

}