Грокаем алгособесы: Паттерн Fast and Slow

Находим циклы в связном списке

Что такое паттерн «Быстрый и медленный указатели»?

Определение

Техника, при которой два указателя проходят по структуре данных (чаще всего связному списку) с разной скоростью. Этот подход позволяет решать сложные задачи с минимальными затратами памяти.

Медленный указатель — движется на 1 шаг за итерацию

Быстрый указатель — движется на 2 шага за итерацию

Принцип работы

Механика паттерна основана на простой математической закономерности:

- Если в структуре есть цикл → указатели обязательно встретятся внутри цикла
- Если цикла нет → быстрый указатель достигнет конца структуры

Эта особенность делает паттерн невероятно эффективным для детекции циклов.

Ключевые преимущества подхода

Временная сложность

O(n) — линейное время выполнения

Алгоритм проходит по структуре данных не более одного-двух раз, что обеспечивает оптимальную производительность

Пространственная сложность

O(1) — константная память

Используются только два указателя, независимо от размера входных данных. Не требуется дополнительных структур данных

Без дополнительного хранения

Не требует хеш-таблиц, множеств или массивов для отслеживания посещённых элементов

Когда применять этот паттерн?

✓ Применяется, когда:

Обнаружение циклов

Нужно определить наличие цикла в связном списке — классический сценарий использования паттерна

Поиск середины

Необходимо найти средний элемент списка за один проход без предварительного подсчёта длины

Свести задачу к модели связного списка

Найти дубликат в массиве с числами от 1 до n без модификации данных и extra memory

✗ Не подходит, если:

Нелинейные структуры

Структура данных не линейная (Найти общий узел в двух бинарных деревьях)

Задачи, требующие хранения истории

Пример: "Найти первую повторяющуюся букву в строке"

Динамические задачи

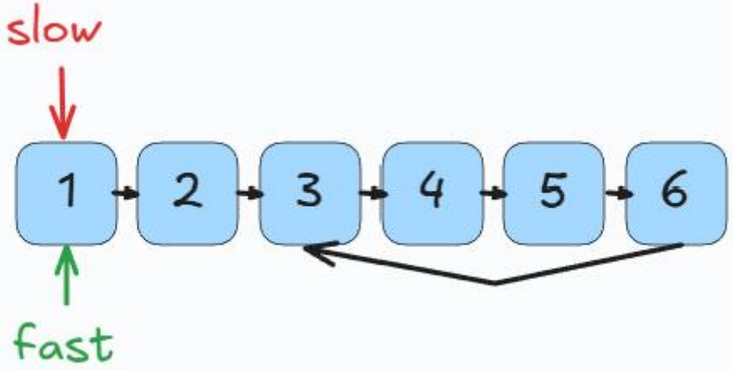
Пример: "Максимальная сумма подмассива с ограничениями"

Визуализация работы алгоритма

01

Инициализация

Оба указателя стартуют с головы списка



Алгоритм встречного движения

01

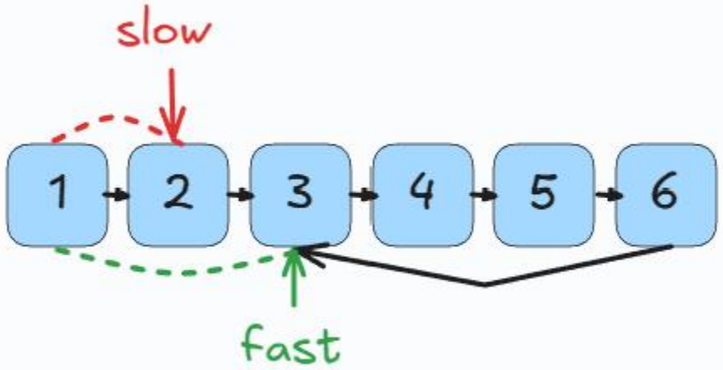
Инициализация

Устанавливаем left = 0, right = n-1 (начало и конец массива)

02

Движение

Slow += 1 шаг, **Fast** += 2 шага на каждой итерации



Алгоритм встречного движения

01

Инициализация

Устанавливаем left = 0, right = n-1 (начало и конец массива)

