Infosys Coding Exam practice Questions

Infosys coding Prepration

Question 1

The Gift

John is trying to send a gift to his friend **Dave** for his birthday through his relatives. So, he will first send the gift to his relative, and then that relative will send it to **their relative**, and this process continues until it reaches **Dave**.

You are given:

- The **number of people** in the city
- The **relationships** between them

Your task is to **help John find and return the minimum number of relatives involved** in sending the gift from **John to Dave**.

If the gift is **not possible** to reach Dave through his relatives, then return **-1**.

Input Specification

input1: An integer value representing the number of people in the city
input2: An integer value representing the number of relationships between people
input3: A 2D integer array of size input2 × 2, representing the relationship between two relatives
input4: An integer value representing John in the given array
input5: An integer value representing Dave in the given array

Output Specification

Return the **minimum number of relatives involved** in sending the gift from John to his friend Dave.
If the gift is not possible to reach Dave through his relatives, return **-1**.

Example 1

input1: 7
input2: 6
input3: {{1,2}, {2,4}, {1,3}, {1,7}, {7,5}, {7,6}}
input4: 4
input5: 6

https://saurabhjha.live/study_material

Infosys Coding Exam practice Questions

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

int minimumRelatives(
    int n,
    int m,
    vector<vector<int>>& relations,
    int john,
    int dave
) {
    vector<vector<int>> graph(n + 1);

    for (auto &r : relations) {
        graph[r[0]].push_back(r[1]);
        graph[r[1]].push_back(r[0]);
    }

    vector<bool> visited(n + 1, false);
    queue<pair<int,int>> q;

    q.push({john, 0});
    visited[john] = true;

    while (!q.empty()) {
        auto [person, dist] = q.front();
        q.pop();

        if (person == dave) {
            return max(0, dist - 1); // exclude John & Dave
        }

        for (int nei : graph[person]) {
            if (!visited[nei]) {
                visited[nei] = true;
                q.push({nei, dist + 1});
            }
        }
    }

    return -1;
}
```

Question 2

You are given a sequence **A** containing **N positive integers**.
The task is to calculate the **special value** of the sequence.

Definition of Special Value

The **special value** of a sequence is calculated as follows:

- For **every possible subarray** of the sequence, identify: the **largest** number in that subarray the **smallest** number in that subarray

https://saurabhjha.live/study_material

- Multiply the **largest** and **smallest** numbers of each subarray and store the result in an array **S**.

The **special value** of the sequence **A** is the **sum of all values** stored in array **S**.

Since the answer can be very large, return it **modulo** 109+710^9 + 7109+7.

Input Format

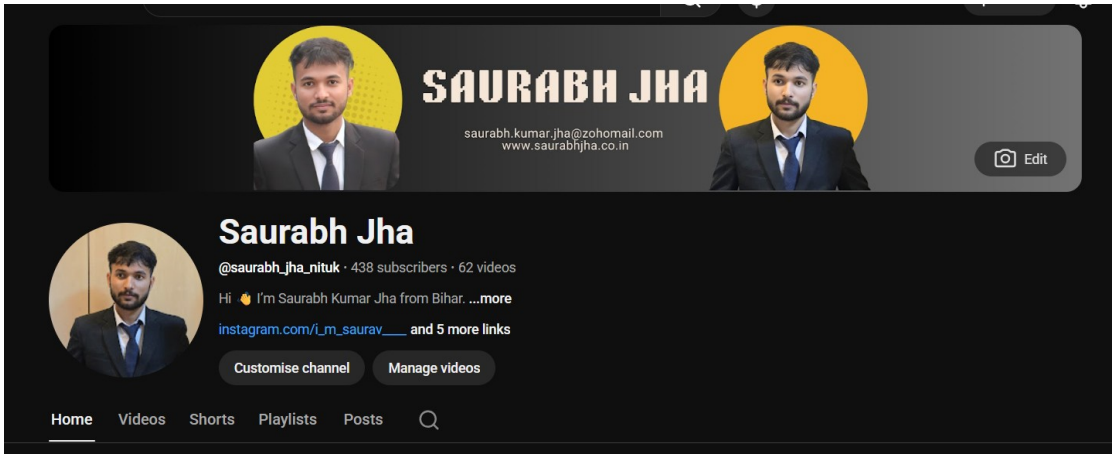
The first line contains an integer **N**, the size of the array.

The next **N lines** contain the elements of array **A**.

Constraints

1≤N≤105 1 ≤ N \le 10^5 1≤N≤105

1≤A[i]≤109 1≤ A[i] ≤ 10^9 1≤A[i]≤109



Infosys Coding Exam practice Questions

