15B17CI371 – Data Structures Lab

Week 3-LAB B

Q1. Given an array containing n distinct numbers in the range [0, n]. Write a

program to return the only number in the range that is missing from the array.

#include <iostream>

#include <vector>

using namespace std;

void bubble\_sort(vector<int>& arr) {

int n = arr.size();

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

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

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

int findmiss(vector<int>& arr, int size) {

bubble\_sort(arr);

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

if (arr[i] != i) {

return i;

}

}

return size;

}

int main() {

int n;

cout << "Enter the number of elements : ";

cin >> n;

int size = n + 1;

vector<int> arr(size - 1);

cout << "Enter " << size - 1 << " elements: " << endl;

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

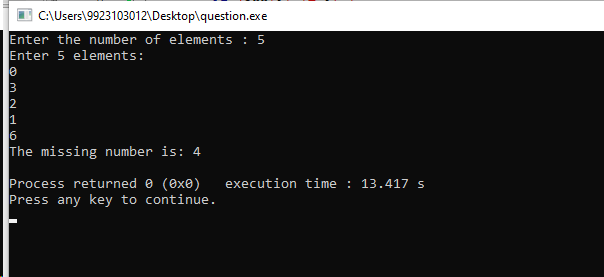
cin >> arr[i];

}

cout << "The missing number is: " << findmiss(arr, size) << endl;

return 0;

}



Q2. Given a 1D array of integers, first sort the array in non-decreasing order, and

then find two numbers such that the sum of two numbers add up to a specific

value. If such a pair of numbers can be found in the array, return the indices, else

return a suitable message.

Example 1:

Input: numbers = [2,7,11,15], target = 9

Output: [1,2]

Hint: The sum of 2 and 7 is 9.

Example 2:

Input: numbers = [2,3,4], target = 6

Output: [1,3]

Hint: The sum of 2 and 4 is 6.

#include <iostream>

#include <vector>

using namespace std;

void bubble\_sort(vector<int>& arr) {

int n = arr.size();

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

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

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

vector<int> find\_pair(vector<int>& nums, int target) {

int n = nums.size();

vector<int> sorted\_nums = nums;

bubble\_sort(sorted\_nums);

int left = 0;

int right = n - 1;

while (left < right) {

int sum = sorted\_nums[left] + sorted\_nums[right];

if (sum == target) {

vector<int> result(2);

bool found\_left = false, found\_right = false;

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

if (!found\_left && nums[i] == sorted\_nums[left]) {

result[0] = i + 1;

found\_left = true;

}

if (!found\_right && nums[i] == sorted\_nums[right]) {

result[1] = i + 1;

found\_right = true;

}

if (found\_left && found\_right) break;

}

return result;

} else if (sum < target) {

++left;

} else {

--right;

}

}

return {};

}

int main() {

int n, target;

cout << "Enter the number of elements: ";

cin >> n;

vector<int> nums(n);

cout << "Enter the elements:\n";

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

cin >> nums[i];

}

cout << "Enter the target sum: ";

cin >> target;

vector<int> result = find\_pair(nums, target);

if (result.empty()) {

cout << "No such pair exists.\n";

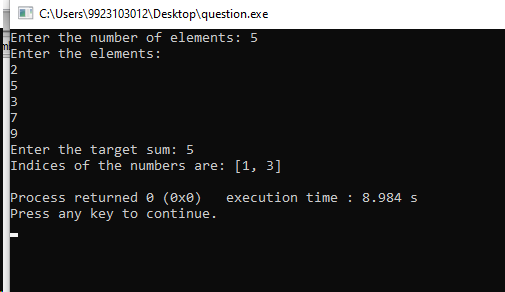
} else {

cout << "Indices of the numbers are: [" << result[0] << ", " << result[1] << "]\n";

}

return 0;

}



Q3. You are given a list of numbers. They are unsorted. Sort this list. Assume that

consecutive elements form pairs of numbers. Determine which pair or pairs of

elements have the smallest absolute difference between them.

Example:

Arr = {2,5,4,89,1}

Sorted array= {1,2,4,5,89}

Pairs: {1,2}, {2,4}, {4,5}, {5,89}; Difference: 1,2,1,84

Output: Smallest: {1,2}, {4,5}

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

void bubble\_sort(vector<int>& arr) {

int n = arr.size();

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

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

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

void find\_min(vector<int>& arr) {

int n = arr.size();

if (n < 2) {

cout << "Not enough elements to form pairs.\n";

return;

}

bubble\_sort(arr);

int min\_diff = INT\_MAX;

vector<pair<int, int> > min\_pairs;

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

int diff = (arr[i] > arr[i + 1]) ? (arr[i] - arr[i + 1]) : (arr[i + 1] - arr[i]);

if (diff < min\_diff) {

min\_diff = diff;

min\_pairs.clear();

min\_pairs.push\_back({arr[i], arr[i + 1]});

} else if (diff == min\_diff) {

min\_pairs.push\_back({arr[i], arr[i + 1]});

}

}

cout << "Smallest Difference: " << min\_diff << endl;

for (int i = 0; i < min\_pairs.size(); ++i) {

cout << "{" << min\_pairs[i].first << "," << min\_pairs[i].second << "} ";

}

cout << endl;

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

vector<int> arr(n);

cout << "Enter " << n << " elements: " << endl;

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

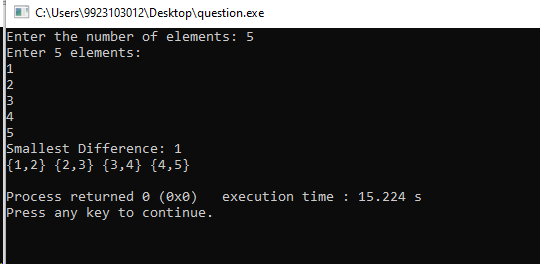
cin >> arr[i];

}

find\_min(arr);

return 0;

}



Q4. Given a sorted array of size N and an integer K, find the position at which K is

present in the array using interpolation search.

