Media.NET 2023

Q1 – Infinity Stones AvPpg

The mighty Thanos is obsessed with destroying the universe. But in order to do that he has to collect K Infinity Stones

Thelos is tired of fighting. So he decides to behave civilized this time please stop as you are his most trusted ally.

He gives you the task to bring him Infinity Stones. In order to make an infinity stone we need to buy the corresponding nanocrystals. Nano crystals can be combined to make infinity stone of some size in simple words if you want to make the ith infinity stone of size z and you have the size of Ayeth Nanocrystal P then amount of nano crystals required equals to square bracket z upon P.

Nenu crystal 4 one particular infinity stone cannot be used to make another infinity stone. Each of the given K Infinity Stones should be of the same size and as big as possible.

There are there are N shops A1, A2…..An where You can buy these infinity stones. Each shop a sells the Jth nanocrystal of size Ai,j . you can only take integral amount of stones from any shop.

But since the avengers are coming You will only have the time to visit any 2 shops. You can buy any type of nanocrystal in any amount from these 2 shops each Crystal is each crystal of any type from any shop cost 1 dollar since fighting the war is financially challenging thanos only gave you S dollars find the maximum size of all the infinity stones you can make since the answer can be large output module 1e9+7.

Problem constraints

1<=N<=10^5

1<=k<=7

1<=A[i][j]<=10^5

1<=A[i][j]<=10^3

1<=S<=10^3

Input format

First argument contains a 2D array A of size N*K

Second argument contains an integer S

Input 1

A = [[9,4,1],

[2,5,10],

[6,8,3]]

S = 10

Input 2:

A = [[4,1],

[1,4],]

S = 10

Output 1

24

Output 2

20

Explanation 1:

We can use shop 2 and 3 … total cost for size = 24. 24/6+24/8+ceil(24/10) => 4+3+3 =10

We can prove that it is the optimal answer

Explanation 2:

We will use shop1 and shop2

We can prove that it is the optimal answer

```cpp
#define MOD 1000000007

bool is_possible(int size, int shop1, int shop2, int S, int K, vector<vector<int>>& A) {
    int cost = 0;
    for (int i = 0; i < A[0].size(); i++) {
        cost += ceil((double)size / min(A[shop1][i], A[shop2][i]));
    }
    return cost * K <= S;
}

int max_infinity_stone_size(vector<vector<int>>& A, int S) {
    int N = A.size();
    int K = A[0].size();
    int max_size = 0;

    for(int shop1 = 0; shop1 < N; shop1++) {
        for(int shop2 = shop1; shop2 < N; shop2++) {
            int low = 1, high = *min_element(A[shop1].begin(), A[shop1].end()) * S;
            while(low <= high) {
                int mid = (low + high) / 2;
                if(is_possible(mid, shop1, shop2, S, K, A)) {
                    max_size = max(max_size, mid);
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            }
        }
    }
    return max_size;
}

int main() {
    vector<vector<int>> A = {{9, 4, 1}, {2, 5, 10}, {6, 8, 3}};
    int S = 10;

    cout << "Maximum size of infinity stones: " << max_infinity_stone_size(A, S) << endl;
```

Sol -2 -→

```cpp
int Solution(vector<vector<ll>> A, ll S)
{
    ll l = 0;
    ll r = 1e18;
    ll maxSize = -1;
    while (r - l >= 0)
    {
        ll mid = (l + r) / 2;
        if (helper(mid, A, S))
        {
            l = mid + 1;
            maxSize = mid;
        }
        else {
            r = mid - 1;
        }
    }

    return maxSize;
}
void solve()
{
    vector<vector<ll>> A = {{9, 4, 1}, {2, 5, 10}, {6, 8, 3}};
    ll S = 10;
    // vector<vector<ll>> A = {{1, 4}, {4, 1}};
    // ll S = 10;
    cout << Solution(A, S) << endl;
}
int main()
{
    solve();
}
```

```cpp
bool helper(ll P, vector<vector<ll>> &A, ll S)
{
    ll k = A[0].size();
    vector<ll> minMaskCost((1 << k), 1e18);
    ll n = A.size();

    for (ll i = 0; i < n; i++)
    {
        for (ll mask = 0; mask < (1 << k); mask++)
        {
            ll cost = 0;
            for (ll j = 0; j < k; j++)
            {
                if ((mask >> j) & 1)
                {
                    cost += (P + A[i][j] - 1) / (A[i][j]);
                }
            }
            minMaskCost[mask] = min(minMaskCost[mask], cost);
        }
    }
    ll minCost = INT64_MAX;
    for (ll mask = 0; mask < (1 << k); mask++)
    {
        ll invertmask = 0;
        for (ll j = 0; j < k; j++)
        {
            if (!((mask >> j) & 1))
            {
                invertmask += (1 << j);
            }
        }
        minCost = min(minCost, minMaskCost[mask] + minMaskCost[invertmask]);
    }

    if (minCost <= S)return 1;
    else return 0;
}
int Solution(vector<vector<ll>> A, ll S)
```

Q2 – Mask Update

Bitmasks are cool. A Bitmask is a string of binary bits (0s and 1s). For example: “01010” is a bitmask.