```cpp
#include <bits/stdc++.h>
using namespace std;
static const long long MOD = 1e9 + 7;
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int N;
    cin >> N;

    vector<long long> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }
    vector<long long> leftMin(N), rightMin(N);
    vector<long long> leftMax(N), rightMax(N);

    stack<int> st;
    for (int i = 0; i < N; i++) {
        while (!st.empty() && A[st.top()] > A[i]) st.pop();
        leftMin[i] = st.empty() ? (i + 1) : (i - st.top());
        st.push(i);
    }
    while (!st.empty()) st.pop();
    for (int i = N - 1; i >= 0; i--) {
        while (!st.empty() && A[st.top()] >= A[i]) st.pop();
        rightMin[i] = st.empty() ? (N - i) : (st.top() - i);
        st.push(i);
    }
    while (!st.empty()) st.pop();
    for (int i = 0; i < N; i++) {
        while (!st.empty() && A[st.top()] < A[i]) st.pop();
        leftMax[i] = st.empty() ? (i + 1) : (i - st.top());
        st.push(i);
    }
    while (!st.empty()) st.pop();
    for (int i = N - 1; i >= 0; i--) {
        while (!st.empty() && A[st.top()] <= A[i]) st.pop();
        rightMax[i] = st.empty() ? (N - i) : (st.top() - i);
        st.push(i);
    }
    long long answer = 0;

    for (int i = 0; i < N; i++) {
        long long minCount = (leftMin[i] * rightMin[i]) % MOD;
        long long maxCount = (leftMax[i] * rightMax[i]) % MOD;
        long long contribution = (A[i] * minCount) % MOD;
        contribution = (contribution * maxCount) % MOD;
        answer = (answer + contribution) % MOD;
    }

    cout << answer << "\n";
    return 0;
}
```

https://saurabhjha.live/study_material

Question 3

Magical Group

Noah and James are playing a game in which James gives Noah an integer array **A** of **N** elements.
The task of the game is to make the **largest possible magical group**.

To make the magical group, you can follow the steps below:

First, you add **any one number** to the magical group.

Then, you can add the remaining numbers from the array **A** such that they have a **common factor greater than 1** with **any of the current elements** of the magical group.

Your task is to **find and return the length of the largest possible magical group** that can be made from the given array.

Note

Single elements can also be considered as a magical group.

Input Specification

input1: An integer value **N**, representing the number of elements in the array

input2: An integer array **A**

Output Specification

Return the **length of the largest possible magical group** that can be made from the given array.

Example 1

input1: 4 input2 : {2, 3, 5, 10}

Infosys Coding Exam practice Questions

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

int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}

int magicalGroup(int N, vector<int>& A) {
    vector<bool> visited(N, false);
    int ans = 1;

    for (int i = 0; i < N; i++) {
        if (visited[i]) continue;

        queue<int> q;
        q.push(i);
        visited[i] = true;
        int count = 1;

        while (!q.empty()) {
            int u = q.front();
            q.pop();

            for (int v = 0; v < N; v++) {
                if (!visited[v] && gcd(A[u], A[v]) > 1) {
                    visited[v] = true;
                    q.push(v);
                    count++;
                }
            }
        }
        ans = max(ans, count);
    }
    return ans;
}

int main() {
    int N = 4;
    vector<int> A = {2, 3, 5, 10};
    cout << magicalGroup(N, A);
    return 0;
}
```

https://saurabhjha.live/study_material

Question 4

Fractional Knapsack

You are inside **Gringotts Gold Bank** with a knapsack that can hold **B weight** of gold coins.

In the Gringotts bank, gold coins have some unique properties:

- Each gold coin has a weight equal to some **non-negative power of 2**.
- Each gold coin can be **further divided into two gold coins of equal weight**.

There are **N gold coins** in the bank, given by an array **A**, where
A[i] is the weight of the *i-th* gold coin.

You have to **fill the knapsack completely** by making the **minimum number of divisions** of the gold coins.

Your task is to **calculate the minimum number of divisions** needed to fill the knapsack completely.

