

Walchand College of Engineering, Sangli  
Computer Science & Engineering  
Third Year  
Course: Design and analysis of algorithm Lab  
Lab course coordinator:  
Dr. B. F. Momin- Batch: - T6, T7, T8  
Mr. Kiran P. Kamble- Batch: - T1, T2, T3, T4, T5

## Week 2 Assignment

Part: 2

### Sorting Algorithm

**Name:** Trupti Rajendra Patil

**Prn:** 2020BTECS00051

Q) Given an array  $A[0 \dots n-1]$  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).

**Algorithm:**

1. Traverse the given array from start to end.
2. For every element in the array increment the  $\text{arr}[i] \% n$ 'th element by  $n$ .
3. Now traverse the array again and print all those indexes  $i$  for which  $\text{arr}[i]/n$  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  $\text{arr}[i]$  would be greater than  $n$  only if a value " $i$ " has appeared more than once.

**Code:**

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin>>n;
```

```

int arr[n];
for(int i=0;i<n;i++){
    cin>>arr[i];
}
// 1 6 3 1 3 6 6
for(int i=0;i<n;i++){
    if(arr[arr[i]%n]>0){
        arr[arr[i]%n]=arr[arr[i]%n]+n;
    }
}

bool ans=false;
for(int i=0;i<n;i++){
    if(arr[i]>=n*2){
        // cout<<i<<" ";
        ans=true;
    }
}cout<<endl;

if(ans==true){
    cout<<"Repeated elements present in array"<<endl;
}else{
    cout<<"Repeated elements not present in array"<<endl;
}
}

```

### Complexity Analysis:

Time complexity:  $O(n)$

Space complexity:  $O(1)$

### Output:

```

PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Us
f ($?) { g++ A2Q1.cpp -o A2Q1 } ; if ($?) { .\A2Q1 }
7
1 6 3 1 3 6 6

Repeated elements present in array
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> █

```

Q) Given an array  $A[0 \dots n-1]$ , 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. Create unordered\_map m
2. Store frequency of voters
3. Search for maximum votes that is the winner.

**Code:**

```
#include<bits/stdc++.h>
using namespace std;
//0 0 1 2 4 4 3 3 3 0 1 1
int main(){
    int n;
    cin>>n;

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

    unordered_map<int,int>m;
    for(int i=0;i<n;i++){
        m[arr[i]]++;
    }

    int maxVotes=INT_MIN, winner=INT_MIN;
    for(auto it: m){
        if(it.second > maxVotes){
            maxVotes=it.second;
            winner=it.first;
        }
    }

    cout<<"Winner of election : "<<winner<<endl;
    cout<<"Maximum votes : "<<maxVotes<<endl;
}
```

### Complexity Analysis:

Time complexity:  $O(n)$

Space Complexity:  $O(n)$

### Output:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ"
f ($?) { g++ A2Q2.cpp -o A2Q2 } ; if ($?) { .\A2Q2 }
13
0 0 1 2 4 4 3 3 3 0 1 1
Winner of election : 3
Maximum votes : 4
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

Q) 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?

We can sort elements by using radix sort in  $O(n)$  time with some constant space. Here we can also use count sort as it has  $O(n)$  time complexity but it will consume  $O(n^2)$  space, so Radix sort is better than count sort.

### Algorithm:

1. Find the maximum element of the array, let it be  $\text{max}$
2. Find the number of digits in  $\text{max}$  let it be  $k$
3. For each,  $i$  ranging from 1 to  $k$ , apply the counting sort algorithm for the  $i$ th least significant digit of each element. If any element has less than  $i$  digits consider 0 at its place

### Code:

```
#include<iostream>
using namespace std;

int getMax(int arr[],int n){
    int k=arr[0];
    for(int i=0;i<n;i++){
        k=max(k,arr[i]);
    }
    return k;
}
```

```

}

void countSort(int arr[],int n,int pos){
    int output[n],i;
    int count[10]={0};

    for(i=0;i<n;i++){
        count[(arr[i]/pos)%10]++;
    }

    for(i=1;i<10;i++){
        count[i]+=count[i-1];
    }

    for(i=n-1;i>=0;i--){
        output[--count[(arr[i]/pos)%10]]=arr[i];
    }

    for(i=0;i<n;i++){
        arr[i]=output[i];
    }
}

void radixSort(int arr[],int n){
    int m=getMax(arr,n);

    for(int pos=1;m/pos>0;pos*=10){
        countSort(arr,n,pos);
    }
}

int main(){
    int n;
    cin>>n;

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

```

```

    }

    radixSort(arr,n);

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

```

### Complexity Analysis:

Time Complexity:  $O(n)$

Space Complexity:  $O(n)$

### Output:

```

PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ"
f ($?) { g++ A2Q3.cpp -o A2Q3 } ; if ($?) { .\A2Q3 }
7
23 906 54 7 16 88 12
7 12 16 23 54 88 906
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>

```

Q) Let  $A$  and  $B$  two arrays of  $n$  elements, each. Given a number  $K$ , give an  $O(n \log n)$  time algorithm for determining whether there exists  $a \in A$  and  $b \in B$  such that  $a+b = K$ .

### Algorithm:

1. Sort the given two arrays using sorting algorithm of time complexity  $O(n \log n)$
2. Traverse array  $A$  and search for element  $K-A[i]$  in array  $B$ , using Binary search.
3. If element found then return true, else search for other pair of elements.

### HeapSort:

1. Build a max heap from the input data.
2. At this point, the maximum element is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of the heap by 1. Finally, heapify the root of the tree.
3. Repeat step 3 while the size of the heap is greater than 1.

Code:

```
#include<bits/stdc++.h>
using namespace std;

void heapify(int arr[],int n,int i){
    int largest=i;
    int l=2*i;
    int r=2*i+1;

    if(l<n && arr[l]>arr[largest])
        largest=l;
    if(r<n && arr[r]>arr[largest])
        largest=r;

    if(largest!=i){
        swap(arr[i],arr[largest]);

        heapify(arr,n,largest);
    }
}

void heapSort(int arr[],int n){
    for(int i=n/2-1;i>=0;i--){
        heapify(arr,n,i);
    }

    for(int i=n-1;i>0;i--){
        swap(arr[0],arr[i]);

        heapify(arr,i,0);
    }
}

int binarySearch(int arr[],int n,int key){
    int l=0,r=n;
    while(l<=r){
        int mid=(l+r)/2;
```

```

        if(arr[mid]==key){
            return mid;
        }else if(arr[mid]>key){
            r=mid-1;
        }else{
            l=mid+1;
        }
    }
}

int main(){
    int n=5;
    int a[5]={2,1,4,6,3};
    int b[5]={3,1,5,2,6};

    int x=5;

    heapSort(a,n);
    heapSort(b,n);

    int index=-1;
    for(int i=0;i<n;i++){
        index=binarySearch(b,n,abs(a[i]-x));
        if(index!=-1){
            break;
        }
    }

    if(index!=-1){
        cout<<"Pair a,b such that a+b=k is present."<<endl;
    }else{
        cout<<"Pair a,b such that a+b=k is not present."<<endl;
    }
}

```



### Complexity Analysis:

Time Complexity:  $O(n \log n)$

Space Complexity:  $O(1)$

### Output:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Use
f ($?) { g++ A2Q4.cpp -o A2Q4 } ; if ($?) { .\A2Q4 }
Pair a,b such that a+b=k is present.
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```