Technique: Sliding Window using Two Pointers

Level: Medium

Given a String, find the longest substring with unique characters.

For example: "whatwhywhere" --> "atwhy"

Questions to Clarify:
Q. How to return the result?
A. Return the start and end indexes of the substring.

Q. What to return if the array is empty or null?
A. Return null.

Q. So the shortest possible substring is just one character?
A. Yes.

Q. What to do if there are multiple longest substrings?
A. Return any one of them.

Q. Do we have only alphabets?
A. No, there can be any character.

Note: If the interviewer restricts the input to alphabets, you can use an array of size 52 (26 x 2) instead of using a Hashmap.

Solution:
We use the sliding window technique. At any point, our window will have unique characters, i.e, no duplicates. Let's say we are in the middle of processing.

Our window is as follows:

w h **a t w h y** w h e r e
| |

We try to expand our window by increasing the *end* pointer by 1:

w h **a t w h y w** h e r e
| |

We look at the new character, w. We see that w is already in our window. We have two w's now. Remember, we only care about windows with unique characters. To make our window unique, we advance the start pointer till after the previous w (to h):

w h a t w **h y w** h e r e

```
       |  |
```

Now, our window has unique characters again. We keep doing this - moving the window forward - until we reach the end of the array.

The largest window we encounter is the result.

To quickly look up if a character is in our window, we use a Hash map.

Pseudocode:

(Note: Never write pseudocode in an actual interview. Unless you're writing a few lines quickly to plan out your solution. Your actual solution should be in a real language and use good syntax.)

```
// init variables with first character as result.
start = 0, end = 0, longest = 1
result = (0,0)
initialize hash map, put first character in it.

while end is less than a.length - 1:
     end++
     if the new character (ch) is already in hash map:
         update start - put it ahead of ch's previous appearance

     add ch to map at index end

     // our new window is (start, end)
     check if new window is larger and update result

return result
```

Test Cases:
Edge Cases: empty array, null array
Base Cases: single char, 2 chars (same, different)
Regular Cases: multiple chars (with/without duplicates)

Time Complexity: O(n)

Space Complexity: O(size of character set), which is typically a fixed number, so O(1)

```java
public static Pair<Integer> allUnique(String input) {
    if (input == null || input.isEmpty()) {
        return null;
    }

    HashMap<Character, Integer> map = new HashMap<>();
    Pair<Integer> result = new Pair<Integer>(0, 0);
    int start = 0, end = 0, longest = 1;
    map.put(input.charAt(0), 0);


    while (end < input.length() - 1) {
        // expand end pointer
        end++;
        char toAdd = input.charAt(end);
        if (map.containsKey(toAdd) && map.get(toAdd) >= start) {
            start = map.get(toAdd) + 1;
        }
        map.put(toAdd,  end);

        // update result
        if (end - start + 1 > longest) {
            longest = end - start + 1;
            result.setFirst(start);
            result.setSecond(end);
        }
    }

    return result;
}
```