1624. Largest Substring Between Two Equal Characters

Given a string return length of longest substring between two equal characters. If none, return -1

Examples:

Input : "$\overset{\downarrow}{a}$bcde$\overset{\downarrow}{a}$ei"

Output: 4

Idea was a "sliding window" approach

"$\overset{\downarrow}{a}$bcdeae$\overset{\downarrow}{i}$" result = -1

"$\overset{\downarrow}{a}$bcdea$\overset{\downarrow}{e}$i" result = -1

"$\overset{\downarrow}{a}$bcde$\overset{\downarrow}{a}$ei" result = 4

Question: how do we determine which side to increment/decrement?

I don't think it is possible. We do not know where the matching characters are and it is either valid or invalid.

New approach:

1.) Iterate over entire array and set the first index of new/unique characters.

2.) Iterate over entire array and if index for character do not match, update.

3.) Return result

```
int maxLen( const std::string& s ) {

    int result { -1 };
    std::unordered_map<char, int> m;

    for (int i=0; i< s.size(); ++i) {
        if ( ! m.contains(s[i])) {
            m[s[i]] = i;
        }
    }

    for (int i=0; i< s.size() ; ++i) {
        if ( m.contains(s[i])) {
            const auto distance {  i - m[s[i]] - 1 };
            result =  std::max( result, distance );
        }
    }

    return result;
}
```