Name:Vishal Chauhan
PRN:2020BTECS00090
Course: Design and analysis of algorithm Lab
Batch:T6
Class:TYCSE,WCE

Q) Given an array *A[0…n-1]* of *n* numbers containing repetition of some number. Given an algorithmfor checking whether there are repeated element or not. Assume that we are not allowed to use additional space (i.e., we can use a few temporary variable, *O(1)* storage).

ALGORITHM→

```
// Algorithm:-->
// 1.   Traverse the given array from start to end.
// 2.   For every element in the array increment the arr[i]%n'th element by n.
// 3.   Now traverse the array again and print all those indexes i for which
arr[i]/n is greater than 1. Which guarantees that the number n has been added to
that index
// 4.   This approach works because all elements are in the range from 0 to n-1
and arr[i] would be greater than n only if a value "i" has appeared more than
once.
```

CODE→

```
// Q1) Given an array A[0…n-1] of n numbers containing repetition of some number.
Given an algorithm for checking whether there are repeated
// element or not. Assume that we are not allowed to use additional space (i.e.,
we can use a few temporary variable, O(1) storage).
// ANS==>


#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cout << "enter size of arr:";
    cin >> n;
    vector<int> arr(n);

    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
```

```
    }

    // count the frequency
    int arr_size=arr.size();
    for (int i = 0; i < arr_size; i++)
    {
        arr[arr[i] % arr_size] = arr[arr[i] % arr_size] + arr_size;
    }

    int count_repeating_ele=0;
    for (int i = 0; i < arr_size; i++)
    {
        if (arr[i] >= arr_size * 2)
        {
            count_repeating_ele++;
        }
    }
    if(count_repeating_ele>0)
    cout<<"YES,arr have contained some repetition number\n";
    else
    cout<<"No,reqpeated nuber founded\n";
    return 0;
}

OUTPUT:
enter size of arr:5
2 1 3 4 2
YES,arr have contained some repetition number
```

Complexity Analysis→

1>Time Complexity: O(n), Only two traversals are needed. So the time complexity is O(n).

2>Auxiliary Space: O(1), No extra space is needed, so the space complexity is constant.

Q2.Given an array *A[0…n-1]* , where each element of the array represents a vote in the election. Assume that each vote is given as an integer representing the ID of the chosen candidate. Given an algorithm for determining who wins the election.

```
// ALGORITHM-->
// 1.   take input array of vote
// 2.   find each frequency using map
// 3.   find max frequency from map
// 4.   print max frequency condidate
```

```cpp
// Q)   Given an array A[0…n-1] , where each element of the array represents a vote in the election. Assume that each vote is given as an
// integer representing the ID of the chosen candidate. Given an algorithm for determining who wins the election.

#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cout << "enter size of arr:";
    cin >> n;
    vector<int> arr(n);
    map<int, int> count_vote;

    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
        count_vote[arr[i]]++;
    }

    map<int, int>::iterator it;
    it = count_vote.begin();
    int max_vote = it->second;
    int winner_condidate = it->first;

    for (auto val : count_vote)
    {
        // cout<<<<"->"<<val.second<<endl;
        // break;
        if (max_vote < val.second)
        {
            max_vote = val.second;
            winner_condidate = val.first;
        }
    }
    cout << "winner is ocndidate:" << winner_condidate << endl;
}
```

```
OUTPUT:
enter size of arr:5
22 22 5 6 3
winner is ocndidate:22
```

Complexity Analysis→
1>Time Complexity: O(n), Only two traversals are needed first in arr then in map. So the time complexity is O(n).
2>Auxiliary Space: O(n), As we have used map to stored frequency of vote

Q3) Given an array *A* of *n* elements, each of which is an integer in the range *[1, n^2]*. How do we sort the array in *O(n)* time?

```
// ALGORITHM:-
// 1.   Take all the array elements as the input.
// 2.   Reduce all the elements of the array by 1
// 3.   Countsort the elements of the array two times and store the result back
into the same array.
// 4.   Add 1 to each element of the array.
// 5.   Display the sorted array.
```

Code:

```
#include <bits/stdc++.h>
#define ll long long int
using namespace std;

void countSort(vector<ll> &v, int n, int exp)
{

    int output[n];
    int i, count[n];
    for (int i = 0; i < n; i++)
        count[i] = 0;

    for (i = 0; i < n; i++)
        count[(v[i] / exp) % n]++;

    for (i = 1; i < n; i++)
        count[i] += count[i - 1];

    for (i = n - 1; i >= 0; i--)
    {
        output[count[(v[i] / exp) % n] - 1] = v[i];
        count[(v[i] / exp) % n]--;
    }
```

```
    for (i = 0; i < n; i++)
        v[i] = output[i];
}

void sort(vector<ll> &v, int n)
{
    countSort(v, n, 1);
    countSort(v, n, n);
}

void printv(vector<ll> &v, int n)
{
    for (int i = 0; i < n; i++)
        cout << v[i] << " ";
}
int main()
{

    vector<ll> v={1,6,4,8,26,15,20,9};
    int n = v.size();

    for (ll i = 0; i < n; i++)
    {
        v[i]--;
    }
    sort(v, n);
    for (ll i = 0; i < n; i++)
    {
        v[i]++;
    }
    printv(v, n);
    return 0;
}
```

```
OUTPUT:
PS C:\Users\Dell\OneDrive\Desktop\DAA\Ass-1> cd
"c:\Users\Dell\OneDrive\Desktop\DAA\Ass-1\" ; if ($?) { g++ q3.cpp -o q3 } ; if
($?) { .\q3 }
1 4 6 8 9 15 20 26
```

Complexity Analysis→

```
1>The time complexity of the above algorithm is O(N)
2>the space complexity is O(N).
```

Q4) Let *A* and *B* tow arrays of n elements, each. Given a number *K*, give an *O (nlogn*) time algorithm for determining whether there exists $a \epsilon A$ and $b \epsilon B$ such that $a+b = K$.

```
Algorithm→
1.take input array A,B of size n and K
2.traverse in loop to find K=A[i]+B[j]
3.if foound then break both loop
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;
    vector<int> A(n);
    vector<int> B(n);
    for (int i = 0; i < n; i++)
    {
        cin >> A[i];
    }
    for (int i = 0; i < n; i++)
    {
        cin >> B[i];
    }

    int K;
    cout<<"enter K:";
    cin>>K;

    int flag=1;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(A[i]+B[j]==K)
            {
                cout<<"Yes,there exit a from A and b from B such that a+b=K\n";
                cout<<"a:"<<A[i]<<"  b:"<<B[j]<<"  K:"<<K<<endl;
                flag=0;
                break;
            }
        }
        if(flag==0)
        break;
```

```
    }

    if(flag)
    cout<<"No,there is no --> a from A and b from B such that a+b=K\n";


}
```

```
OUTPUT:
5
1 3 4 5 7
2 6 7 8 9
enter K:12
Yes,there exit a from A and b from B such that a+b=K
a:3  b:9  K:12
```

Complexity Analysis→

```
Time complecity ->O(n^2)  as we have travverse twicly in for loop
Space complecity is O(1)-->as we have used only constant space only
```