1590. Make Sum Divisible by P

Given an array of positive integers `nums`, remove the **smallest** subarray (possibly **empty**) such that the **sum** of the remaining elements is divisible by `p`. It is **not** allowed to remove the whole array.

Return *the length of the smallest subarray that you need to remove, or* `-1` *if it's impossible*.

A **subarray** is defined as a contiguous block of elements in the array.

Example 1:

```
Input: nums = [3,1,4,2], p = 6
Output: 1
Explanation: The sum of the elements in nums is 10, which is not divisible by 6. We can remove the subarray [4], and the sum of the remaining elements is 6, which is divisible by 6.
```

Example 2:

```
Input: nums = [6,3,5,2], p = 9
Output: 2
Explanation: We cannot remove a single element to get a sum divisible by 9. The best way is to remove the subarray [5,2], leaving us with [6,3] with sum 9.
```

Example 3:

```
Input: nums = [1,2,3], p = 3
Output: 0
Explanation: Here the sum is 6. which is already divisible by 3. Thus we do not need to remove anything.
```

Constraints:

- $1 <= nums.length <= 10^5$
- $1 <= nums[i] <= 10^9$
- $1 <= p <= 10^9$

1590. Make Sum Divisible by P

```
/*
    Brute force
    Time complexity: O(n+n^2) = O(n^2)
    Space complexity: O(1)
*/
class Solution{
    public:
        int minSubarray(vector<int>& nums, int p){
            int n=nums.size();
            int r=0;
            for(auto& e: nums) r=(r+e)%p;

            if(r==0) return 0;

            int ans=INT_MAX;
            for(int i=0;i<n;++i){
                long long sum=0;
                for(int j=i;j<n;++j){
                    sum+=nums[j];
                    if((sum-r)%p==0) ans=std::min(ans,j-i+1);
                }
            }
            if(ans==n) return -1;
            return ans==INT_MAX?-1:ans;
        }
};
```

1590. Make Sum Divisible by P

```
/*
   Prefix sums+hash map
   Time complexity: O(n)
   Space complexity: O(n)
*/
class Solution{
   public:
      int minSubarray(vector<int>& nums, int p){
         int n=nums.size();
         int r=0;
         for(auto& e: nums) r=(r+e)%p;

         if(r==0) return 0;

         std::unordered_map<int, int> prefix_indexes;
         prefix_indexes[0]=-1;
         long prefix_sum=0;
         int ans=INT_MAX;

         for (int i=0;i<n;++i) {
            prefix_sum=(prefix_sum+nums[i])%p;
            int test=(((prefix_sum-r)%p)+p)%p; // (prefix_sum-r)%p;
            if(prefix_indexes.find(test)!=prefix_indexes.end()) ans=std::min(ans,i-prefix_indexes[test]);
            prefix_indexes[prefix_sum]=i;
         }

         if(ans==n) return -1;
         return ans==INT_MAX?-1:ans;
      }
};
```