

15B17CI371 – Data Structures Lab

ODD 2024

Week 4-LAB A

Practice Lab

[CO: C270.2]

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

```
Enter the value of key: 0
key 0 is found at:
INDEX: 2
INDEX: 6
INDEX: 9

Process returned 0 (0x0)   execution time : 1.871 s
Press any key to continue.
```

```

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int ssearch(int arr[],int n,int k)
{
    vector<int> v;

    for(int i=0;i<n;i++)
    {
        if(arr[i]==k)
        {
            v.push_back(i);
        }
    }

    cout<<"key "<<k<<" is found at: "<<endl;

    for(int i=0;i<v.size();i++)
    {
        cout<<"INDEX: "<<v[i]<<endl;
    }
}

int main()
{
    int arr[10]={7,10,0,2,4,6,0,5,2,0};

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

    int k;

```

```

cout<<"Enter the value of key: ";

cin>>k;

sssearch(arr,n,k);

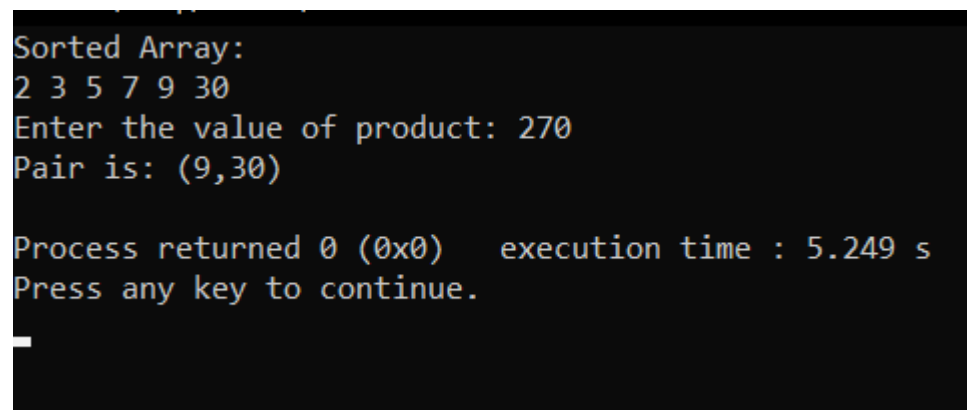
}

```

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)



```

Sorted Array:
2 3 5 7 9 30
Enter the value of product: 270
Pair is: (9,30)

Process returned 0 (0x0)   execution time : 5.249 s
Press any key to continue.

```

```

int part(int arr[],int low,int high)
{
    int i=low;
    int j=high;
    int pivot=arr[low];
    while(i<j)
    {
        while(arr[i]<=pivot && i<=high)
        {

```

```

        i++;
    }
    while(arr[j]>pivot && j>=low)
    {
        j--;
    }
    if(i<j)
    {
        swap(arr[i],arr[j]);
    }
}
swap(arr[low],arr[j]);
return j;
}

```

```

void quicksort(int arr[],int low,int high)
{
    if(low<high)
    {
        int p=part(arr,low,high);
        quicksort(arr,low,p-1);
        quicksort(arr,p+1,high);
    }
}

```

```

void prod(int arr[],int n)
{
    int p;

```

```

cout<<"Enter the value of product: ";

cin>>p;

int low=0;

int high=n-1;

while(low<=high)

{

    int val=arr[low]*arr[high];

    if(val==p)

    {

        cout<<"Pair is: ("<<arr[low]<<","<<arr[high]<<")"<<endl;

        exit(0);

    }

    else if(p<val)

    {

        high--;

    }

    else

    {

        low++;

    }

}

}

int main()

{

    int arr[6]={30,3,2,5,7,9};

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

    quicksort(arr,0,n-1);

```

```

cout<<"Sorted Array: "<<endl;
for(int i=0;i<6;i++)
{
    cout<<arr[i]<<" ";
}
cout<<endl;
prod(arr,n);

}

```

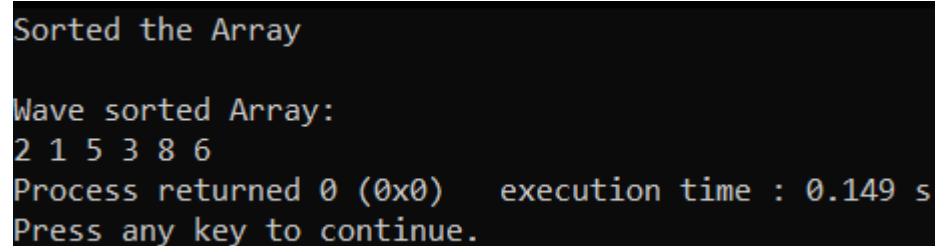
3. Given an unsorted array of integers, sort the array into a wave-like array.

