Грокаем алгособесы: Префиксная сумма

Новая рубрика для разработчиков: разбираем ключевые паттерны алгоритмических собеседований

Почему важно знать паттерны алгоритмов?

Универсальные шаблоны

Алгособесы проверяют не знание конкретных задач, а умение применять универсальные шаблоны решений к различным проблемам

Скорость и эффективность

Понимание паттернов позволяет решать множество задач быстрее, находить оптимальные решения и писать более чистый код

Уверенность на интервью

Знание паттернов экономит время подготовки, снижает стресс перед собеседованием и повышает шансы на успех

Что такое паттерны в алгоритмах?

Суть подхода

Паттерны — это повторяющиеся подходы к решению задач с похожей структурой. Они представляют собой проверенные временем стратегии, которые можно адаптировать под конкретные условия.

Освоив основные паттерны, вы получаете мощный инструментарий для решения широкого спектра алгоритмических задач.

Популярные паттерны

Prefix Sum — предподсчет сумм

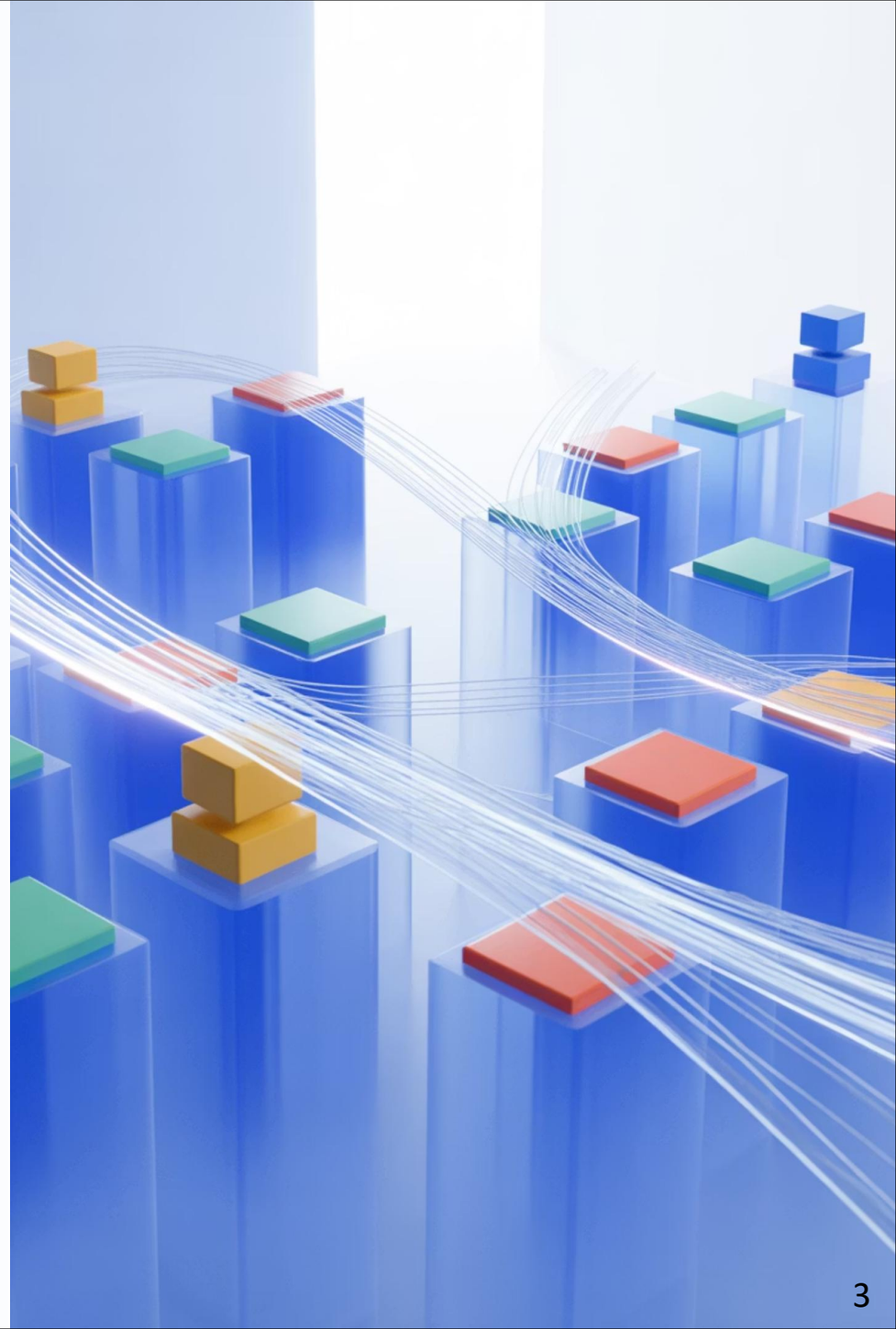
Two Pointers — работа с двумя указателями

