Code Challenge #2 Validate Subsequence (Easy)

Sample Input

```
array = [5, 1, 22, 25, 6, -1, 8, 10]

sequence = [1, 6, -1, 10]
```

Sample Output

```
true
```

Solution #1

```
function isValidSubsequence(array, sequence) {
   let arrayIndex = 0;
  let sequenceIndex = 0;
  while (arrayIndex < array.length && sequenceIndex < sequence.length) {
        if (array[arrayIndex] === sequence[sequenceIndex]) {sequenceIndex++;}
        arrayIndex++;
  }
  return sequenceIndex === sequence.length;
}

```

Explanation

The first solution uses two tracking variables to iterate through the items in the two arrays given as arguments to the function. We start both variables at zero and iterate through both the arrays using a while loop. The while loop runs as long as both tracking variable are lower than the length of the arrays. If one or both tracking variable values are no longer less than length of the arrays the loop will break. The tracker variables check each item in their respective arrays to see if they match. If they do match we increment the value of the second tracker variable by one. Regardless if they match or not the first tracker variable is incremented by one. This will eventually lead to the while loop breaking. The final value for the second variable should match the length of the second array if the second array in the argument is a subsequence of the first array. The second variable value wouldn’t be the same number as the length of the second array if it didn’t find the **same exact number** of matches. This solution is O(n) time complexity.

Code Pattern: Two Tracker variables and while loop

```
 let arrayIndex = 0;
  let sequenceIndex = 0;
  while () {
         if (   ) {sequenceIndex++;}
         arrayIndex++;
  }
  return
}
```

Solution #2

```
function isValidSubsequence(array, sequence) {
let seqIdx = 0;
  for (const value of array) {
         if (seqIdx === sequence.length) break;
         if (sequence[seqIdx] === value) seqIdx++;
  }
  return seqIdx === sequence.length;
}

```

Explanation

The second solution has only one tracker variable for the second array in the argument. It uses a syntactic sugar loop to loop through the items in the array provided by the first argument. It first checks for an edge case where if the value of the tracker variable matches the length of the second array. If it does it breaks the loop and finishes the last part of the function which is the return part. If the edge case does not occur, it checks to see if the value in the second array is equal to the value of the first array. If it matches, then it increments the tracker variable by one. It finally returns a true or false value if tracker variable value is equal to the length of the second array.

Code Pattern: One tracker variable with syntactic sugar for loop.

```
let seqIdx = 0;
  for (const value of array) {

  }
  return;
}

```