Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

I. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts ( shifts - total number of times the array elements are shifted from their place) required for sorting the array.

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

void insertionSort(int arr[], int n, int& comparisons, int& shifts) {

int i, j, key;

comparisons = 0;

shifts = 0;

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

key = arr[i];

j = i - 1;

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

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

j--;

comparisons++;

shifts++;

}

arr[j + 1] = key;

shifts++;

}

}

int main() {

ifstream inputFile("input.txt");

if (!inputFile) {

cerr << "Error opening input file." << endl;

return 1;

}

ofstream outputFile("output.txt");

if (!outputFile) {

cerr << "Error opening output file." << endl;

return 1;

}

int T;

inputFile >> T;

for (int testCase = 0; testCase < T; testCase++) {

int n, comparisons, shifts;

inputFile >> n;

vector<int> arr(n);

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

inputFile >> arr[i];

}

insertionSort(&arr[0], n, comparisons, shifts);

outputFile << "Test Case " << testCase + 1 << ":" << endl;

outputFile << "Sorted array: ";

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

outputFile << arr[i] << " ";

outputFile << "\nNumber of comparisons: " << comparisons << endl;

outputFile << "Number of shifts: " << shifts << endl;

}

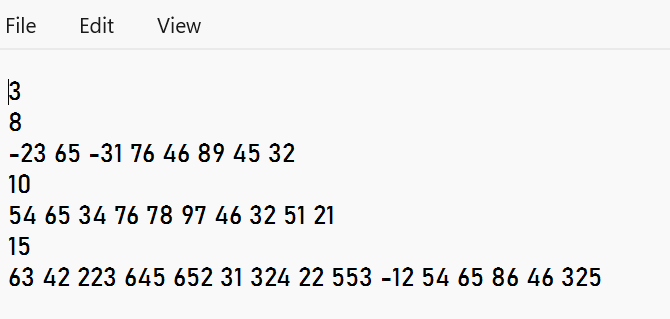
inputFile.close();

outputFile.close();

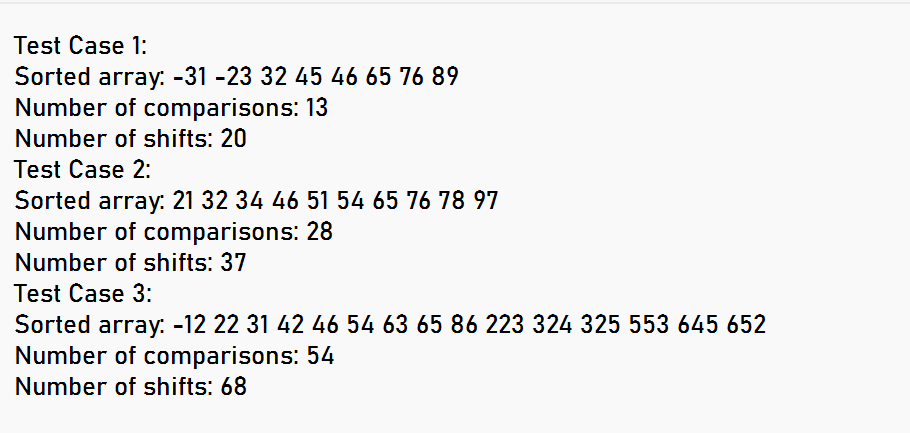
return 0;

}

INPUT FILE



OUTPUT FILE



Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

II. Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

void selectionSort(int arr[], int n, int& comparisons, int& swaps) {

comparisons = 0;

swaps = 0;

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

int min\_idx = i;

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

if (arr[j] < arr[min\_idx]) {

min\_idx = j;

}

comparisons++;

}

if (min\_idx != i) {

swap(arr[i], arr[min\_idx]);

swaps++;

}

}

}

int main() {

ifstream inputFile("input.txt");

if (!inputFile) {

cerr << "Error opening input file." << endl;

return 1;

}

ofstream outputFile("output.txt");

if (!outputFile) {

cerr << "Error opening output file." << endl;

return 1;

}

int T;

inputFile >> T;

for (int testCase = 0; testCase < T; testCase++) {

int n, comparisons, swaps;

inputFile >> n;

vector<int> arr(n);

