<https://github.com/sakibchonka/cpp_learning/tree/main/Assignment3>

Q1.

Given two arrays: a1[0..n-1] of size n and a2[0..m-1] of size m. Task is to check whether a2[] is a subset of a1[] or not.

Both the arrays can be sorted or unsorted. It may be assumed that elements in both array are distinct.

Example 1:

Input:

a1[] = {11, 1, 13, 21, 3, 7}

a2[] = {11, 3, 7, 1}

Output:

Yes

Explanation:

a2[] is a subset of a1[]

#include<bits/stdc++.h>

using namespace std;

bool isSubset(int \*a1, int \*a2, int n, int m){

    if(m>n) return false;

    sort(a1,a1+n);

    sort(a2,a2+m);

    for(int i=0;i<m;i++){

        if(a2[i]!=a1[i])

            return false;

    }

    return true;

}

int main(){

    int n,m;

    cin>>n>>m;

    int a1[n];

    int a2[m];

    for(int i=0;i<n;i++){

        cin>>a1[i];

    }

    for(int i=0;i<m;i++){

        cin>>a2[i];

    }

    cout<<isSubset(a1,a2,n,m);

}

Q2.

Given an array arr of size n and an integer X. Find if there's a triplet in the array which sums up to the given integer X.

Example 1:

Input:

n = 6, X = 13

arr[] = [1 4 45 6 10 8]

Output:

1

Explanation:

The triplet {1, 4, 8} in

the array sums up to 13.

#include<bits/stdc++.h>

using namespace std;

bool tripletSum(int \*arr,int n,int X){

    for(int i=0; i<n-2; i++){

        for(int j=i+1; j<n-1; j++){

            for(int k=j+1; k<n; k++){

                if(arr[i]+arr[j]+arr[k]==X)

                    return true;

            }

        }

    }

    return false;

}

int main(){

    int n,X;

    cin>>n>>X;

    int arr[n];

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    cout<<tripletSum(arr,n,X);

}

Q3.

A conveyor belt has packages that must be shipped from one port to another within days days.

The ith package on the conveyor belt has a weight of weights[i]. Each day, we load the ship with packages on the conveyor

belt (in the order given by weights).

We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within days days.

Example 1:

Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 5

Output: 15

Explanation: A ship capacity of 15 is the minimum to ship all the packages in 5 days like this:

1st day: 1, 2, 3, 4, 5

2nd day: 6, 7

3rd day: 8

4th day: 9

5th day: 10

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

#include<bits/stdc++.h>

using namespace std;

int findMax(int \*a,int n){

    int max=a[0];

    for(int i=1;i<n;i++){

        if(a[i]>max) max=a[i];

    }

    return max;

}

// Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 5

// Output: 15

int minCapacity(int \*weights, int n, int days){

    int min\_cap = findMax(weights,n);

    while(true){

        int sum=0;

        int d=1;

        for(int i=0;i<n;i++){

            if(d>days){

                break;

            }

            if(sum+weights[i]<=min\_cap){

                sum+=weights[i];

            }else{

                sum=weights[i];

                d++;

            }

        }

        if(d==days)

            return min\_cap;

        else

            min\_cap++;

    }

}

int main(){

    int n, days;

    cin>>n>>days;

    int weights[n];

    for(int i=0;i<n;i++){

        cin>>weights[i];

    }

    cout<<minCapacity(weights,n,days);

}