Sliding Window — скользящее окно

DFS/BFS — обход графов

Dynamic Programming — динамическое программирование

Binary Search — бинарный поиск



Паттерн Prefix Sum — что это?

01

Определение

Префиксные суммы — это массив, где каждый элемент хранит сумму всех элементов исходного массива от начала до текущего индекса включительно

02

Производительность

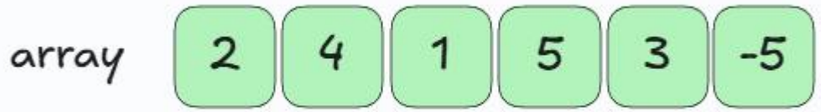
Позволяет быстро вычислять сумму любого подотрезка за O(1) после предварительной обработки за O(n)

03

Применение

Часто используется для задач с подсчетом сумм на диапазонах, вычислением частот элементов и оптимизацией запросов

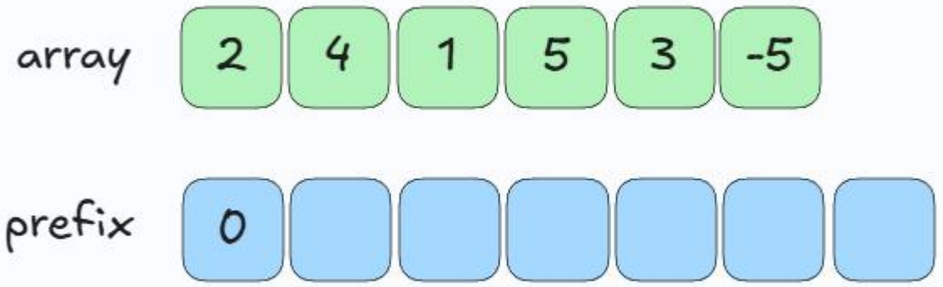
Как построить префиксный массив?



Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1,
где `prefix[0] = 0`



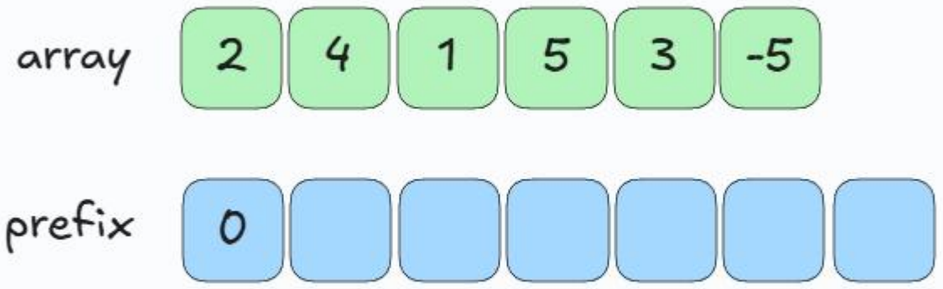
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



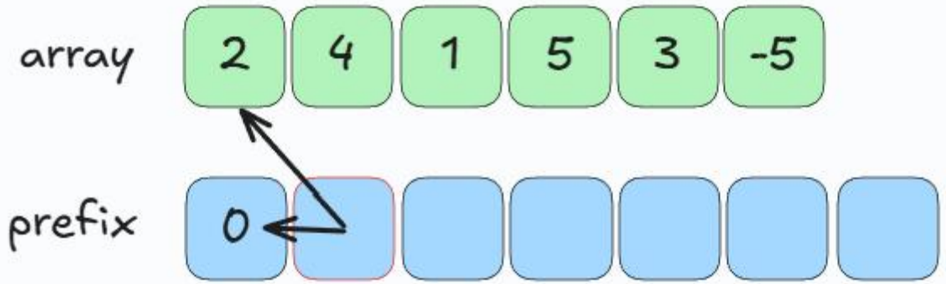
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