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

inputFile >> arr[i];

}

selectionSort(&arr[0], n, comparisons, swaps);

outputFile << "Test Case " << testCase + 1 << ":" << endl;

outputFile << "Sorted array: ";

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

outputFile << arr[i] << " ";

outputFile << "\nNumber of comparisons: " << comparisons << endl;

outputFile << "Number of swaps: " << swaps << endl;

}

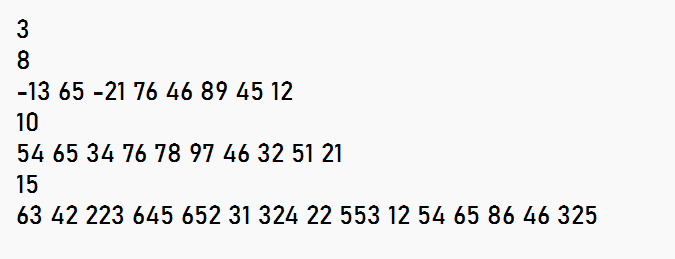
inputFile.close();

outputFile.close();

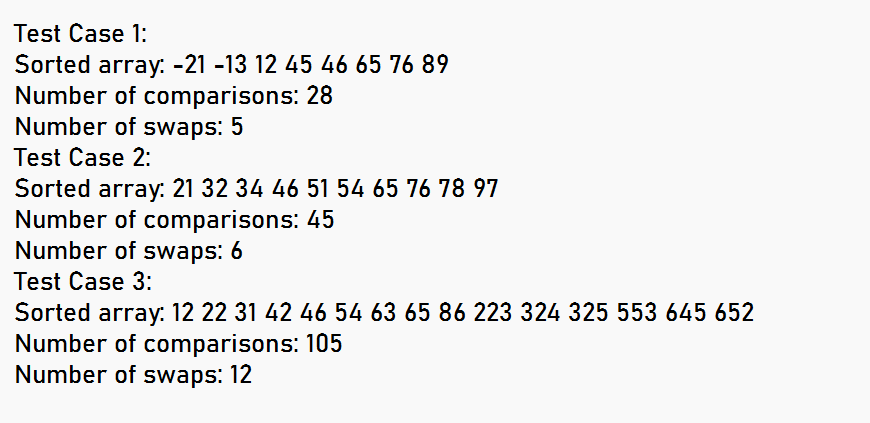
return 0;

}

INPUT FILE



OUTPUT FILE



Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

III. Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

void merge(vector<int>& arr, int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

vector<int> L(n1), R(n2);

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

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

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

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

int 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(vector<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);

}

}

bool hasDuplicates(vector<int>& arr) {

mergeSort(arr, 0, arr.size() - 1);

for (int i = 0; i < arr.size() - 1; i++) {

if (arr[i] == arr[i + 1]) {

return true;

}

}

return false;

}

int main() {

ifstream inputFile("input.txt");

if (!inputFile) {

cerr << "Error opening input file." << endl;

return 1;

}

ofstream outputFile("output.txt");

if (!outputFile) {

cerr << "Error opening output file." << endl;

return 1;

}

int T;

inputFile >> T;

for (int testCase = 0; testCase < T; testCase++) {

int n;

inputFile >> n;

vector<int> arr(n);

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

inputFile >> arr[t];

}

outputFile << "Test Case " << testCase + 1 << ":" << endl;

if (hasDuplicates(arr)) {

outputFile << "YES\n";

} else {

outputFile << "NO\n";

}

}

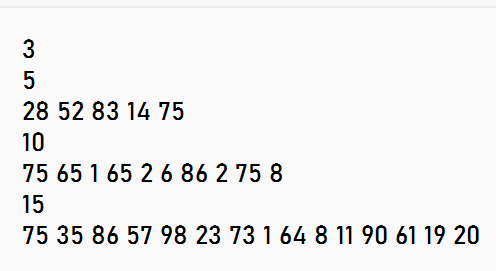
inputFile.close();

outputFile.close();

return 0;

}

INPUT FILE



OUTPUT FILE

