525. Contiguous Array

Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0's and 1's.

Ex.

Input: 0 0 1 0 1 1 0 0 0

Output: 6

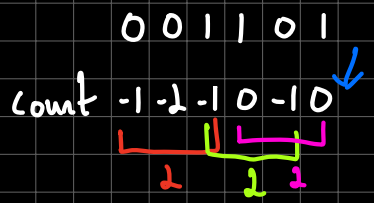
Constraints:

- $1 <= nums.length <= 10^5$
- nums[i] = 1 or 0

Solution #1: Brute Force

```
for (auto i{ arr.cbegin() }; i< arr.cend(); ++i){        O(n)
    for (auto j{ i+1 }; j < = arr.cend(); ++j){          O(n)
        const auto zero = std::count(i, j, 0);           O(n) => O(1)
        const auto one = std::count(i, j, 1);            by having variables
        if (one == zero) {                                  w/ count.
            answer = std::max( one+zero, answer);
        }
    }
}
```

Improvements:



1.) If count == 0 => ans = current iterator

2.) If m.contains(count) => ans = std::max(ans, i - m.at(count))

m will contain the index of the first count.

```
auto findMaxLength(const std::vector<int>& nums) -> int {
    if (nums.size() == 1) {
        return 1;
    }
    int count{};
    int ans{1};
    std::unordered_map<int, int> m;

    for (int i{}; i < nums.size(); ++i) {
        count += nums.at(i) ? 1 : -1;
        if (0 == count) {
            ans = i + 1;
            continue;
        }
        if (m.contains(count)){
            ans = std::max(ans, i- m.at(count));
            continue;
        }
        m.insert({count, i});
    }
    return ans;
}
```