**Assignment-49: A Job Ready Bootcamp in c++,DSA and IOT**

**unordered\_set**

1. Count of distinct pair sums in a given Array arr[] of size N, the task is to find the total

number of unique pair sums possible from the array elements.

Sol – 1.

#include<iostream>

#include<unordered\_set>

#include<vector>

using namespace std;

int cntDisPairs(vector<int>arr,int target)

{

unordered\_set<int>set;

unordered\_set<int>seen;

int count=0;

for(int num : arr)

{

if(set.find(target-num)!=set.end()&&seen.find(num)==seen.end())

{

count++;

seen.insert(num);

seen.insert(target-num);

}

set.insert(num);

}

return count;

}

int main()

{

vector<int>arr={1,5,3,3,1,5};

int k=6;

cout<<cntDisPairs(arr,k);

return 0;

}

2. C++ Program to Print all triplets in sorted array that form AP(or Arithmetic

Progression) Example..Input : arr[] = { 2, 6, 9, 12, 17, 22, 31, 32, 35, 42 };

Output :

6 9 12

2 12 22

12 17 22

2 17 32

12 22 32

9 22 35

2 22 42

22 32 42

Sol – 2.

#include<bits/stdc++.h>

using namespace std;

void printAllTriplet(int a[],int n)

{

unordered\_set<int>s;

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

{

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

{

int diff=a[j]-a[i];

if(s.find(a[i]-diff)!=s.end())

cout<<a[i]-diff<<" "<<a[i]<<" "<<a[j]<<endl;

}

s.insert(a[i]);

}

}

int main()

{

int a[]={2,6,9,12,17,22,31,32,35,42};

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

printAllTriplet(a,n);

return 0;

}

3. C++ Program for Number of unique triplets whose XOR is zero.

Input : a[] = {1, 3, 5, 10, 14, 15};

Output : 2

Explanation : {1, 14, 15} and {5, 10, 15} are the

unique triplets whose XOR is 0.

{1, 14, 15} and all other combinations of

1, 14, 15 are considered as 1 only.

Input : a[] = {4, 7, 5, 8, 3, 9};

Output : 1

Explanation : {4, 7, 3} is the only triplet whose XOR is 0

Sol – 3.

#include<bits/stdc++.h>

using namespace std;

int countTriplets(int a[],int n)

{

unordered\_set<int>s;

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

{

s.insert(a[i]);

}

int count=0;

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

{

for(int j=i+1;j<n;j++)

{

int xr=a[i]^a[j];

if(s.find(xr)!=s.end()&&xr!=a[i]&&xr!=a[j])

{

count++;

}

}

}

return count/3;

}

int main()

{

int a[]={1,3,5,10,14,15};

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

cout<<countTriplets(a,n);

return 0;

}

4. C++ Program to give two arrays with size n, maximise the first array by using the

elements from the second array such that the new array formed contains n greatest

but unique elements of both the arrays giving the second array priority (All elements

of second array appear before first array). The order of appearance of elements is

kept the same in output as in input.

Examples:

Input : arr1[] = {2, 4, 3}

arr2[] = {5, 6, 1}

Output : 5 6 4

As 5, 6 and 4 are maximum elements from two arrays giving the second array

higher priority. Order of elements is the same in output as in input.

Input : arr1[] = {7, 4, 8, 0, 1}

arr2[] = {9, 7, 2, 3, 6}

Output : 9 7 6 4 8

Sol – 4.

#include<bits/stdc++.h>

#include<string>

using namespace std;

bool compare(int a,int b)

{

return a>b;

}

void maximizeArray(int a1[],int a2[],int n)

{

int a3[2\*n],k=0;

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

{

a3[k++]=a1[i];

}

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

{

a3[k++]=a2[i];

}

unordered\_set<int>hash;

sort(a3,a3+2\*n,compare);

int i=0;

while(hash.size()!=n)

{

if(hash.find(a3[i])==hash.end())

hash.insert(a3[i]);

i++;

}

k=0;

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

{

if(hash.find(a2[i])!=hash.end())

{

a3[k++]=a2[i];

hash.erase(a2[i]);

}

}

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

{

if(hash.find(a1[i])!=hash.end())

{

a3[k++]=a1[i];

hash.erase(a1[i]);

}

}

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

{

a1[i]=a3[i];

}

}

void printArray(int a[],int n)

{

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

{

cout<<a[i]<<" ";

}

cout<<endl;

}

int main()

{

int a[]={7,4,8,0,1};

int b[]={9,7,2,3,6};

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

maximizeArray(a,b,n);

printArray(a,n);

return 0;

}

5. C++ Program to given an array of positive and negative numbers, find if there is a

subarray (of size at-least one) with 0 sum.

Examples :

Input: {4, 2, -3, 1, 6}

Output: true

Explanation:

There is a subarray with zero sum from index 1 to 3.

Input: {4, 2, 0, 1, 6}

Output: true

Explanation:

There is a subarray with zero sum from index 2 to 2.

Sol – 5.

