|  |  |
| --- | --- |
| **Reverse bits in C++** | |
| #include <iostream>  #include <vector>  #include <deque>  using namespace std;  class SlidingWindowMinimum {  public:  vector<int> getMinimums(vector<int>& nums, int k) {  int n = nums.size();  vector<int> ans;  if (k > n) return ans;  deque<int> deque;    // Process the first window of size k  for (int i = 0; i < k; i++) {  while (!deque.empty() && deque.back() > nums[i]) {  deque.pop\_back();  }  deque.push\_back(nums[i]);  }  ans.push\_back(deque.front()); // Store the minimum for the first window    // Process the rest of the elements  for (int i = k; i < n; i++) {  if (deque.front() == nums[i - k]) {  deque.pop\_front(); // Remove the element that is no longer in the window  }  while (!deque.empty() && deque.back() > nums[i]) {  deque.pop\_back(); // Maintain the deque in descending order  }  deque.push\_back(nums[i]);  ans.push\_back(deque.front()); // Store the minimum for the current window  }  return ans;  }  };  int main() {  SlidingWindowMinimum swm;  // Test case 1  vector<int> nums1 = {1, 3, -1, -3, 5, 3, 6, 7};  int k1 = 3;  vector<int> result1 = swm.getMinimums(nums1, k1);  cout << "Minimums for nums1 and k=" << k1 << ": ";  for (int num : result1) {  cout << num << " ";  }  cout << endl;  return 0;  } | Step-by-Step Dry Run (Tracking All Key Values):  | **i** | **nums[i]** | **Deque (indices)** | **Deque (values)** | **Action** | **Window** | **Min** | | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | [0] | [1] | Initial push | - | - | | 1 | 3 | [0, 1] | [1, 3] | 3 >= 1, keep 0, push 1 | - | - | | 2 | -1 | [2] | [-1] | Pop 1 and 0 (both > -1), push 2 | [1, 3, -1] | -1 | | 3 | -3 | [3] | [-3] | Pop 2 (nums[2]=-1 > -3), push 3 | [3, -1, -3] | -3 | | 4 | 5 | [3, 4] | [-3, 5] | 5 > -3, keep 3, push 4 | [-1, -3, 5] | -3 | | 5 | 3 | [3, 5] | [-3, 3] | Pop 4 (5 > 3), keep 3, push 5 | [-3, 5, 3] | -3 | | 6 | 6 | [5, 6] | [3, 6] | Pop 3 (index out of range), pop 3 (nums[3]=-3 is out), push 6 | [5, 3, 6] | 3 | | 7 | 7 | [5, 6, 7] | [3, 6, 7] | 7 > 6, keep 6, push 7 | [3, 6, 7] | 3 |  ✅ Final Output: Minimums for nums1 and k=3: -1 -3 -3 -3 3 3 |
| Minimums for nums1 and k=3: -1 -3 -3 -3 3 3 | |