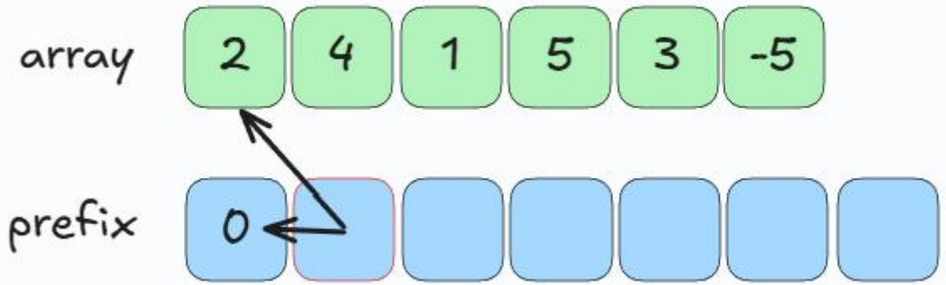
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



Как построить префиксный массив?

Шаг 1: Инициализация

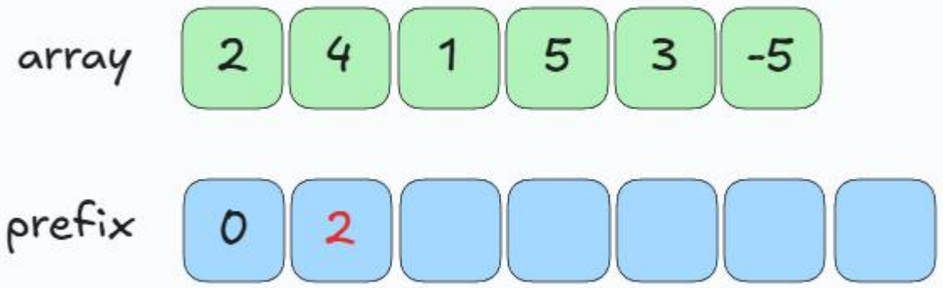
Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`

Шаг 3: Запросы

Сумма на отрезке [l, r]: `prefix[r+1] - prefix[l]`



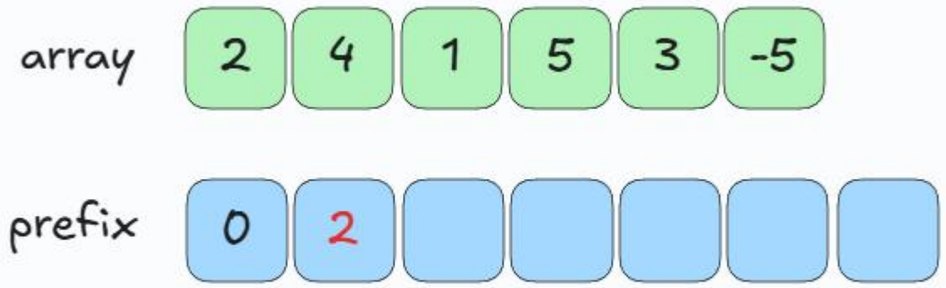
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



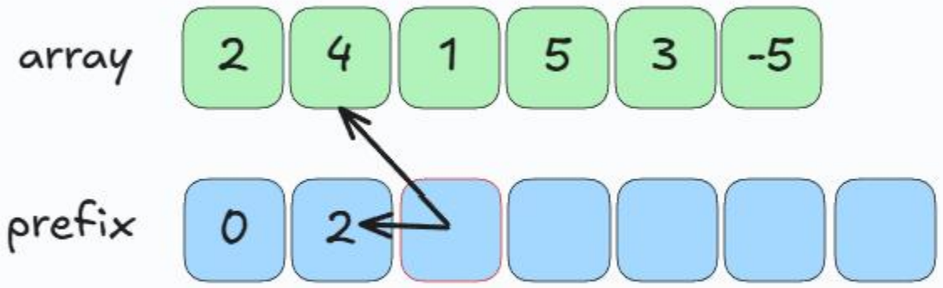
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



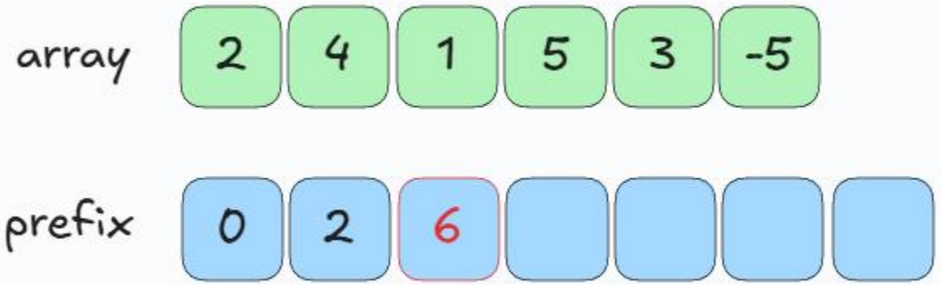
Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`



Как построить префиксный массив?