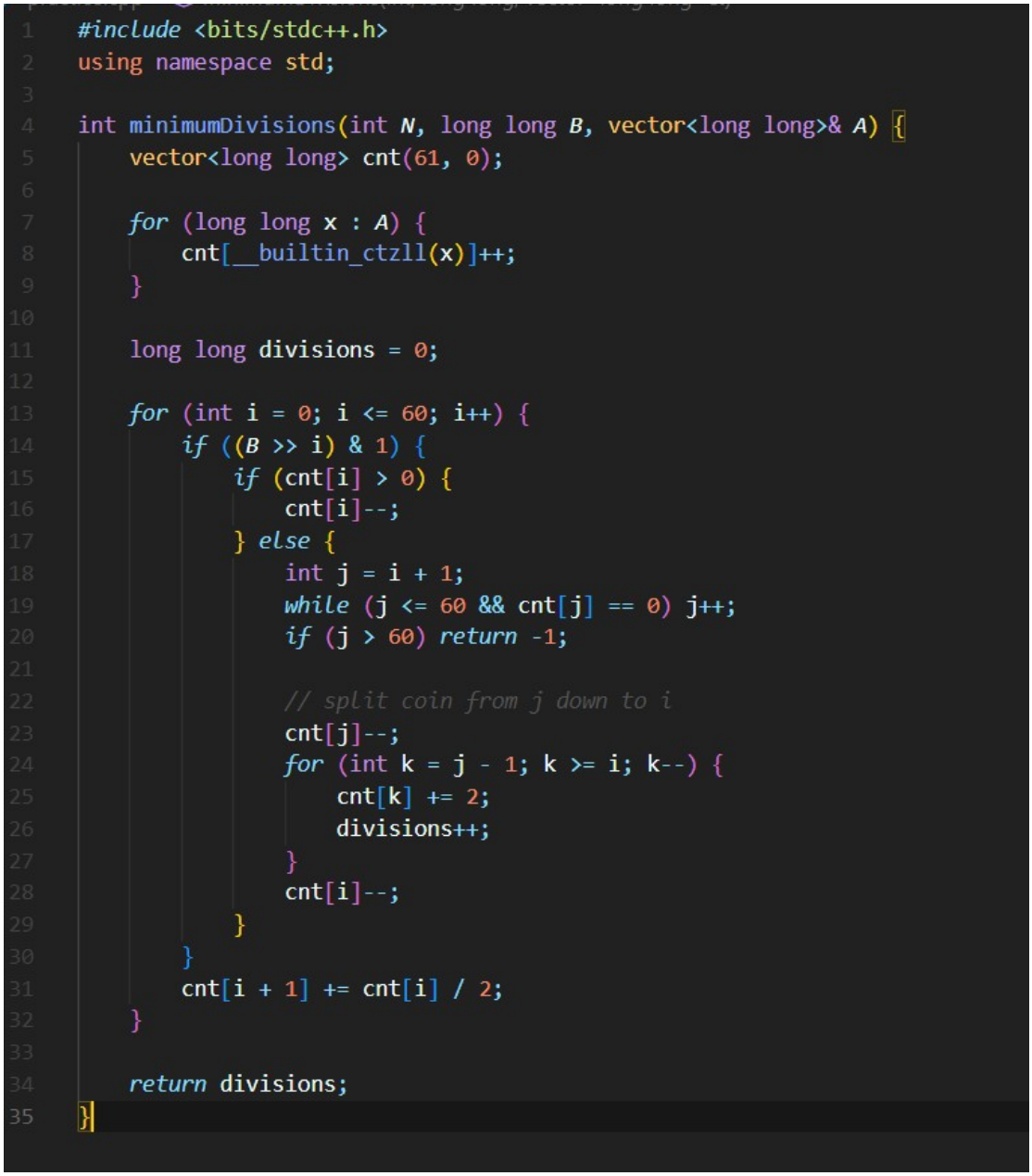
Note

If it is **impossible** to fill the knapsack completely, output **-1**.

Problem Constraints

$1 \le N \le 10^5$
$1 \le B \le 10^{18}$
$A[i] = 2^k$ (for some $k \ge 0$)

Infosys Coding Exam practice Questions



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

int minimumDivisions(int N, long long B, vector<long long>& A) {
    vector<long long> cnt(61, 0);

    for (long long x : A) {
        cnt[__builtin_ctzll(x)]++;
    }

    long long divisions = 0;

    for (int i = 0; i <= 60; i++) {
        if ((B >> i) & 1) {
            if (cnt[i] > 0) {
                cnt[i]--;
            } else {
                int j = i + 1;
                while (j <= 60 && cnt[j] == 0) j++;
                if (j > 60) return -1;

                // split coin from j down to i
                cnt[j]--;
                for (int k = j - 1; k >= i; k--) {
                    cnt[k] += 2;
                    divisions++;
                }
                cnt[i]--;
            }
        }
        cnt[i + 1] += cnt[i] / 2;
    }

    return divisions;
}
```

https://saurabhjha.live/study_material

Question 5

Array Challenge – Moving Median
Problem Description

Have the function **ArrayChallenge(arr)** read the array of numbers stored in `arr`. The **first element** in the array represents a **sliding window size N**, and the rest of the elements are the numbers.

Your program should return the **Moving Median** for each element based on the current element and its **previous N − 1 elements**, where **N** is the sliding window size.

Important Rules

For the **first few elements**, until the window size is reached, the median is calculated using the **available elements only**.

The final output should be a **string**, where the moving median corresponding to each element is **comma-separated**.

Input Format

arr = [N, a1, a2, a3, ..., ak]

N → sliding window size

Remaining values → array elements

Output Format

A **comma-separated string** of moving medians

Example 1

Input

[5, 2, 4, 6]
Output: 2,3,4

Infosys Coding Exam practice Questions

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

string ArrayChallenge(vector<int> arr) {
    int N = arr[0];
    vector<int> nums(arr.begin() + 1, arr.end());
    vector<string> result;

    for (int i = 0; i < nums.size(); i++) {
        int start = max(0, i - N + 1);
        vector<int> window(nums.begin() + start, nums.begin() + i + 1);
        sort(window.begin(), window.end());

        int len = window.size();
        int median;

        if (len % 2 == 1) {
            median = window[len / 2];
        } else {
            median = (window[len / 2 - 1] + window[len / 2]) / 2;
        }

        result.push_back(to_string(median));
    }

    // Join results with commas
    string output = result[0];
    for (int i = 1; i < result.size(); i++) {
        output += "," + result[i];
    }

    return output;
}
```

https://saurabhjha.live/study_material