3. Longest Substring Without Repeating Characters

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        se = {}
        start = 0
        maxx_len = 0

        for i in range(len(s)):
            if s[i] in se and se[s[i]] >= start:
                start = se[s[i]] + 1
            se[s[i]] = i
            maxx_len = max(maxx_len, i - start + 1)

        return maxx_len
```

424. Longest Repeating Character Replacement

```
class Solution:
    def characterReplacement(self, s: str, k: int) -> int:
        count = {}
        res = 0

        l = 0
        maxf = 0
        for r in range(len(s)):
            count[s[r]] = count.get(s[r], 0) + 1
            maxf = max(maxf, count[s[r]])

            if (r - l + 1) - maxf > k:
                count[s[l]] -= 1
                l += 1
            res = max(res, r-l+1)
        return res
```

930. Binary Subarrays With Sum

```
from collections import defaultdict
from typing import List

class Solution:
    def numSubarraysWithSum(self, nums: List[int], goal: int) -> int:
        count = defaultdict(int)
        count[0] = 1
        prefix_sum = 0
        result = 0

        for num in nums:
            prefix_sum += num
            result += count[prefix_sum - goal]
            count[prefix_sum] += 1

        return result
```

1004. Max Consecutive Ones III

```
from typing import List

class Solution:
    def longestOnes(self, nums: List[int], k: int) -> int:
        left = 0
        zeros = 0
        max_len = 0

        for right in range(len(nums)):
            if nums[right] == 0:
                zeros += 1

            while zeros > k: #shirk important
                if nums[left] == 0:
                    zeros -= 1
                left += 1

            max_len = max(max_len, right - left + 1)

        return max_len
```

1248. Count Number of Nice Subarrays

```
from collections import defaultdict
class Solution:
    def numberOfSubarrays(self, nums: List[int], k: int) -> int:
        cnt = defaultdict(int)
        cnt[0] = 1
        prefix = 0
        res = 0
        for num in nums:
            if num%2 ==1:
                prefix+=1
            res += cnt[prefix-k ]
            cnt[prefix]+=1
        return res
```

1358. Number of Substrings Containing All Three Characters

```
from collections import defaultdict

class Solution:
    def numberOfSubstrings(self, s: str) -> int:
        count = defaultdict(int)
        total = 0
        left = 0

        for right in range(len(s)):
            count[s[right]] += 1

            while count['a'] > 0 and count['b'] > 0 and count['c'] > 0:
                total += len(s) - right
                count[s[left]] -= 1
                left += 1

        return total
```

1423. Maximum Points You Can Obtain from Cards

```python
from typing import List

class Solution:
    def maxScore(self, cardPoints: List[int], k: int) -> int:
        n = len(cardPoints)
        window_size = n - k
        total_sum = sum(cardPoints)

        # Find the minimum sum subarray of size n - k
        curr_sum = sum(cardPoints[:window_size])
        min_sub_sum = curr_sum

        for i in range(window_size, n):
            curr_sum += cardPoints[i] - cardPoints[i - window_size]
            min_sub_sum = min(min_sub_sum, curr_sum)

        return total_sum - min_sub_sum
```