Kuldeep is a naughty but brilliant computer scientist. In his free time, he writes some random programs to play with bitmask. He has a phd Student under him and to test him (and entertain himself). He has given him the following task:

Given a number N. Write a bitmask of length N containing all 0s. Now you are given Q Operations. Each operation contains two number (l, r) as input. An operation can be one of the following

Number One Update Operation: take the XOR of all the bits in the bitmask from index l to r both inclusive with one

Number two query operations: count the number of sad bits in the bitmask between index L2 are both inclusive

You need to find the sum of all the queries

Note:

Pin case there are no queries return zero.

As the answer can be large output the answer mod 1000000007.

Consider 0 based indexing.

Constraints:

1<=N<=100000

1<=Q<=100000

0<=l, r<N

Input format

Int A: Number N as described above

Vector<vector<int>> B: Q operations as described above

Each row contains three number (type, l, r). if type is 0 , its an update operation else a quey operation.

```
int query(int node, int start, int end, int l, int r) {
    if(start > r || end < l) {
        return 0;
    }

    if(lazy[node] != 0) {
        tree[node] = (end - start + 1) - tree[node]; // Flip the bits
        if(start != end) { // propagate to the children if they exist
            lazy[node*2] ^= 1;
            lazy[node*2+1] ^= 1;
        }
        lazy[node] = 0; // Reset
    }

    if(start >= l && end <= r) {
        return tree[node];
    }

    int mid = (start + end) / 2;
    return query(node*2, start, mid, l, r) + query(node*2+1, mid+1, end, l, r);
}

int main() {
    int N, Q;
    cin >> N >> Q;

    tree.assign(4*N, 0);
    lazy.assign(4*N, 0);

    int result = 0;

    while(Q--) {
        int type, l, r;
        cin >> type >> l >> r;

        if(type == 0) {
            update(1, 0, N-1, l, r);
        } else {
            result = (result + query(1, 0, N-1, l, r)) % MOD;
        }
    }

    cout << result << endl;

    return 0;
```

```
using namespace std;
#define MOD 1000000007

vector<int> tree, lazy;

void update(int node, int start, int end, int l, int r) {
    if(lazy[node] != 0) {
        tree[node] = (end - start + 1) - tree[node]; // F
        if(start != end) { // propagate to the children i
            lazy[node*2] ^= 1;
            lazy[node*2+1] ^= 1;
        }
        lazy[node] = 0; // Reset
    }

    if(start > r || end < l) {
        return;
    }

    if(start >= l && end <= r) {
        tree[node] = (end - start + 1) - tree[node]; // F
        if(start != end) { // propagate to the children i
            lazy[node*2] ^= 1;
            lazy[node*2+1] ^= 1;
        }
        return;
    }

    int mid = (start + end) / 2;
    update(node*2, start, mid, l, r);
    update(node*2+1, mid+1, end, l, r);
    tree[node] = tree[node*2] + tree[node*2+1]; // Merge
}
```

Q3 – Game of Ways

Batman is about to take off from Gotham's airport which has am runways (numbered from 1 to M) of length n units

As always, the Joker has come up with an insane game to have fun with Batman

The rules of the game are as follows

- Batman plane can take off only after running for an units distance in the runways.
- two Batman can start on any runway and end on any runway.
- Batman can switch his plane from Ranveer i to j only if i and j are co prime.
- If the Batman fails to switch his plane to a coprime runway for 1 unit distance on a single runway. The Joker will bomb the plane.

The Joker does not want to kill batman, because what will he do without him . So he asks for your help to find out number of ways in which Batman can take off his plane without getting bombed.

As the answer can be very large , output answer modulo (1000000007)

Input format

First argument is given is an Integer A , length of the runway.

Second argument given is an Integer B , Number of different runways available

Output format

Return single integer X , the number of ways Batman can take off his plane without getting bombed.

