//time : O(N)

//space:O(N)

class Solution {

public int subarraySum(int[] nums, int k) {

HashMap<Integer,Integer> hmap = new HashMap<>();

int count=0;

int rsum=0,compliment=0;

if(nums.length==0 ||nums==null)

return 0;

hmap.put(0,1);

for(int i=0;i<nums.length;i++)

{

rsum+=nums[i];

compliment=rsum-k;

if(hmap.containsKey(compliment))

count+=hmap.get(compliment);

hmap.put(rsum,hmap.getOrDefault(rsum,0)+1);

}

return count;

}

}

class Solution {

public int findMaxLength(int[] nums) {

if(nums==null || nums.length==0)

return 0;

int max=0;int rsum=0;

HashMap<Integer,Integer> hmap = new HashMap<>();

hmap.put(0,-1);

for(int i=0;i<nums.length;i++){

if(nums[i]==1)

rsum+=nums[i];

else

rsum+=-1;

if(!hmap.containsKey(rsum))

hmap.put(rsum,i);

else

max=Math.max(max,i-hmap.get(rsum));

}

return max;

}

}

class Solution {

public int longestPalindrome(String s) {

if(s==null || s.length()==0)

return 0;

HashSet<Character> hs = new HashSet<>();

int result=0;

for(int i=0;i<s.length();i++){

char c=s.charAt(i);

if(hs.contains(c)){

result+=2;

hs.remove(c);

}

else{

hs.add(c);

}

}

if(!hs.isEmpty()) result++;

return result;

}

}