Input: An array of integers
array = [1, 11, 3, 0, 15, 5, 2, 4, 10, 7, 12, 6]

Output: An array of length 2 representing the largest range of integers contained in that array
[0, 7] since we have: 0, 1, 2, 3, 4, 5, 6, 7

```
// O(n) time | O(n) space
function largestRange(array) {
  let bestRange = [];
  let longestLength = 0;
  const nums = {};

  for (const num of array) {
    nums[num] = true;
  }

  for (const num of array) {
    if (!nums[num]) continue;
    nums[num] = false;
    let currentLength = 1;
    let left = num - 1;
    let right = num + 1;
    while (left in nums) {
      nums[left] = false;
      currentLength++;
      left--;
    }
    while (right in nums) {
      nums[right] = false;
      currentLength++;
      right++;
    }

    if (currentLength > longestLength) {
      longestLength = currentLength;
      bestRange = [left + 1, right - 1];
    }
  }
  return bestRange;
}
```

Ideas:

- Iterate through the array and add each nums to the hash table with the value true
- Iterate through the array and at each iteration:
  - Check if that num value is false (meaning we already "seen" it) if it is, continue to the next num
  - Set the current length to 0
  - Set the left number to num - 1
  - Set the right number to num + 2
  - Check if left is in the hash table:
    - if it is, then increment current length, decrement left and set the left num to false
  - Check if right is in the hash table:
    - If it is, set right num to false in hash table, increment right and increment current length
  - If the curr length > max length, update the max length and the final array to [left + 1, right - 1]
- return the final array which has the best range

Time: O(n) (where n is the # of elements in the input array)
since we iterate thru once (n) then once again (n + n)
but at each one we only iterate if it has not been visited before (n + n + n) or O(n)

Space: O(n) bc of our hash table. Stores all elements to be used for constant time look up.