As X can be very large , return X MOD 10^9+7

Constraints

1<=A<=1000000000

0<=B<=10

For example

Input 1

A = 1

B = 3

Output 1

3

Input 2

A = 2

B = 3

Output 2

7

Explanation:

For test 1:

3 ways starting at 1,2,3

For test 2:

1st way : starting and covering whole distance at runway 1. i.e 1->1

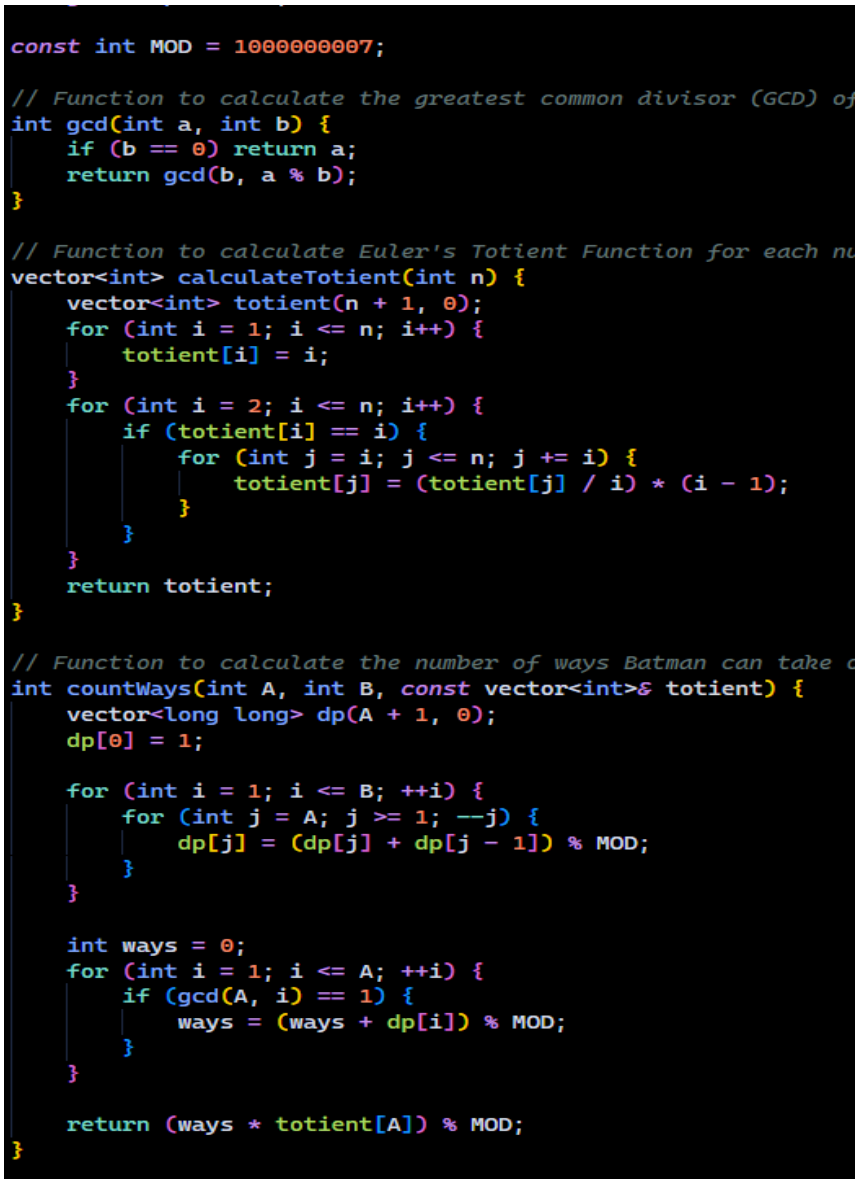
(1 and 1 are co-prime so Batman can continue on runway 1 without getting bombed)

2nd way : starting and covering distance of 1 at runway 1 and covering remaining distance at runway 2. i.e 1->2

3rd way : starting and covering distance of 1 at runway 1 and covering remaining distance at runway 3. i.e 1->3

Similarly there are 4 more ways i.e 2->1, 2->3 , 3->1 , 3->2

We can't go from 2->2 and 3 -> 3 as per given rules.



```cpp
const int MOD = 1000000007;

// Function to calculate the greatest common divisor (GCD) of
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

// Function to calculate Euler's Totient Function for each nu
vector<int> calculateTotient(int n) {
    vector<int> totient(n + 1, 0);
    for (int i = 1; i <= n; i++) {
        totient[i] = i;
    }
    for (int i = 2; i <= n; i++) {
        if (totient[i] == i) {
            for (int j = i; j <= n; j += i) {
                totient[j] = (totient[j] / i) * (i - 1);
            }
        }
    }
    return totient;
}

// Function to calculate the number of ways Batman can take o
int countWays(int A, int B, const vector<int>& totient) {
    vector<long long> dp(A + 1, 0);
    dp[0] = 1;

    for (int i = 1; i <= B; ++i) {
        for (int j = A; j >= 1; --j) {
            dp[j] = (dp[j] + dp[j - 1]) % MOD;
        }
    }

    int ways = 0;
    for (int i = 1; i <= A; ++i) {
        if (gcd(A, i) == 1) {
            ways = (ways + dp[i]) % MOD;
        }
    }

    return (ways * totient[A]) % MOD;
}
```