#include<bits/stdc++.h>

using namespace std;

bool subArrayExists(int a[],int n)

{

unordered\_set<int>sumSet;

int sum=0;

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

{

sum+=a[i];

if(sum==0||sumSet.find(sum)!=sumSet.end())

{

return true;

}

sumSet.insert(sum);

}

return false;

}

int main()

{

int a[]={4,2,-3,1,6};

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

if(subArrayExists(a,n))

cout<<"Found a subarray with 0 sum"<<endl;

else

cout<<"No such subarray exists!"<<endl;

return 0;

}

6. Given an array arr[] consisting of N positive integers, the task is to find the number of

pairs such that the Greatest Common Divisor(GCD) of the pairs is not a prime

number. The pair (i, j) and (j, i) are considered the same.

Examples:

Input: arr[] ={ 2, 3, 9}

Output: 10

Explanation:

Following are the possible pairs whose GCD is not prime:

(0, 1): The GCD of arr[0](= 2) and arr[1](= 3) is 1.

(0, 2): The GCD of arr[0](= 2) and arr[2](= 9) is 1.

Therefore, the total count of pairs is 2.

Input: arr[] = {3, 5, 2, 10}

Output: 4

Sol – 6.

#include<bits/stdc++.h>

using namespace std;

int gcd(int a,int b)

{

int n=(a<=b?a:b),hcf=1;

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

{

if(a%i==0&&b%i==0)

{

hcf=i;

}

}

return hcf;

}

bool isPrime(int n)

{

if(n==1)

return false;

if(n==2)

return true;

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

{

if(n%i==0)

{

return false;

}

}

return true;

}

int noOfPair(int a[],int n)

{

int count=0;

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

{

for(int j=i+1;j<n;j++)

{

if(!isPrime(gcd(a[i],a[j])))

count++;

}

}

return count;

}

int main()

{

int a[]={2,3,9};

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

cout<<"No. of pair : "<<noOfPair(a,n)<<endl;

return 0;

}

7. Given an array of strings arr[] of size N, the task is to print all the distinct strings

present in the given array.

Examples:

Input: arr[] = { “Good”, “God”, “Good”, “God”, “god” }

Output: god Good God

Sol – 7.

#include<bits/stdc++.h>

using namespace std;

void Printuniq(string a[],int n)

{

unordered\_set<string>s;

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

{

s.insert(a[i]);

}

for(auto it:s)

{

cout<<it<<" ";

}

cout<<endl;

}

int main()

{

string a[]={"Good","God","Good","God","god"};

int n=5;

Printuniq(a,n);

return 0;

}

8. Find all matrix elements which are minimum in their row and maximum in their

Column

Sol – 8.

#include<bits/stdc++.h>

using namespace std;

vector<int> minmaxNumbers(vector<vector<int>>&matrix,vector<int>&res)

{

unordered\_set<int>set;

for(int i=0;i<matrix.size();i++)

{

int minr=INT\_MAX;

for(int j=0;j<matrix[i].size();j++)

{

minr=minr<matrix[i][j]?minr:matrix[i][j];

}

set.insert(minr);

}

for(int j=0;j<matrix[0].size();j++)

{

int maxc=INT\_MIN;

for(int i=0;i<matrix.size();i++)

{

maxc=maxc>matrix[i][j]?maxc:matrix[i][j];

}

if(set.find(maxc)!=set.end())

res.push\_back(maxc);

}

return res;

}

int main()

{

vector<vector<int>>mat={{1,10,4},{9,3,8},{15,16,17}};

vector<int>ans;

minmaxNumbers(mat,ans);

if(ans.size()==0)

cout<<"-1"<<endl;

for(int i=0;i<ans.size();i++)

cout<<ans[i]<<endl;

return 0;

}

9. Given N strings of equal lengths. The strings contain only digits (1 to 9). The task is

to count the number of strings that have an index position such that the digit at this

index position is greater than the digits at the same index position of all the other

strings.

Examples:

Input: arr[] = {“223”, “232”, “112”}

Output: 2

First digit of the 1st and 2nd strings are the largest.

Second digit of the string 2nd is the largest.

Third digit of the string 1st is the largest.

Input: arr[] = {“999”, “122”, “111”}

Output: 1

Sol – 9.

#include<bits/stdc++.h>

using namespace std;

int countStrings(int n,int m,string s[])

{

unordered\_set<int>ind;

for(int j=0;j<m;j++)

{

int mx=0;

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

{

mx=mx>(int)s[i][j]-'0'?mx:(int)s[i][j]-'0';

}

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

{

if(s[i][j]-'0'==mx)

ind.insert(i);

}

}

return ind.size();

}

int main()

{

string s[]={"223","232","112"};

int m=s[0].length();

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

cout<<countStrings(n,m,s);

return 0;

}

10. Unordered\_set operators in C++ STL(== and !=)

Sol – 10.

#include<bits/stdc++.h>

using namespace std;

int main()

{

unordered\_set<int>s1={10,20,30,40,50,40};

unordered\_set<int>s2={10,30,50,40,20};

unordered\_set<int>s3={10,20,30,50,60};

if(s1==s2)

cout<<"S1 and S2 are equal"<<endl;

else

cout<<"S1 and S2 are not equal"<<endl;

if(s2==s3)

cout<<"S2 and S3 are equal"<<endl;

else

cout<<"S2 and S3 are not equal"<<endl;

return 0;

}