Longest Subarray with equal no. of 0's and 1's

Array → CurrentMax , maxSubarray = ~~2 4 6~~ 8

arr = { 1 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 1 }

PreSum = [1 2 3 2 3 2 1 0 1 2]

Idx → 0th 1st 2nd 3rd 4th 5th 6th 7th 8th 9th

PreSum	Index
1	0
2	1
3	2

CurrentMax = 9 − 1 = 8

maxSubarray = Max(8, 8) = 8

```
int maxLen(int arr[], int n)
   {
      // Your code here
      unordered_map<int , int> um;
      int pre_sum = 0 , longest_subarray = 0;
      for(int i = 0; i < n; i++)
      {
         pre_sum += arr[i] == 0 ? -1 : 1;

         if(pre_sum == 0 )
         {if(longest_subarray < i + 1)
         longest_subarray = i + 1;}

         else if(um.find(pre_sum) != um.end())
         {if(longest_subarray <  i - um[pre_sum]) longest_subarray = i - um[pre_sum];}

         else um[pre_sum] = i;
      }
      return longest_subarray;
   }
```

Current Subarray Size

Calculating Prefix Sum.

if Sum is 0
Max Subarray becomes Smaller than Current Subarray
Update the Max Subarray

if the prefix sum is available in map

if MaxSubarray is Smaller
then Size of Current Subarray
then, Update Max Subarray

if new element is found
then update hash table
with,
Key as Prefix Sum
and, Value as index.
Ex:
<Prefix Sum, index>