Another Campus

Q1 – **Vanish the Gems**

In front of you there is a treasure containing many gems. Soon you realize that they are not precious as they seem in fact they are cursed with black magic and bring bad luck with them to you try to minimize the bad luck you get

You have the power to destroy one gem in one second. You can destroy gems in any order

You are given a 2D array a where for i'th gem it takes A[i][0] seconds to give A[i][1] units of bad luck at any instant only one jam can give you the bad luck and you can decide the order in which you want to receive the bad luck. Once the receiving of bad luck starts from any gem it cannot be destroyed and it automatically vanishes after giving AI1 units of bad luck. Although while receiving bad luck from any gem you can choose to destroy any other than gem which is available in the treasure. There is no time lapse in between receiving of bad luck from any two gems as soon as 1 jam finishes giving bad luck you have to choose 2^{nd} gem from the treasure to receive that bad luck

Return the maximum amount of bad luck that you can avoid slash destroy

Note: Receiving bad luck is a continuous process. And at any instant you cannot receive bad luck from more than one gem. if you are receiving bad luck from any gem then you can choose second gem only after A[i][0] seconds.

Problem constraints

1<=|A|<=2*10^3

|A[i]| ==2

0<=A[i][0]<=2*10^3

1<=A[i][1]<=10^6

Input format

First argument is 2D array A containing information about each gem

Output format

Return single integer the maximum amount of bad luck destroyed .

Inout 1

A = [[0,1],

[0,10]]

Input 2

A = [[0,10],

[1,4],

[1,20]]

Output 1

0

Output 2

30

Q2- **Loud Speaker**

You have a cities and a Arab size M to the power 2 where BI0 and BI1 represent there is a route between these cities.

Line there is another array seek containing list of all cities having loudspeakers. A speaker's voice can propagate bracket travel through only roads. All the loud speakers have some instant intensities associated with them and its value is similar for all of them if the intensity of a speaker is X and its voice can be heard to all the other cities without shortest distance from this city is less than or equal to x.

A loudspeaker gets automatically started if it hears voice from any other loudspeaker . You are given a source city D and a destination city E and there will be always a speaker at source city which someone is starting manually . you have to tell the minimum intensity of the loudspeaker so that the voice reaches to the destination city. See example for illustrations.

Return -1 if it is impossible

Problem constraints

1<=A<=10^5

0<=|B|<=10^5

1<=B[i][0],B[i][1]<=|A|

1<=|C|<=10^5

1<=C[i]<=|A|

1<=D<=10^5

1<=E<=10^5

Input format

First argument is an integer A.

Second argument is an integer 2-D array B.

Third argument is an integer array C.

Fourth argument is an integer D.

Fifth argument is an integer E.

Output format

You have to return an intefer as per the question.

Example input

Input 1

A =0

B=[[1 2][2 3][3 4][4 5][5 6]]

C = [1 3]

D =1

E =6

Input 2

A=5

B=[[1 2][2 3][2 4][4 5]]

C= [2 3 4]

D =3

E =5

Input 3

A =2

B=[]

C =[1 2]

D=1

E=2

Q3 – **Required Substring**

Given an array of strings A of size N and two integers B and C.

Let D be the number of strings of length C that contain exactly B of the strings in words as substring.

Return D Modulo (10^9+9).

Note: Your solution will run on multiple test cases. If you're using global variables. Make sure to clear them.

Input format

The first argument given is the string array A.

The second argument given is integer B.

The third argument given is integer C.

Output format

Return the value of D modulo (10^9+9)

Constraints

1<=N<=6

1<=|A[i]|<=50

Each string consist of lowercase English alphabet (‘a’ –‘z’).

All the N strings are distinct.

0<=B<=N

1<=L<=50

Input 1

A = [“z”,”zz”,”zzz”,”zzzz”]

B=2

C=3

Ouput 1

50

The only valid strings re strings from “#zz” or “zz#”, where # is one of the letters ‘a’ – “x” .

Input 2

A = [“uvwxyz”]

B=1

C=5

Output 2

0

A string of length 5 cannot have a substring of length 6.