02

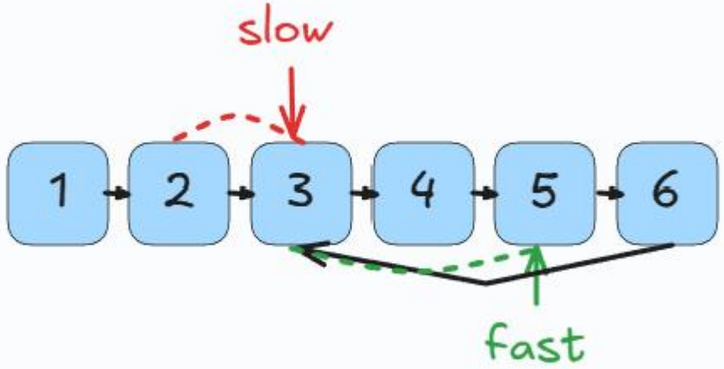
Движение

Slow += 1 шаг, **Fast** += 2 шага на каждой итерации

03

Проверка условия

Если **slow == fast**, обнаружен цикл



Алгоритм встречного движения

01

Инициализация

Устанавливаем left = 0, right = n-1 (начало и конец массива)

02

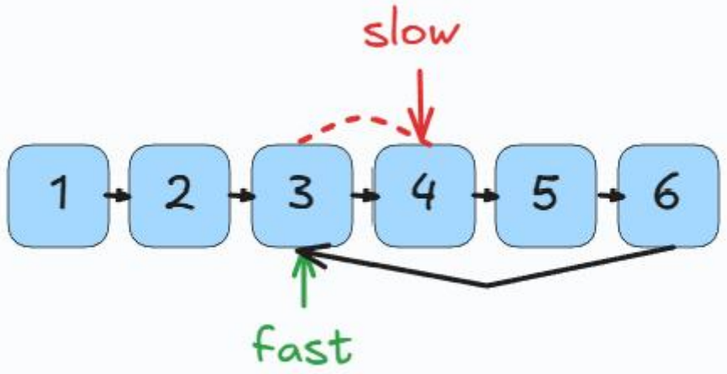
Движение

Slow += 1 шаг, **Fast** += 2 шага на каждой итерации

03

Проверка условия

Если **slow == fast**, обнаружен цикл



Алгоритм встречного движения

01

Инициализация

Устанавливаем left = 0, right = n-1 (начало и конец массива)

02

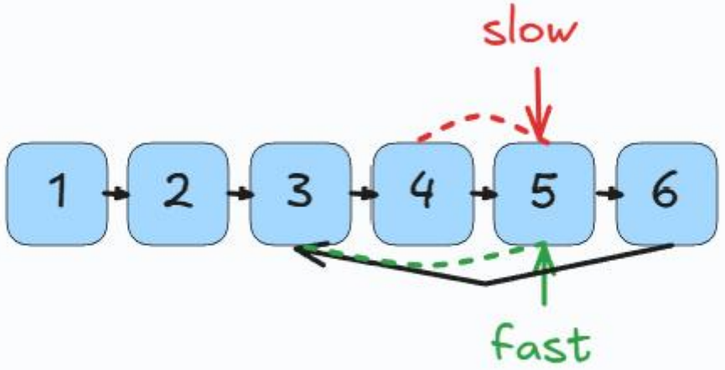
Движение

Slow += 1 шаг, **Fast** += 2 шага на каждой итерации

03

Проверка условия

Если **slow == fast**, обнаружен цикл



Алгоритм встречного движения

01

Инициализация

Устанавливаем left = 0, right = n-1 (начало и конец массива)

