|  |  |
| --- | --- |
| **Sliding Window max in C++** | |
| #include <iostream>  #include <vector>  #include <stack>  using namespace std;  vector<int> slidingWindowMaximum(vector<int>& arr, int k) {  int n = arr.size();  vector<int> result;  stack<int> st;  vector<int> nge(n);  st.push(n-1);  nge[n-1] = n;  for (int i = n-2; i >= 0; i--) {  while (!st.empty() && arr[i] >= arr[st.top()]) {  st.pop();  }  if (st.empty()) {  nge[i] = n;  } else {  nge[i] = st.top();  }  st.push(i);  }  for (int i = 0; i <= n-k; i++) {  int j = i;  while (nge[j] < i+k) {  j = nge[j];  }  result.push\_back(arr[j]);  }  return result;  }  int main() {  // Hardcoded input  vector<int> arr = {1, 3, -1, -3, 5, 3, 6, 7};  int k = 3;  vector<int> result = slidingWindowMaximum(arr, k);  // Output the result  for (int num : result) {  cout << num << " ";  }  cout << endl;  return 0;  } | Input: arr = {1, 3, -1, -3, 5, 3, 6, 7}  k = 3  n = 8 Step 1: Compute **Next Greater Element Index Array (nge[])** We initialize an array nge[n], where:   * nge[i] = index of the next greater element to the right of arr[i] * If no such index, set nge[i] = n  📊 NGE Construction Table We build from **right to left** using a stack:   | **i** | **arr[i]** | **Stack (Top to Bottom)** | **nge[i]** | | --- | --- | --- | --- | | 7 | 7 | [7] | 8 | | 6 | 6 | [7, 6] | 7 | | 5 | 3 | [7, 6, 5] | 6 | | 4 | 5 | [7, 6, 4] | 6 | | 3 | -3 | [7, 6, 4, 3] | 4 | | 2 | -1 | [7, 6, 4, 2] | 4 | | 1 | 3 | [7, 6, 4, 1] | 4 | | 0 | 1 | [7, 6, 4, 1, 0] | 1 |   ➡ Final nge[] = {1, 4, 4, 4, 6, 6, 7, 8} Step 2: Compute **Max in Each Sliding Window** For each window starting at i, you walk forward using nge[] until nge[j] >= i + k. 🧮 Sliding Window Loop (i = 0 to n - k)  | **i** | **Window** | **j Traversal (via NGE)** | **Max Value** | | --- | --- | --- | --- | | 0 | [1 3 -1] | 0 → 1 | 3 | | 1 | [3 -1 -3] | 1 → 4 (exits, 4 ≥ 4) | 3 | | 2 | [-1 -3 5] | 2 → 4 | 5 | | 3 | [-3 5 3] | 3 → 4 | 5 | | 4 | [5 3 6] | 4 → 6 | 6 | | 5 | [3 6 7] | 5 → 6 → 7 | 7 |  ✅ Output: 3 3 5 5 6 7 |
| 3 3 5 5 6 7 | |