1. Prefix Sum with Hashing

These problems use a hashmap (or dictionary) to optimize prefix sum calculations, especially for problems involving subarray sums or divisibility.

1. **Subarray Sum Equals K**
    - Problem: Given an array of integers, find the total number of subarrays with a sum equal to k.
    - Example:
        - Input: $nums = [1, 1, 1]$, $k = 2$
        - Output: 2 (subarrays $[1, 1]$ and $[1, 1]$)
2. **Continuous Subarray Sum**
    - Problem: Given an array of integers, check if there is a subarray of size at least 2 with a sum that is a multiple of k.
    - Example:
        - Input: $nums = [23, 2, 4, 6, 7]$, $k = 6$
        - Output: $True$ (subarray $[2, 4]$ sums to 6)
3. **Subarray Sum Divisible by K**
    - Problem: Given an array of integers, find the total number of subarrays with a sum divisible by k.
    - Example:
        - Input: $nums = [4, 5, 0, -2, -3, 1]$, $k = 5$
        - Output: 7
4. **Longest Subarray with Sum Divisible by K**
    - Problem: Given an array of integers, find the length of the longest subarray with a sum divisible by k.
    - Example:
        - Input: $nums = [2, 7, 6, 1, 4, 5]$, $k = 3$
        - Output: 4 (subarray $[7, 6, 1, 4]$)

---

2. Prefix Sum in 2D Arrays

These problems involve applying prefix sums to 2D arrays (matrices) to efficiently compute sums of rectangular regions.

5. **Range Sum Query 2D - Immutable**

- Problem: Given a 2D matrix, answer multiple queries to find the sum of elements in a rectangular region.
- Example:
  - Input: $\text{matrix} = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]$, queries = $[(0, 0, 1, 1), (1, 1, 2, 2)]$
  - Output: $[12, 28]$ (sum of $[[1, 2], [4, 5]]$ and $[[5, 6], [8, 9]]$)

6. **Number of Submatrices with Sum K**
   - Problem: Given a 2D matrix, find the number of submatrices with a sum equal to k.
   - Example:
     - Input: $\text{matrix} = [[0, 1, 0], [1, 1, 1], [0, 1, 0]]$, $k = 0$
     - Output: 4
7. **Maximum Side Length of a Square with Sum Less Than or Equal to Threshold**
   - Problem: Given a 2D matrix and a threshold, find the maximum side length of a square submatrix with a sum ≤ threshold.
   - Example:
     - Input: $\text{matrix} = [[1, 1, 1], [1, 1, 1], [1, 1, 1]]$, $\text{threshold} = 4$
     - Output: 2 (square of size 2x2)

---

3. Prefix Sum with Sliding Window

These problems combine prefix sums with the sliding window technique to solve problems involving subarrays or substrings.

8. **Minimum Size Subarray Sum**
   - Problem: Given an array of positive integers, find the length of the smallest subarray with a sum ≥ k.
   - Example:
     - Input: $\text{nums} = [2, 3, 1, 2, 4, 3]$, $k = 7$
     - Output: 2 (subarray $[4, 3]$)
9. **Maximum Sum of Two Non-Overlapping Subarrays**
   - Problem: Given an array, find the maximum sum of two non-overlapping subarrays of lengths L and M.
   - Example:
     - Input: $\text{nums} = [0, 6, 5, 2, 2, 5, 1, 9, 4]$, $L = 1$, $M = 2$
     - Output: 20 (subarrays $[9]$ and $[6, 5]$)
10. **Maximum Sum of Three Non-Overlapping Subarrays**

- Problem: Given an array, find the maximum sum of three non-overlapping subarrays of size k.
- Example:
  - Input: $\text{nums} = [1, 2, 1, 2, 6, 7, 5, 1]$, $k = 2$
  - Output: 23 (subarrays $[1, 2]$, $[6, 7]$, $[5, 1]$)

---

4. Prefix Sum with Binary Search

These problems involve using prefix sums with binary search for optimization.

11. **Minimum Limit of Balls in a Bag**
    - Problem: Given an array of balls and a maximum number of operations, find the minimum possible maximum number of balls in any bag after performing the operations.
    - Example:
      - Input: $\text{nums} = [9]$, $\text{maxOperations} = 2$
      - Output: 3
12. **Split Array Largest Sum**
    - Problem: Given an array of integers, split it into m subarrays such that the largest sum among the subarrays is minimized.
    - Example:
      - Input: $\text{nums} = [7, 2, 5, 10, 8]$, $m = 2$
      - Output: 18 (subarrays $[7, 2, 5]$ and $[10, 8]$)

---

5. Prefix Sum with Frequency Maps

These problems involve maintaining a frequency map or hashmap to track elements or characters.

13. **Count Number of Nice Subarrays**
    - Problem: Given an array of integers, count the number of subarrays with exactly k odd numbers.
    - Example:
      - Input: $\text{nums} = [1, 1, 2, 1, 1]$, $k = 3$
      - Output: 2
14. **Longest Subarray with At Most K Distinct Elements**

- Problem: Given an array of integers, find the length of the longest subarray with at most k distinct elements.
- Example:
  - Input: $\text{nums} = [1, 2, 1, 2, 3]$, $k = 2$
  - Output: 4 (subarray $[1, 2, 1, 2]$)

---

6. Prefix Sum with Circular Arrays

These problems involve applying prefix sums to circular arrays (where the end of the array wraps around to the beginning).

15. **Maximum Sum Circular Subarray**
    - Problem: Given a circular array, find the maximum sum of any subarray.
    - Example:
      - Input: $\text{nums} = [5, -3, 5]$
      - Output: 10 (subarray $[5, -3, 5]$ wrapping around)
16. **Minimum Sum Circular Subarray**
    - Problem: Given a circular array, find the minimum sum of any subarray.
    - Example:
      - Input: $\text{nums} = [1, -2, 3, -2]$
      - Output: -2 (subarray $[-2]$)

---

7. Prefix Sum with Multiple Arrays

These problems involve computing prefix sums across multiple arrays or dimensions.

17. **Range Sum Query - Mutable**
    - Problem: Given an array, answer multiple queries to find the sum of elements between indices i and j, while allowing updates to the array.
    - Example:
      - Input: $\text{nums} = [1, 3, 5]$, queries $= [(0, 2), \text{update}(1, 2), (0, 2)]$
      - Output: $[9, 8]$
18. **Prefix Sum with Difference Arrays**
    - Problem: Given an array and a list of range updates, compute the final array after all updates.
    - Example:

- Input: $\text{nums} = [1, 2, 3, 4, 5]$, updates = $[(1, 3, 2), (0, 2, -1)]$
- Output: $[0, 3, 4, 6, 5]$

---

8. Miscellaneous Prefix Sum Problems

19. **Product of Array Except Self**
    - Problem: Given an array, return an array where each element is the product of all elements except itself.
    - Example:
      - Input: $\text{nums} = [1, 2, 3, 4]$
      - Output: $[24, 12, 8, 6]$
20. **Maximum Points You Can Obtain from Cards**
    - Problem: Given an array of card points, pick k cards from either end to maximize the total points.
    - Example:
      - Input: $\text{nums} = [1, 2, 3, 4, 5, 6, 1]$, $k = 3$
      - Output: 12 (cards $[1, 6, 5]$)