An array arr[] is in wave form if $arr[0] \geq arr[1] \leq arr[2] \geq arr[3] \leq arr[4]$

$\geq \dots$

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

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



```

Sorted the Array
Wave sorted Array:
2 1 5 3 8 6
Process returned 0 (0x0) execution time : 0.149 s
Press any key to continue.

```

```

int part(int arr[],int low,int high)
{
    int i=low;
    int j=high;
    int pivot=arr[low];
    while(i<j)

```

```

{
while(arr[i]<=pivot && i<=high)
{
    i++;
}
while(arr[j]>pivot && j>=low)
{
    j--;
}
if(i<j)
{
    swap(arr[i],arr[j]);
}
}
swap(arr[low],arr[j]);
return j;
}

```

```

void quicksort(int arr[],int low,int high)
{
    if(low<high)
    {
        int p=part(arr,low,high);
        quicksort(arr,low,p-1);
        quicksort(arr,p+1,high);
    }
}

```

```

int main()
{
    int arr[6]={5,1,2,8,3,6};

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

    quicksort(arr,0,n-1);

    cout<<"Sorted the Array"<<endl;

    cout<<endl;

    for(int i=0;i<n-1;i=i+2)
    {
        /*cout<<arr[i]<<" ";*/

        int temp=arr[i];

        arr[i]=arr[i+1];

        arr[i+1]=temp;
    }

    cout<<"Wave sorted Array: "<<endl;

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

}

```

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.

Q1


```
Enter the value of key: 15
Key 15 is found at:
INDEX: 2
INDEX: 3
INDEX: 4

Process returned 0 (0x0)   execution time : 9.197 s
Press any key to continue.
```

```
int binarySearch(int arr[], int low, int high, int k, bool findFirst) {

    int result = -1;

    while (low <= high) {

        int mid = low + (high - low) / 2;

        if (arr[mid] == k) {

            result = mid;

            if (findFirst) {

                high = mid - 1;

            } else {

                low = mid + 1;

            }

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

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    return result;
}
```

```
}
```

```
void findAllOccurrences(int arr[], int n, int k) {  
    int first = binarySearch(arr, 0, n - 1, k, true);  
    if (first == -1) {  
        cout << "Key " << k << " is not found." << endl;  
        return;  
    }  
}
```

```
    int last = binarySearch(arr, 0, n - 1, k, false);  
    cout << "Key " << k << " is found at: " << endl;  
    for (int i = first; i <= last; ++i) {  
        if (arr[i] == k) {  
            cout << "INDEX: " << i << endl;  
        }  
    }  
}
```

```
int main() {  
    int arr[10] = {16, 31, 15, 27, 9, 15, 39, 15, 17, 12};  
    int n = sizeof(arr) / sizeof(arr[0]);  
    sort(arr, arr + n);  
    int k;  
    cout << "Enter the value of key: ";  
    cin >> k;  
    findAllOccurrences(arr, n, k);  
    return 0;  
}
```

```
}
```

Q2

```
Enter the value of product: 400
Pair Found: (5, 80)

Process returned 0 (0x0)   execution time : 33.886 s
Press any key to continue.
```

```
bool binarySearch(int arr[], int low, int high, int target) {

    while (low <= high) {

        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {

            return true;

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

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    return false;

}
```

```
void findPairWithProduct(int arr[], int n, int product) {

    sort(arr, arr + n);

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

        if (product % arr[i] == 0) {

            int complement = product / arr[i];
```

```
    if (binarySearch(arr, 0, n - 1, complement)) {  
        cout << "Pair Found: (" << arr[i] << ", " << complement << ")" << endl;  
        return;  
    }  
}  
}  
}  
cout << "No pair found with product " << product << endl;  
}
```

```
int main() {  
    int arr[] = {5, 20, 3, 2, 50, 80};  
    int n = sizeof(arr) / sizeof(arr[0]);  
    int product;  
    cout << "Enter the value of product: ";  
    cin >> product;  
    findPairWithProduct(arr, n, product);  
    return 0;  
}
```