Largest subarray with equal number of 0s and 1s

Given an array containing only 0s and 1s, find the largest subarray which contains equal no of 0s and 1s. The expected time complexity is O(n).

Examples:

```
Input: arr[] = {1, 0, 1, 1, 1, 0, 0}
Output: 1 to 6
(Starting and Ending indexes of output subarray)

Input: arr[] = {1, 1, 1, 1}
Output: No such subarray

Input: arr[] = {0, 0, 1, 1, 0}
Output: 0 to 3 Or 1 to 4
```

Approach: The concept of taking **cumulative sum, taking 0's as -1** will help us in optimizing the approach. While taking the cumulative sum, there are two cases when there can be a sub-array with equal number of 0's and 1's.

1. When **cumulative sum=0**, which signifies that sub-array from index **(0)** till present index has equal number of **0's and 1's**.
2. When we encounter a cumulative sum value which we have already encountered before, which means that sub-array from the **previous index+1** till the **present index** has equal number of 0's and 1's as they give a **cumulative sum of 0** .

In a nutshell this problem is equivalent to finding two indexes **i & j in array[] such that array[i] = array[j] and (j-i) is maximum**. To store the first occurrence of each unique cumulative sum value we use a **hash_map** wherein if we get that value again we can find the sub-array size and compare it with the maximum size found till now.

Complexity

- **Time Complexity:** O(n). As the given array is traversed only once.
- **Auxiliary Space:** O(n). As **hash_map** has been used which takes extra space.