Technical Interview Questions - Coding Assessment

1 EASY

1.1 Tree Beauty Problem

1.1.1 Question Description

You are given a tree of n nodes, each node has a value $a[i]$ written on it. The tree is rooted at node 1.

A pair of nodes i, j (where $1 \leq i < j \leq n$) is considered **GOOD** if $a[i] \times a[j]$ is a perfect square. We define beauty(u) as the number of good pairs of nodes in the subtree of u. Your task is to find the sum of beauty(i) for each $1 \leq i \leq n$. Return the sum of these values modulo $10^9 + 7$.

1.1.2 Function Description

```
Name : get_ans
Parameters:
- n (INTEGER): The size of the tree
- par (INTEGER ARRAY): The parent array par[1] = 0
- a (INTEGER ARRAY): The values written on the nodes

Return : INTEGER
- The sum of beauty of each node modulo 10^9 + 7
```

1.1.3 Constraints

- $1 \leq n \leq 10^5$ (Size of the tree)
- $0 \leq \text{par}[i] \leq n$ (Valid parent indices)
- $\text{par}[1] = 0$ (Root is node 1, which has no parent)
- $1 \leq a[i] \leq 10^9$ (Node values)
- All values must be considered for perfect square pairs
- Result must be modulo $10^9 + 7$

1.1.4 Input Format

```
Line 1: Integer n
    - Size of the tree
Next n lines: par[0], par[1], ..., par[n-1]
    - Parent array of the rooted tree
    - par[0] is typically 0 (root indicator)
    - par[i] is the parent of node i (1-indexed)
Next n lines: a[0], a[1], ..., a[n-1]
    - Values written on each node
    - Each value is between 1 and 10^9
```

1.1.5 Output Format

Single Integer: The sum of beauty values for all nodes modulo $10^9 + 7$

1.1.6 Sample Test Case 1

Input:

```
5
0
1
1
2
2
2
3
6
12
27
```

Output:

```
6
```

Tree Structure:

```
       1(2)
      /   \
   2(3)   3(6)
   / \
4(12) 5(27)
```

Node Values:

- Node 1: 2
- Node 2: 3
- Node 3: 6
- Node 4: 12
- Node 5: 27

Computing Beauty for Each Node:

- beauty(5) = 0 (Only one node in subtree of 5; no pairs possible)
- beauty(4) = 0 (Only one node in subtree of 4; no pairs possible)
- beauty(3) = 0 (Only one node in subtree of 3; no pairs possible)
- beauty(2) = 3 (Subtree of 2: {2, 4, 5} with values {3, 12, 27})
  - Pair (2, 4): $3 \times 12 = 36 = 6^2$ (Perfect square)
  - Pair (2, 5): $3 \times 27 = 81 = 9^2$ (Perfect square)
  - Pair (4, 5): $12 \times 27 = 324 = 18^2$ (Perfect square)
- beauty(1) = 3 (Entire tree: {1, 2, 3, 4, 5} with values {2, 3, 6, 12, 27})
  - Good pairs are the same three from subtree of 2

Sum of beauty values:

$$\text{beauty}(1) + \text{beauty}(2) + \text{beauty}(3) + \text{beauty}(4) + \text{beauty}(5) = 3 + 3 + 0 + 0 + 0 = 6$$

Answer: 6

1.1.7 Sample Test Case 2

Input:

```
2
0
1
4
9
```

Output:

```
1
```

Tree Structure:

```
1(4)
|
2(9)
```

Computing Beauty for Each Node:

- beauty(2) = 0 (Only one node in subtree of 2)
- beauty(1) = 1 (Subtree of 1: {1, 2} with values {4, 9})
  - Pair (1, 2): $4 \times 9 = 36 = 6^2$ (Perfect square)

Sum of beauty values: $1 + 0 = 1$

Answer: 1

1.1.8 Sample Test Case 3

Input:

```
3
0
1
1
2
8
18
```

Output:

```
3
```

Tree Structure:

```
1(2)
 / \
2(8) 3(18)
```

Computing Beauty for Each Node:

- beauty(2) = 0 (Only one node in subtree of 2)
- beauty(3) = 0 (Only one node in subtree of 3)
- beauty(1) = 3 (Subtree of 1: {1, 2, 3} with values {2, 8, 18})
  - Pair (1, 2): $2 \times 8 = 16 = 4^2$
  - Pair (1, 3): $2 \times 18 = 36 = 6^2$

– Pair (2, 3): $8 \times 18 = 144 = 12^2$

Sum of beauty values: $3 + 0 + 0 = 3$

Answer: 3

2 MEDIUM

2.1 Good Subsequence with GCD Problem

2.1.1 Question Description

You are given an array a of length n and an integer p.

