**9923103023 – F1**

**15B17CI371 – Data Structures Lab ODD 2024**

**Week 4-LAB A**

**Practice Lab**

1. Write a program to find all occurrence of a key within a given array using

sequential search algorithm.

Test Case:

Input:

array = {16, 31, 15, 27, 9, 15, 39, 15, 17, 12}; Key: 15

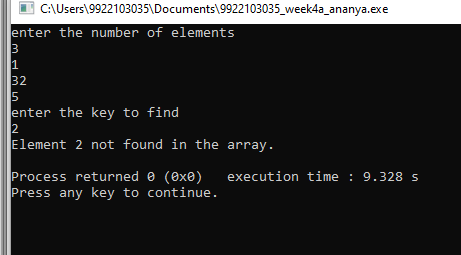
Output:

Element found at index 2

Element found at index 5

Element found at index 7

Ans :



#include <iostream>

using namespace std;

int sequential\_search( int arr[], int size, int key) {

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

if (arr[i] == key) {

return i;

}

}

return -1; // Key not found

}

int main() {

int n;

cout<<"enter the number of elements"<<endl;

cin>>n;

int \*test\_array = new int[n];

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

cin>>test\_array[i];

}

int key\_to\_find ;

cout<<"enter the key to find"<<endl;

cin>>key\_to\_find;

int index = sequential\_search(test\_array, sizeof(test\_array) / sizeof(test\_array[0]), key\_to\_find);

if (index != -1) {

cout << "Element " << key\_to\_find << " found at index: " << index << endl;

} else {

cout << "Element " << key\_to\_find << " not found in the array." << endl;

}

delete[] test\_array;

return 0;

}

2. Given an unsorted array and a number n, find if there exists a pair of

elements in the array whose product is given number n.

Input: arr[] = {5, 20, 3, 2, 50, 80}, n = 150

Output: Pair Found: (3, 50)

#include <iostream>

using namespace std;

bool has\_pair\_with\_product(const int arr[], int size, int n) {

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

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

if (arr[i] \* arr[j] == n) {

cout << "Pair Found: (" << arr[i] << ", " << arr[j] << ")" << endl;

return true;

}

}

}

cout << "No such pair exists." << endl;

return false;

}

int main() {

int n;

cout<<"enter the number of elements"<<endl;

cin>>n;

int \*input\_array = new int[n];

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

cin>>input\_array[i];

}

int target\_product ;

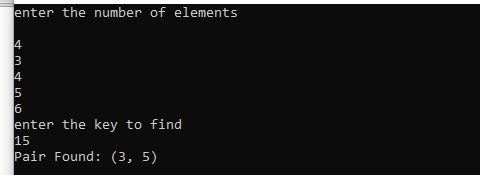
cout<<"enter the key to find"<<endl;

cin>>target\_product;

has\_pair\_with\_product(input\_array, n, target\_product);

return 0;

}



3. Given an unsorted array of integers, sort the array into a wave-like array.

An array arr[] is in wave form if arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4]

>=...

Input: arr[] = {10, 90, 49, 2, 1, 5, 23}

Output: Pair Found: (10, 2, 90, 1, 49, 5, 23)

#include <iostream>

using namespace std;

void sort\_into\_wave(int arr[], int n) {

for (int i = 0; i < n; i += 2) {

if (i > 0 && arr[i] < arr[i - 1]) {

swap(arr[i], arr[i - 1]);

}

if (i < n - 1 && arr[i] < arr[i + 1]) {

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

}

}

}

int main() {

int n;

cout<<"enter the number of elements"<<endl;

cin>>n;

int \*input\_arr = new int[n];

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

cin>>input\_arr[i];

}

sort\_into\_wave(input\_arr, n);

cout << "Wave-like array: ";

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

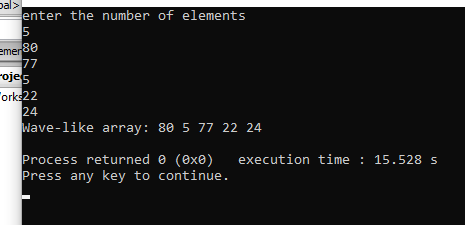
cout << input\_arr[i] << " ";

}

cout << endl;

return 0;

}



4. Perform the above-mentioned questions using binary and interpolation search

techniques as well. Try to figure out the differences not only in the algorithms

but also in the number of iterations, number of swaps, etc.

binary search

#include <iostream>

namespace std;

int binary\_search(const int arr[], int size, int key) {

int left = 0, right = size - 1;

while (left <= right) {

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

if (arr[mid] == key) {

return mid;

} else if (arr[mid] < key) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1; // Key not found

}

int main() {

int sorted\_array[] = {1, 2, 5, 9, 12, 15, 15, 16, 17, 27, 31, 39};

int key\_to\_find = 15;

int index = binary\_search(sorted\_array, sizeof(sorted\_array) / sizeof(sorted\_array[0]), key\_to\_find);

if (index != -1) {

cout << "Element " << key\_to\_find << " found at index: " << index << endl;

} else {

cout << "Element " << key\_to\_find << " not found in the array." << endl;

}

return 0;

}

Interpolation search

#include <iostream>

namespace std;

int interpolation\_search(const int arr[], int size, int key) {

int left = 0, right = size - 1;

while (left <= right && key >= arr[left] && key <= arr[right]) {

int pos = left + ((key - arr[left]) \* (right - left)) / (arr[right] - arr[left]);

if (arr[pos] == key) {

return pos;

} else if (arr[pos] < key) {

left = pos + 1;

} else {

right = pos - 1;

}

}

return -1;

}

int main() {

int sorted\_array[] = {1, 2, 5, 9, 12, 15, 15, 16, 17, 27, 31, 39};

int key\_to\_find = 15;

int index = interpolation\_search(sorted\_array, sizeof(sorted\_array) / sizeof(sorted\_array[0]), key\_to\_find);

if (index != -1) {

cout << "Element " << key\_to\_find << " found at index: " << index << endl;

} else {

cout << "Element " << key\_to\_find << " not found in the array." << endl;

}

return 0;

}