02

Движение

Slow += 1 шаг, **Fast** += 2 шага на каждой итерации

03

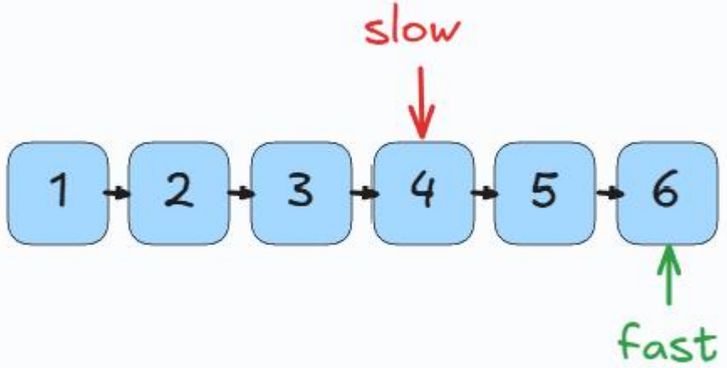
Проверка условия

Если **slow == fast**, обнаружен цикл

04

Завершение

Fast достиг конца или указатели встретились



Практические советы по применению

Проверяйте null

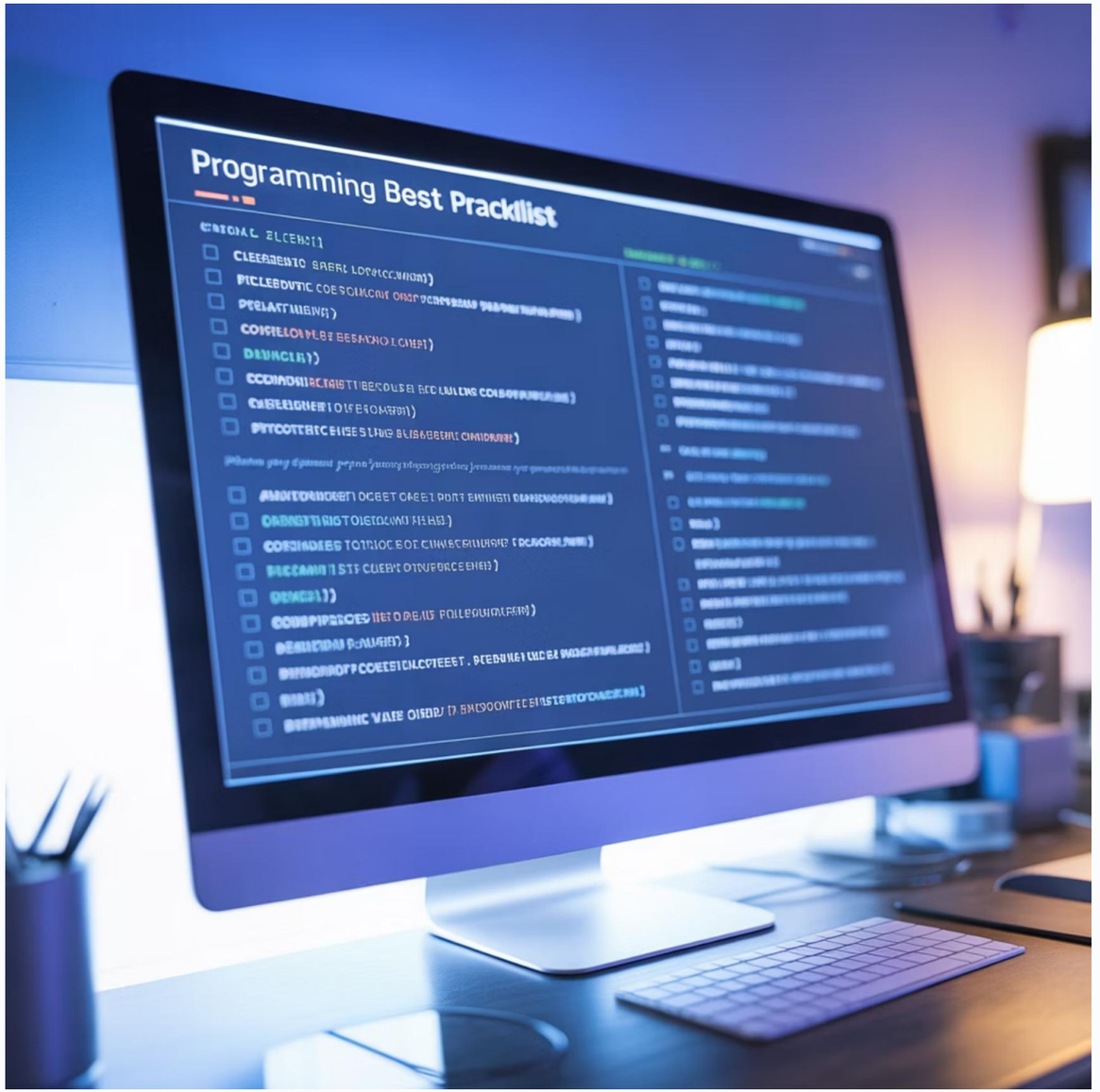
Всегда проверяйте, что `fast` и `fast.next` не равны None перед переходом

Начальная позиция

Оба указателя обычно стартуют с головы списка, но в некоторых задачах (например, поиск n-го элемента с конца) fast может стартовать с отступом

Скорость движения

Классическое соотношение 1:2, но в редких случаях может быть 1:3 или другое в зависимости от специфики задачи



Пример задачи: Cycle Detection (LeetCode №141)

Условие задачи

Определить, содержит ли связный список цикл. Функция должна вернуть `true`, если в списке есть узел, на который можно вернуться, следуя по указателям, и `false` в противном случае.

Наивный подход

Хранить все посещённые узлы в set и проверять каждый новый узел на наличие в множестве.

Время: **O(n)**

Память: **O(n)**

Решение через Fast/Slow

```
def hasCycle(self, head):
    slow = fast = head

    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next

        if slow == fast:
            return True

    return False
```

O(n)

Временная сложность

Линейный проход по списку

O(1)

Пространственная сложность

Только два указателя

Другие популярные задачи с этим паттерном

01

Middle of the Linked List (LeetCode №876)

Найти средний элемент связного списка за один проход. Когда быстрый указатель достигнет конца, медленный указатель окажется ровно в середине списка.