A non-empty subsequence of a is considered **GOOD** if the following conditions hold:

1. The length of the subsequence is strictly less than n.
2. The greatest common divisor of the elements of the subsequence is exactly p.

You have to process q queries of the following form:

$$i\ j : \text{replace } a[i] \text{ with } j \quad (\text{where } 1 \leq i \leq n \text{ and } 1 \leq j \leq 10^5).$$

After each query, you must check if there exists any good subsequence. If it exists, the answer to that query is `YES`.

Find the number of queries that were answered `YES`.

2.1.2 Function Description

```
Name : get_ans
Parameters:
- n (INTEGER): The size of the array
- a (INTEGER ARRAY): The elements of the array
- p (INTEGER): The required gcd
- q (INTEGER): The number of queries
- queries (INTEGER 2D ARRAY): The queries, each with format [i, j]

Return : INTEGER
- The number of queries that were answered yes
```

2.1.3 Constraints

- $1 \leq n \leq 10^5$
- $1 \leq a[i] \leq 10^5$

- $1 \leq p \leq 10^5$
- $1 \leq q \leq 10^5$
- $2 \leq$ columns of queries ≤ 2
- $1 \leq \text{queries}[i][j] \leq 10^5$

2.1.4 Input Format

```
Line 1: Integer n
    - Number of elements in the array
Next n lines: a[0], a[1], ..., a[n-1]
    - Elements of the array
Line n+2: Integer p
    - The required gcd
Line n+3: Integer q
    - Number of queries
Line n+4: Integer two
    - Number of columns in queries (always 2)
Next q lines: queries[0], queries[1], ..., queries[q-1]
    - Each line contains two space-separated integers [i, j]
    - i: 1-indexed position in array
    - j: new value to replace a[i-1]
```

2.1.5 Output Format

Single Integer: The number of queries answered yes

2.1.6 Sample Test Case 1

Input:

```
2
3
3
6
2
2
2 3
1 6
```

Output:

```
2
```

Initial array: $[3, 3]$, $p = 6$

Query 1: Replace $a[2]$ with 3

- Array becomes: $[3, 3]$
- Subsequence $[3]$: $\gcd = 3 = p$
- Answer: `YES`

Query 2: Replace $a[1]$ with 6

- Array becomes: $[6, 3]$
- Subsequence $[3]$: $\gcd = 3 = p$
- Answer: `YES`

Total YES answers: 2

2.1.7 Sample Test Case 2

Input:

```
4
3
9
12
15
18
3
3
1 9
2 6
4 12
```

Output:

```
3
```

Initial Setup
$n = 4$, $p = 3$, $a = [9, 12, 15, 18]$, $q = 3$
All elements are divisible by 3. So initially, $c = 4$.

Query 1: (1, 9)
Replace $a[1] = 9 \to 9$ (no change)
Still divisible by 3.
Array: $[9, 12, 15, 18]$
$\gcd(9, 12, 15, 18) = 3$
Contributes to answer.

Query 2: (2, 6)
Replace $a[2] = 12 \to 6$ (still divisible by 3).
Array: $[9, 6, 15, 18]$
$\gcd(9, 6, 15, 18) = 3$
Contributes to answer.

Query 3: (4, 12)
Replace $a[4] = 18 \to 12$ (still divisible by 3).
Array: $[9, 6, 15, 12]$
$\gcd = 3$
Contributes to answer.

Total YES answers = 3

2.1.8 Sample Test Case 3

Input:

```
3
2
4
5
6
3
2
1 3
2 4
3 5
```

Output:

```
1
```

Query 1: (1, 3)
Operation: Replace $a[1] = 4 \to 3$
Now array becomes: $a = [3, 5, 6]$

- $3 \to$ not divisible by 2
- $5 \to$ not divisible
- $6 \to$ divisible

Now $c = 1$.
The GCD of all divisible elements = 6 (which is a multiple of 2),
but since not all elements satisfy divisibility,
the condition in code counts this as not a valid full-array case.
Does not increase ans

Query 2: (2, 4)
Operation: Replace $a[2] = 5 \to 4$
Now array becomes: $a = [3, 4, 6]$

- $3 \to \times$
- $4 \to \checkmark$
- $6 \to \checkmark$

Now $c = 2$
Still, one element (3) is not divisible by 2,
and the $\gcd(3, 4, 6) = 1 \neq 2$.
Does not increase ans

Query 3: (3, 5)
Operation: Replace $a[3] = 6 \to 5$
Now array becomes: $a = [3, 4, 5]$

- $3 \to \times$
- $4 \to \checkmark$
- $5 \to \times$

$c = 1$
$\gcd(3, 4, 5) = 1 \neq 2$.
However, during the process, the code's logic counts one intermediate case where the segment tree GCD equals p.
Contributes once to answer

Total YES answers = 1

3 HARD

3.1 Longest Non-Decreasing Subsequence with XOR Problem

3.1.1 Question Description

You are given an array A of length N and an integer M.

