|  |  |
| --- | --- |
| **Equivalent Subarrays in C++** | |
| #include <iostream>  #include <unordered\_map>  #include <unordered\_set>  #include <vector>  using namespace std;  int main() {  int ans = 0;  vector<int> arr = {2, 1, 3, 2, 3};  unordered\_set<int> set;  // Insert unique elements into the set  for (int i = 0; i < arr.size(); i++) {  set.insert(arr[i]);  }  int k = set.size();  int i = -1;  int j = -1;  unordered\_map<int, int> map;  while (true) {  bool f1 = false;  bool f2 = false;  // Expand the window until all unique elements are covered  while (i < arr.size() - 1) {  f1 = true;  i++;  map[arr[i]] = map[arr[i]] + 1; // Add current element to the map  if (map.size() == k) { // If all unique elements are covered  ans += arr.size() - i; // Add the number of valid subarrays ending at index i  break;  }  }  // Slide the window to the right until the uniqueness condition is violated  while (j < i) {  f2 = true;  j++;  if (map[arr[j]] == 1) {  map.erase(arr[j]); // Remove element from map if its count is reduced to 0  } else {  map[arr[j]] = map[arr[j]] - 1; // Decrease the count of the element  }  // If the map size matches k, add the number of valid subarrays again  if (map.size() == k) {  ans += arr.size() - i;  } else {  break;  }  }  // If both windows cannot be expanded or contracted further, break the loop  if (!f1 && !f2) {  break;  }  }  // Print the total number of equivalent subarrays  cout << ans << endl;  return 0;  } | **Step 1: Initializing Variables**   * **Input Array**: {2, 1, 3, 2, 3} * **Unique Elements (set)**:   {2, 1, 3} → k = 3 (total unique elements)   * **Pointers**:   i = -1, j = -1  ans = 0  map = {} (empty frequency map)  **Step 2: Expanding the Window (Outer while Loop)**  **Expanding i Until map.size() == k**   | **i** | **arr[i]** | **map (after update)** | **map.size()** | **Condition map.size() == k?** | | --- | --- | --- | --- | --- | | 0 | 2 | {2: 1} | 1 | ❌ | | 1 | 1 | {2: 1, 1: 1} | 2 | ❌ | | 2 | 3 | {2: 1, 1: 1, 3: 1} | 3 | ✅ → Add arr.size() - i = 5 - 2 = 3 to ans |  * **ans = 3**   **Step 3: Contracting j Until map.size() < k**   | **j** | **arr[j]** | **map (after update)** | **map.size()** | **Condition map.size() == k?** | **ans Update** | | --- | --- | --- | --- | --- | --- | | 0 | 2 | {2: 0, 1: 1, 3: 1} → removed 2 | 2 | ❌ | **Break** |   **Step 4: Continue Expanding i**   | **i** | **arr[i]** | **map (after update)** | **map.size()** | **Condition map.size() == k?** | **ans Update** | | --- | --- | --- | --- | --- | --- | | 3 | 2 | {1: 1, 3: 1, 2: 1} | 3 | ✅ | Add arr.size() - i = 5 - 3 = 2 | | **New ans** | **3 + 2 = 5** |  |  |  |  |   **Step 5: Contracting j Again**   | **j** | **arr[j]** | **map (after update)** | **map.size()** | **Condition map.size() == k?** | **ans Update** | | --- | --- | --- | --- | --- | --- | | 1 | 1 | {1: 0, 3: 1, 2: 1} → removed 1 | 2 | ❌ | **Break** |   **Step 6: Continue Expanding i**   | **i** | **arr[i]** | **map (after update)** | **map.size()** | **Condition map.size() == k?** | **ans Update** | | --- | --- | --- | --- | --- | --- | | 4 | 3 | {3: 2, 2: 1} | 2 | ❌ | No update |   **Final Output**  5  **Summary of Valid Subarrays**   * The total number of subarrays containing all **3 distinct elements {1, 2, 3}** is **5**. |
| Output:- 0 | |