Question: _

Input: Your Solution Our Solution Run C

```
// Copyright © 2019 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <unordered_map>
using namespace std;

// O(n) time | O(n) space
vector<int> largestRange(vector<int> array) {
  vector<int> bestRange = {};
  int longestLength = 0;
  unordered_map<int, bool> nums = {};
  for (int num : array) {
    nums[num] = true;
  }
  for (int num : array) {
    if (!nums[num]) {
      continue;
    }
    nums[num] = false;
    int currentLength = 1;
    int left = num - 1;
    int right = num + 1;
    while (nums.find(left) != nums.end()) {
      nums[left] = false;
      currentLength++;
      left--;
    }
    while (nums.find(right) != nums.end()) {
      nums[right] = false;
      currentLength++;
      right++;
    }
    if (currentLength > longestLength) {
      longestLength = currentLength;
      bestRange = {left + 1, right - 1};
    }
  }
  return bestRange;
}
```

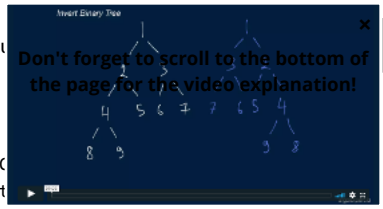
Output: Custom Output Raw Output

Help: Hide Show

Hint #1 Hint #2 Hint #3 Optimal Space & Time Complexity

How can you use a hash table to solve this problem with an algorithm that runs in linear time?

Tests: Ou

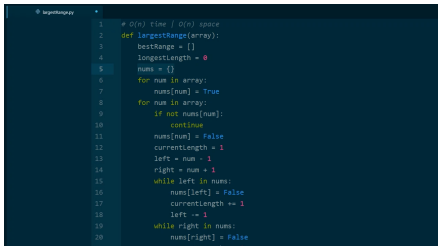



```
                                                    1,  6,  13, 14, -2}) ;
}

TEST_C
  vect
  REQUIRE(largestRange({-7, -7, -7, -7, 8,  -8, 0,  9, 19, -1, -3,
                        17, 2,  10, 3,  12, 5,  16, 4, 11, -6, 8,
                        6,  15, 12, 12, -5, 2,  1,  6, 13, 14, -4,
          expected);
}
TEST_END;
```

Video Explanation

Go to Conceptual Overview



Go to Code Walkthrough

Questions List (/questions)