Шаг 1: Инициализация

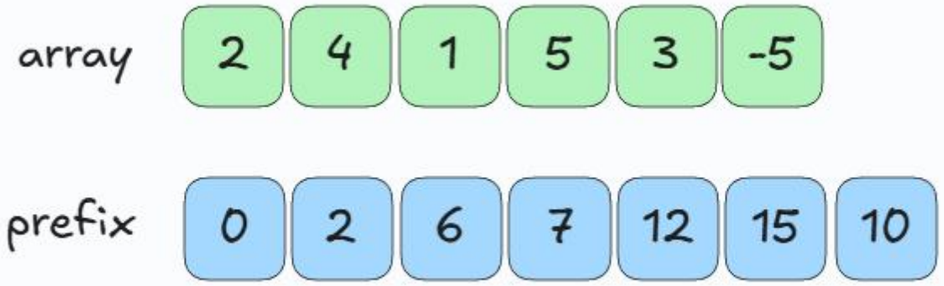
Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`

Шаг 3: Запросы

Сумма на отрезке [l, r]: `prefix[r+1] - prefix[l]`



Как построить префиксный массив?

Шаг 1: Инициализация

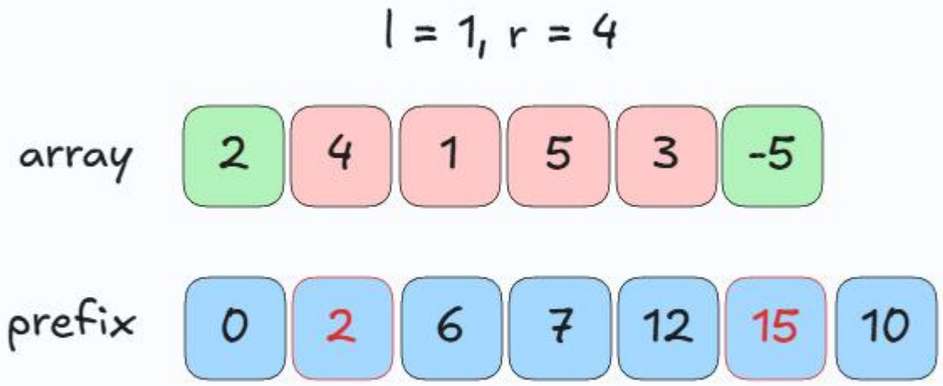
Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`

Шаг 3: Запросы

Сумма на отрезке [l, r]: `prefix[r+1] - prefix[l]`



Как построить префиксный массив?

Шаг 1: Инициализация

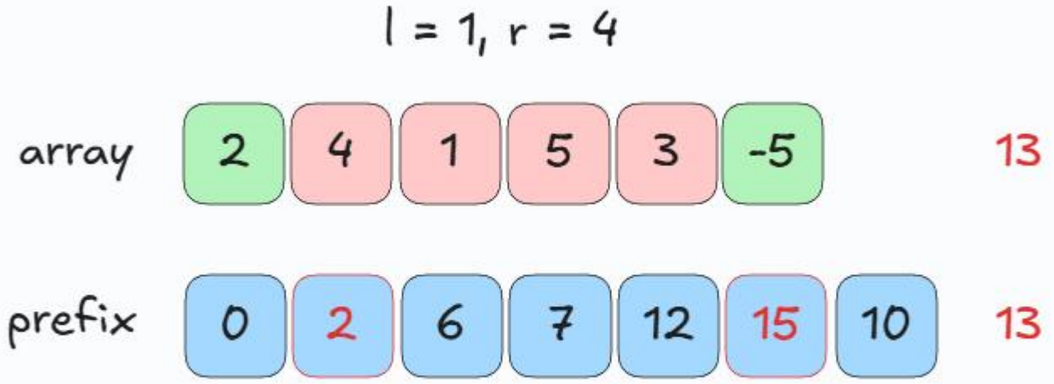
Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`

Шаг 3: Запросы

Сумма на отрезке [l, r]: `prefix[r+1] - prefix[l]`



Как построить префиксный массив?

