**15B17CI371 – Data Structures Lab**

**ODD 2024**

**Week 4-LAB B**

**Practice Lab**

**[CO: C270.2]**

**MOHD TARIQ SAMI- 9923103011**

**1. Write a program to count the frequency of each element in a given array.**

**Test Case:**

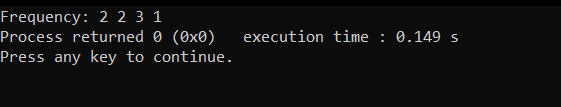
**Input:**

**array = {9, 12, 3, 31, 3, 19, 9, 3};**

**Output:**

**Unique: {9, 12, 3, 31, 19}**

**Frequency: {2, 1, 3, 1, 1}**

****

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int main()

{

int arr[8]={2,3,2,1,1,1,4,3};

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

map<int,int> m;

bool uniq=true;

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

{

if(m[arr[i]]==1) uniq=false;

m[arr[i]]++;

}

if(uniq)

{

cout<<"Unique: ";

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

{

cout<<arr[i]<<" ";

}

}

else

{

map<int,int>::iterator it=m.begin();

cout<<"Frequency: ";

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

{

if(m[arr[i]]>0)

{

cout<<m[arr[i]]<<" ";

m[arr[i]]=0;

}

}

}

}

**2. Given a sorted array of n elements, write a function to search for a given target value**

**using the Jump Search algorithm. The Jump Search algorithm works by dividing the**

**array into blocks of a fixed size, jumping ahead by these block sizes, and then**

**performing a linear search within the block where the target might be present.**

**Test Case:**

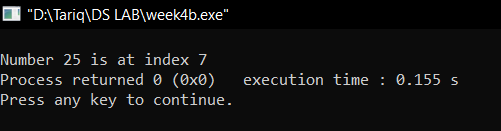
**Input:**

**array = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};**

**size = 10;**

**key = 11;**

**Output: Element found at index 5**

****

int jumpSearch(int arr[], int x, int n)

{

// Finding block size to be jumped

int step = sqrt(n);

int prev = 0;

while (arr[min(step, n)-1] < x)

{

prev = step;

step += sqrt(n);

if (prev >= n)

return -1;

}

while (arr[prev] < x)

{

prev++;

if (prev == min(step, n))

return -1;

}

if (arr[prev] == x)

return prev;

return -1;

}

int main()

{

int arr[] = {0,1,2,3,6,8,13,25,50,52,60,63};

int x = 25;

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

int index = jumpSearch(arr, x, n);

cout << "\nNumber " << x << " is at index " << index;

return 0;

}

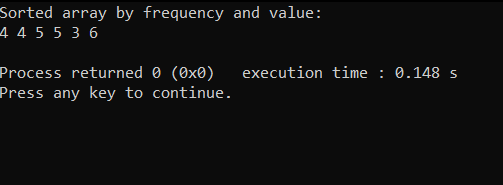
**3. Given an array of integers, sort the array according to the frequency of elements.**

**Elements with higher frequency come first. If two elements have the same frequency,**

**they are sorted by their value.**

**Input: arr[] = {4, 5, 6, 5, 4, 3}**

**Output: Pair Found: (4, 4, 5, 5, 3, 6)**

****

vector<int> sortByFrequency(vector<int>& arr) {

unordered\_map<int, int> frequencyMap;

for (int num : arr) {

frequencyMap[num]++;

}

vector<pair<int, int>> freqVec;

for (const auto& entry : frequencyMap) {

freqVec.push\_back(entry);

}

sort(freqVec.begin(), freqVec.end(), [](const pair<int, int>& a, const pair<int, int>& b) {

if (a.second == b.second) {

return a.first < b.first; // Sort by value if frequencies are the same

}

return a.second > b.second; // Sort by frequency in descending order

});

vector<int> result;

for (const auto& entry : freqVec) {

result.insert(result.end(), entry.second, entry.first);

}

return result;

}

void printArray(const vector<int>& arr) {

for (int num : arr) {

cout << num << " ";

}

cout << endl;

}

int main() {

vector<int> arr = {4, 5, 6, 5, 4, 3};

vector<int> sortedArray = sortByFrequency(arr);

cout << "Sorted array by frequency and value: " << endl;

printArray(sortedArray);

return 0;

}

**4. Given an array of integers, return an array depicting the differences between two**

**successive elements sorted in descending order.**

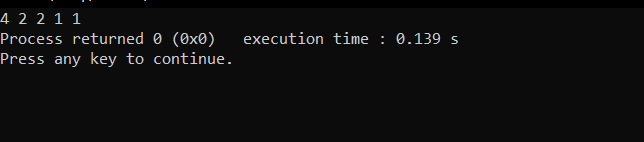
**Input: arr[] = {4, 1, 3, 5, 4, 3}**

**Output: out[] ={3, 2, 2, 1,1}**

**Input: arr[] = {3, 1, 4, 5, 4, 3}**

**Output: out[] = {3,2,1,1,1}**

**Difference array is {2, 3, 1, 1,1}, then it is sorted in descending order.**

****

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 dquicksort(int arr[],int low,int high)

{

if(low<high)

{

int p=part(arr,low,high);

dquicksort(arr,low,p-1);

dquicksort(arr,p+1,high);

}

}

int main()

{

int arr[6]={4,2,1,5,7,6};

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

vector<int> v;

int dArr[n-1];

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

{

int diff;

if(arr[i]>arr[i+1])

{

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

}

else

{

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

}

dArr[i]=diff;

}

dquicksort(dArr,0,n-1-1);

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

{

cout<<dArr[i]<<" ";

}

}