02

Find the Duplicate Number (LeetCode №287)

Найти дубликат в массиве чисел от 1 до n. Ключевая идея — интерпретировать массив как связный список, где значение элемента указывает на следующий индекс, создавая цикл в месте дубликата.

03

Linked List Cycle II (LeetCode №142)

Найти узел, с которого начинается цикл в связном списке. Требует двухэтапного решения: сначала обнаружение цикла, затем определение точки входа с использованием математических свойств.

Математика за паттерном

1 Встреча в цикле

Когда быстрый указатель входит в цикл, он начинает догонять медленный. Разница в скорости составляет 1 шаг за итерацию, поэтому встреча неизбежна.

2 Расстояние до встречи

Если длина цикла C, а быстрый указатель на K шагов впереди медленного, они встретятся через (C - K) итераций.

3 Нахождение начала цикла

После встречи, если переместить один указатель в начало и двигать оба по 1 шагу, они встретятся в точке входа в цикл — элегантное математическое свойство паттерна.

Интересный факт: Этот алгоритм известен как «алгоритм Флойда» или «алгоритм черепахи и зайца» (Floyd's Cycle-Finding Algorithm), названный в честь Роберта Флойда, который его популяризировал.

Заключение и следующие шаги

Практикуйся регулярно

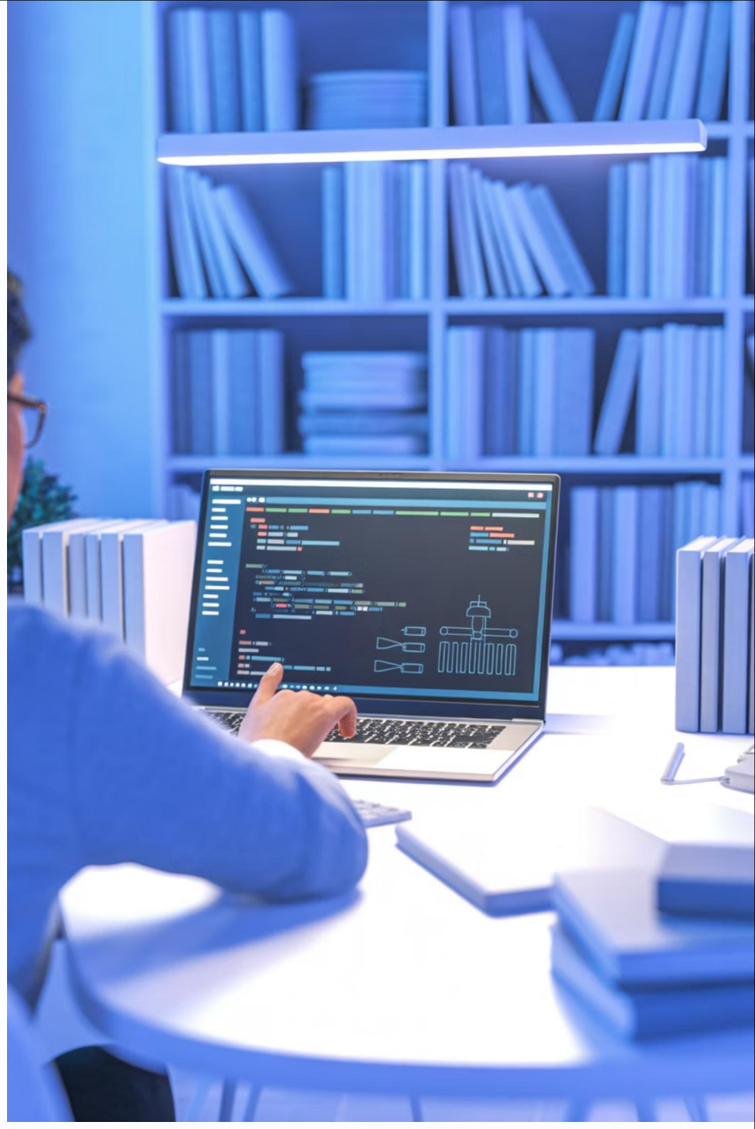
Решай задачи на LeetCode по тегу «Two Pointers» и «Linked List». Начни с простых задач и постепенно переходи к более сложным.

Понимай паттерны

Не заучивай решения наизусть — разбирайся в математике и логике за каждым паттерном. Это позволит адаптировать подход к новым задачам.

Следи за обновлениями

В следующих выпусках «Грокаем алгособесы» мы разберём другой важный паттерн: **Sliding Window**



Подписывайся и оставайся на связи

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