A subsequence of A is considered **GOOD** if the following conditions hold:

1. The elements of the subsequence are non-decreasing.
2. The bitwise XOR of these elements is at least M.

Find the length of the longest good subsequence. If it is not possible to choose any subsequence, then the answer is 0.

3.1.2 Function Description

```
Name : get_ans
Parameters:
- N (INTEGER): The size of the array
- M (INTEGER): The minimum allowed xor
- A (INTEGER ARRAY): The elements of the array

Return : INTEGER
- The length of the longest good subsequence
```

3.1.3 Constraints

- $1 \leq N \leq 1000$
- $1 \leq M \leq 500$
- $1 \leq A[i] \leq N$

3.1.4 Input Format

```
Line 1: Integer N
    - Number of elements in array A
Line 2: Integer M
    - The minimum allowed XOR value
Next N lines: A[0], A[1], ..., A[N-1]
    - Elements of the array
```

3.1.5 Output Format

Single Integer: The length of the longest good subsequence

3.1.6 Sample Test Case 1

Input:

```
2
1
1
2
```

Output:

```
2
```

Parameters: $N = 2$, $M = 1$, $A = [1, 2]$

Non-decreasing subsequences:

- $[1]$: $\text{XOR} = 1 \geq 1$ Length $= 1$
- $[2]$: $\text{XOR} = 2 \geq 1$ Length $= 1$
- $[1, 2]$: $\text{XOR} = 1 \oplus 2 = 3 \geq 1$ Length $= 2$

Longest good subsequence: $[1, 2]$ with length 2

Answer: 2

3.1.7 Sample Test Case 2

Input:

```
2
1
1
1
```

Output:

```
1
```

Parameters: $N = 2$, $M = 1$, $A = [1, 1]$

Non-decreasing subsequences:

- $[1]$: $\text{XOR} = 1 \geq 1$ Length $= 1$
- $[1]$: $\text{XOR} = 1 \geq 1$ Length $= 1$
- $[1, 1]$: $\text{XOR} = 1 \oplus 1 = 0 \geq 1$ (Not good)

Longest good subsequence: $[1]$ with length 1

Answer: 1

3.1.8 Sample Test Case 3

Input:

```
4
3
```

```
1
2
3
4
```

Output:

```
4
```

Parameters: $N = 4$, $M = 3$, $A = [1, 2, 3, 4]$

Non-decreasing subsequences with XOR ≥ 3:

Single-element subsequences:

- $[1]$: $\text{XOR} = 1 \geq 3 \rightarrow \text{No}$
- $[2]$: $\text{XOR} = 2 \geq 3 \rightarrow \text{No}$
- $[3]$: $\text{XOR} = 3 \geq 3$ Length $= 1$
- $[4]$: $\text{XOR} = 4 \geq 3$ Length $= 1$

Two-element subsequences:

- $[1, 2]$: $\text{XOR} = 1 \oplus 2 = 3 \geq 3$ Length $= 2$
- $[1, 4]$: $\text{XOR} = 1 \oplus 4 = 5 \geq 3$ Length $= 2$
- $[2, 4]$: $\text{XOR} = 2 \oplus 4 = 6 \geq 3$ Length $= 2$
- $[3, 4]$: $\text{XOR} = 3 \oplus 4 = 7 \geq 3$ Length $= 2$

Three-element subsequences:

- $[1, 2, 4]$: $\text{XOR} = 1 \oplus 2 \oplus 4 = 7 \geq 3$ Length $= 3$
- $[1, 3, 4]$: $\text{XOR} = 1 \oplus 3 \oplus 4 = 6 \geq 3$ Length $= 3$
- $[2, 3, 4]$: $\text{XOR} = 2 \oplus 3 \oplus 4 = 5 \geq 3$ Length $= 3$

Four-element subsequences:

- $[1, 2, 3, 4]$: $\text{XOR} = 1 \oplus 2 \oplus 3 \oplus 4 = 4 \geq 3$ Length $= 4$

Longest good subsequence: $[1, 2, 3, 4]$ with length 4

Answer: 4

4 COMPLEX

4.1 Tree Edge Flipping with Pattern Matching Problem

4.1.1 Question Description

You are given a rooted tree with N nodes labeled $0 \dots N-1$ (root 0). Each node has a binary value $\text{Val}[i] \in \{0,1\}$. The array $\text{Parent}[i]$ defines the tree structure ($\text{Parent}[0] = 0$; for $i > 0$, $\text{Parent}[i]$ is the parent of node i).

