Sliding Window Maximum (LeetCode 239) - Optimized Approach

Problem Statement

Ek array `nums` diya hai aur ek integer `k` diya hai. Ek sliding window hai jo left se right move karega, aur har position par hum sirf `k` elements dekh sakte hain. Har window ka maximum find karna hai.

Approach

Ye ek **hard** problem hai, isliye brute-force approach time complexity ke wajah se feasible nahi hai. Isko optimize karne ke liye **Next Greater Index (NGI) aur stack** ka use kiya hai.

Steps:

1. **Next Greater Index (NGI) find karna**:
- Right se left traverse karenge aur ek **monotonic decreasing stack** ka use karenge taaki **next greater element ka index** mile.
- Agar kisi number ka right me koi bada element nahi mila, toh `n` store karenge.
2. **NGI ka use karke sliding window ka maximum find karna**:
- Ek variable `j = 0` lenge jo window ka max element track karega.
- Loop chalega `i = 0` se `i <= n - k` tak taaki sabhi windows cover ho sakein.
- Har window ke liye:
- Agar `j < i` hai, toh `j = i` set karenge taaki window ke andar aaye.
- `j` ko `ngi[j]` ke through move karenge jab tak window ke bahar na chala jaye.
- `nums[j]` hi window ka maximum hoga.

Code Implementation

```cpp
class Solution {
public:
vector<int> maxSlidingWindow(vector<int>& v, int k) {
int n = v.size();
stack<int> st;
vector<int> ngi(n);
st.push(n-1);
ngi[n-1] = n;
for(int i = n-2; i >= 0; i--) {
while(st.size() > 0 && v[i] > v[st.top()]) {
st.pop();
}
if(st.size() > 0) {
ngi[i] = st.top();
} else if(st.size() == 0) {
ngi[i] = n;
}
st.push(i);
}

vector<int> ans;
int j = 0;
for(int i = 0; i <= n-k; i++) {
if(j < i) j = i;
int Max = v[i];
while(j < i + k) {
Max = v[j];
if(ngi[j] > i + k) break;
j = ngi[j];
}
ans.push_back(Max);
}
return ans;
}
};
```

Complexity Analysis

- **NGI find karna**: `O(n)` (stack use karke optimize kiya hai).
- **Har window ka max find karna**: `O(n)`, kyunki `j` sirf aage badhta hai.
- **Total Complexity**: `O(n)`, jo ki bahut efficient hai.

Summary

- **Next Greater Index (NGI) aur stack** ka use karke precompute kiya ki har element ka agla bada element kaunsa hai.
- **Sliding window traversal optimize kiya** taaki har bar `k` elements check na karne pade.
- **O(n) time complexity achieve ki**, jo bade inputs ke liye best hai.

Notes

- **Key Idea**: Har `k` elements ko check karne ke bajay **NGI ka use karke** direct max

element par jump karna.
- **Common Mistake**: `j` ko hamesha bounds ke andar rakhna taaki `nums[j]` ka access out-of-bound na ho.
- **Alternative Approach**: **Deque** ka use karke bhi ye problem solve ho sakti hai, jo ek aur efficient tarika hai.

Is approach se har sliding window ka maximum efficiently find ho sakta hai!