Example

Input: N = 5, arr[ ] = {1 2 3 4 5}, K = 4

Output: 3

Explanation: 4 appears at index 3

#include <iostream>

using namespace std;

int interpolation\_search(int arr[], int n, int k) {

int low = 0;

int high = n - 1;

while (low <= high && k >= arr[low] && k <= arr[high]) {

int pos = low + ((k - arr[low]) \* (high - low) / (arr[high] - arr[low]));

if (arr[pos] == k) {

return pos;

}

else if (arr[pos] < k) {

low = pos + 1;

}

else {

high = pos - 1;

}

}

return -1;

}

int main() {

int n, k;

cout << "Enter the number of elements: ";

cin >> n;

int arr[n];

cout << "Enter the elements in sorted order: " << endl;

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

cin >> arr[i];

}

cout << "Enter the value to search for: ";

cin >> k;

int result = interpolation\_search(arr, n, k);

if (result != -1) {

cout << "The position of " << k << " is: " << result << endl;

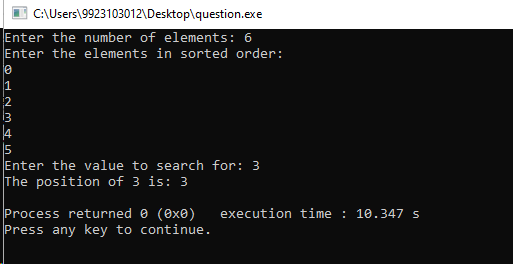
} else {

cout << k << " is not present in the array." << endl;

}

return 0;

}



Q5. Given an array of n distinct elements. Write a program to find the minimum

number of swaps required to sort the array in strictly increasing order.

#include <iostream>

#include <vector>

using namespace std;

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

void bubble\_sort(int arr[], int indices[], int n) {

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

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

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

swap(indices[j], indices[j + 1]);

}

}

}

}

int min\_swaps(int arr[], int n) {

int indices[n];

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

indices[i] = i;

}

bubble\_sort(arr, indices, n);

bool visited[n] = {false};

int swaps = 0;

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

if (visited[i] || indices[i] == i) {

continue;

}

int cycle\_size = 0;

int j = i;

while (!visited[j]) {

visited[j] = true;

j = indices[j];

++cycle\_size;

}

if (cycle\_size > 0) {

swaps += (cycle\_size - 1);

}

}

return swaps;

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

int arr[n];

cout << "Enter the elements: " << endl;

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

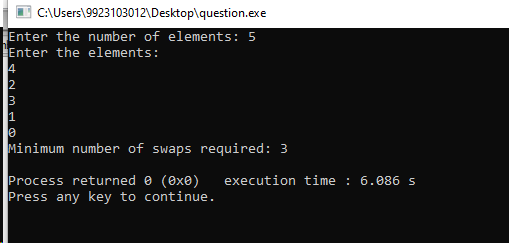
cin >> arr[i];

}

cout << "Minimum number of swaps required: " << min\_swaps(arr, n) << endl;

return 0;

}



Q6. Given an array of integers. Write a program to find the Inversion Count in

the array.

Inversion Count: For an array, inversion count indicates how far (or close) the

array is from being sorted. If the array is already sorted then the inversion count

is 0. If an array is sorted in the reverse order then the inversion count is the

maximum.

Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

#include <iostream>

#include <vector>

using namespace std;

int merge\_and\_count(vector<int>& arr, int temp[], int left, int mid, int right) {

int i = left;

int j = mid + 1;

int k = left;

int inv\_count = 0;

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

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

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

} else {

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

inv\_count += (mid - i + 1);

}

}

while (i <= mid) {

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

}

while (j <= right) {

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

}

for (i = left; i <= right; ++i) {

arr[i] = temp[i];

}

return inv\_count;

}

int merge\_sort\_and\_count(vector<int>& arr, int temp[], int left, int right) {

int inv\_count = 0;

if (left < right) {

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

inv\_count += merge\_sort\_and\_count(arr, temp, left, mid);

inv\_count += merge\_sort\_and\_count(arr, temp, mid + 1, right);

inv\_count += merge\_and\_count(arr, temp, left, mid, right);

}

return inv\_count;

}

int count\_inversions(vector<int>& arr) {

int n = arr.size();

int\* temp = new int[n];

int inv\_count = merge\_sort\_and\_count(arr, temp, 0, n - 1);

delete[] temp;

return inv\_count;

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

vector<int> arr(n);

cout << "Enter the elements: " << endl;

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

cin >> arr[i];

}

cout << "Inversion count: " << count\_inversions(arr) << endl;

return 0;

}