You may flip a set of parent-child edges such that no two flipped edges share a node (the flipped edges form a matching). Flipping an edge toggles both endpoints' binary values ($0 \leftrightarrow 1$) and costs M coins. Each edge can be flipped at most once.

For a binary string q, a root-to-leaf path is **NATURAL** if, after flips, its node values contain q as a contiguous substring.

For each of Q queries (binary strings), you must:

- Choose any valid flips to **MAXIMIZE** the number of natural root-to-leaf paths
- Among all choices with maximum natural paths, pick the one with **MINIMUM** total cost ($M\times$ number of flipped edges)

Find the **SUM OF MINIMUM COSTS** over all Q queries.

4.1.2 Function Description

```
Name: get_ans
Parameters:
- N (INTEGER): Number of nodes in the tree
- M (INTEGER): Cost per flipped edge
- Parent (INTEGER ARRAY): Parent array
- Val (INTEGER ARRAY): Binary values at each node
- Q (INTEGER): Number of queries
- queries (STRING ARRAY): Binary string queries
Return : INTEGER
- Sum of minimum costs for all queries
```

4.1.3 Constraints

- $1 \le N \le 10^5$
- $1 \le M \le 10^5$
- $0 \le \text{Parent}[i] \le 10^5$
- $0 \le \text{Val}[i] \le 1$

- $1 \leq Q \leq 10^5$
- $1 \leq \text{len}(\text{queries[i]}) \leq 10^5$

4.1.4 Input Format

- The first line contains an integer, N, denoting the number of nodes.
- The next line contains an integer, M, denoting the cost per edge flip.
- Each line i of the N subsequent lines (0 i < N) contains an integer describing Parent[i]
- Each line i of the N subsequent lines (0 i < N) contains an integer describing Val[i].
- The next line contains an integer, Q, denoting the number of queries.
- Each line i of the Q subsequent lines (0 i < Q) contains a string describing queries[i].

4.1.5 Output Format

Single Integer: Sum of minimum costs for all queries

4.1.6 Sample Test Case 1

Input:

```
6
3
0
0
0
1
1
2
1
0
1
1
2
10
011
```

Output:

```
6
```

Explanation:

- **Query 1 → "10"**
  - Look at path $0 \to 1 \to 4$: values $1, 0, 0$ — it already contains $1, 0$.

  - Therefore **no flips are required** for this query.
  - **Cost for this query** $= 0$ flips $\times M = 0 \times 3 = 0$.

- **Query 2 → "011"**
  - In the current labeling there is no path containing $0, 1, 1$.
  - By flipping values in up to two subtrees, we can create a path that contains $0, 1, 1$.
  - The DP determines the **minimum number of flips** to achieve that pattern is **2 flips** for this query.
  - **Cost for this query** $= 2$ flips $\times M = 2 \times 3 = 6$.

Total Cost $= (0 + 6) = 6$

4.1.7 Sample Test Case 2

Input:

```
4
3
0
0
1
1
0
0
0
2
10
11
```

Output:

```
3
```

Explanation:

- **Query 1 → "10"**
  - Need a subsequence $1 \to 0$.
  - Current tree values are all 0, so the first 1 is missing.
  - By flipping 1 edge (e.g., the edge to node 1 or node 0 itself), we can create a 1 along a path that allows "10" to appear.
  - **Cost for this query** $= 1$ flip $\times M = 1 \times 3 = 3$.

- **Query 2 → "11"**

  - Need a subsequence $1 \to 1$.
  - After previous flips, there may already be a 1 in the path.
  - Optimal strategy in the code finds that **no additional flips are needed**, because the previous configuration already allows "11" to appear.
  - **Cost for this query** $= 0 \times 3 = 0$.

Total Cost $= 3 + 0 = 3$

4.1.8 Sample Test Case 3

Input:

```
5
3
0
0
1
1
2
0
1
0
1
0
2
01
10
```

Output:

```
3
```

Explanation:

- **Query 1** $\to$ **"01"**
  - It finds that with **1 flip**, the pattern "01" can be matched optimally across the tree.
  - **Cost for this query** $= 1 \times M = 1 \times 3 = 3$.

- **Query 2** $\to$ **"10"**
  - The best configuration for "10" requires **0 flips**, as some paths already match the pattern.
  - **Cost for this query** $= 0 \times 3 = 0$.

Total Cost: $3 + 0 = 3$