Шаг 1: Инициализация

Создаем массив `prefix` длины n+1, где `prefix[0] = 0`

Шаг 2: Заполнение

Для каждого i от 1 до n вычисляем: `prefix[i] = prefix[i-1] + nums[i-1]`

Шаг 3: Запросы

Сумма на отрезке [l, r]: `prefix[r+1] - prefix[l]`

Пример массива

```
nums = [2, 4, 1, 5, 3, -5]
```

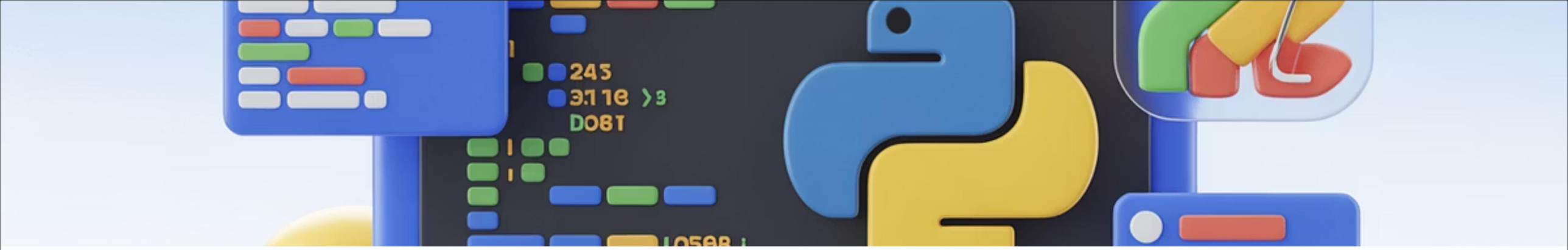
```
prefix = [0, 2, 6, 7, 12, 15, 10]
```

Запрос суммы [1:4]

```
sum = prefix[5] - prefix[1]
```

```
sum = 15 - 2 = 13
```

✓ Результат: 4 + 1 + 5 + 3 = 13



Пример задачи с LeetCode

Range Sum Query - Immutable (№303)

Условие: Дан массив целых чисел nums. Реализуйте класс NumArray, который поддерживает множественные запросы суммы элементов на произвольных отрезках [left, right].

Требования: Метод sumRange(left, right) должен возвращать сумму элементов массива между индексами left и right включительно.

Наивный подход

Каждый раз проходить по массиву и суммировать элементы

Сложность: O(n) на запрос

Оптимальное решение

Построить prefix sum один раз и отвечать на запросы мгновенно

Сложность: O(1) на запрос

Пример кода на Python

```python
class NumArray:
    def __init__(self, nums):
        """
        Инициализирует объект с целочисленным массивом nums.
        Временная сложность: O(n) для предварительной обработки
        Пространственная сложность: O(n) для хранения префиксных сумм
        """
        # Создаем массив префиксных сумм, начинающийся с 0
        # Это упрощает обработку случаев, когда left = 0
        self.prefix_sums = [0]
        for num in nums:
            # Каждый следующий элемент - это сумма предыдущей префиксной суммы и текущего числа
            self.prefix_sums.append(self.prefix_sums[-1] + num)

    def sumRange(self, left, right):
        """
        Возвращает сумму элементов между индексами left и right включительно.
        Временная сложность: O(1) на каждый запрос
```

O(n)

Инициализация

Построение префиксного массива

O(1)

Запрос

Время выполнения sumRange

O(n)

Память

Дополнительное пространство

Спасибо за внимание!

- Увидимся в следующих выпусках серии "Грокаем алгособесы"
- **Следующая серия:** Разберем паттерн Two Pointers и решим задачи на поиск пар и триплетов!

Подписывайтесь!

- **Telegram-канал:** t.me/marat_notes
- **Обучающие видео:**
  - https://vkvideo.ru/@club231048746
  - https://www.youtube.com/@marat_notes
- **Репозитарий:**
  - https://github.com/MaratNotes/marat_notes

Призыв к действию:

- Практикуйтесь на LeetCode
- Применяйте изученные паттерны
- Готовьтесь к собеседованиям уверенно