Module 34 lecture 1 leetcode 1944

Number of Visible People in a Queue - Detailed Explanation & Solution

Problem Statement:

Ek array heights diya gaya hai jisme har person ki height di gayi hai. Humein ye determine karna hai ki har person apne right side kitne logon ko dekh sakta hai. Koi person i, kisi person j ko dekh sakta hai agar:

- min(heights[i], heights[j]) > max(heights[i+1], ..., heights[j-1])

Approach:

Is problem ko efficiently solve karne ke liye hum **monotonic decreasing stack** ka use karenge. Ye stack humein efficiently ye determine karne me madad karega ki kaunse log visible honge jab hum array ko reverse order me traverse karenge.

Steps to Solve:

1. **Data Structures Initialize Karein**:
   - Ek vector ans jo final answer store karega.
   - Ek stack st jo decreasing order me heights ke indices track karega.
2. **Right to Left Traverse Karein**:
   - n-2 se lekar 0 tak iterate karein kyunki last person kisi ko nahi dekh sakta.
3. **Stack ka Use Karke Visibility Check Karein**:
   - Ek count variable rakhein jo track karega ki kitne log visible hain.
   - Agar current person stack ke top wale person se bada hai, to chhoti height wale log pop hote rahenge aur count badhta rahega.
   - Agar stack me ab bhi koi element hai to wo ek aur visible hoga (count++).
4. **Result Store Karein aur Stack Update Karein**:
   - count ko ans[i] me store karein.
   - Current person ka index stack me push karein taaki aage wale log ise check kar sakein.

Code Implementation:

```cpp
class Solution {
public:
    vector<int> canSeePersonsCount(vector<int>& arr) {
        int n = arr.size();
        vector<int> ans(n);
        stack<int> st;
        st.push(n-1);
        ans[n-1] = 0;
        for(int i = n-2; i >= 0; i--) {
            int count = 0;
            while (st.size() > 0 && arr[i] > arr[st.top()]) {
                count++;
                st.pop();
            }
            if (st.size() > 0) {
                count++;
                ans[i] = count;
            }
            else if (st.size() == 0) ans[i] = count;
            st.push(i);
        }
        return ans;
    }
};
```

Code Execution ka Explanation:

1. **Variables Initialize Karein**:
   - ans vector jisme answer store hoga.
   - stack jo decreasing order me heights track karega.
2. **Rightmost Person 0 Log Dekh Sakta Hai (Base Case)**:
   - ans[n-1] = 0
   - n-1 wale index ko stack me push karein.
3. **Loop Chalayen n-2 se 0 tak**:
   - **While loop** chhoti height wale log pop karega aur count badhayega.
   - Agar stack me koi element bacha hai, to ek aur banda visible hoga.
   - count ko store karein aur current index ko stack me push karein.

Dry Run Example:

Input: heights = [10, 6, 8, 5, 11, 9]

Execution Steps:

Index	Height	Stack Before	Visible Count	Stack After
5	9	[]	0	[5]
4	11	[5]	1	[4]
3	5	[4]	1	[4,3]
2	8	[4,3]	2	[4,2]
1	6	[4,2]	1	[4,2,1]
0	10	[4,2,1]	4	[4,0]

Output: [4,1,2,1,1,0]

Complexity Analysis:

- **Time Complexity:** O(N)
  - Har element ek baar push aur ek baar pop hota hai → **Total O(N)**
- **Space Complexity:** O(N) (Stack aur answer vector ke liye)

Key Takeaways:

- **Monotonic decreasing stack** efficiently visibility track karta hai.
- **Reverse traversal** ensures ki rightmost persons pehle process ho jayein.
- **Optimized approach** jo har element ko sirf ek baar process karti hai.

Is approach se hum efficiently problem solve kar sakte hain bina kisi extra traversal ke! 🚀