

# Грокаем алгособесы: Разворот списка

Как развернуть список за  $O(n)$  по времени и  $O(1)$  по памяти

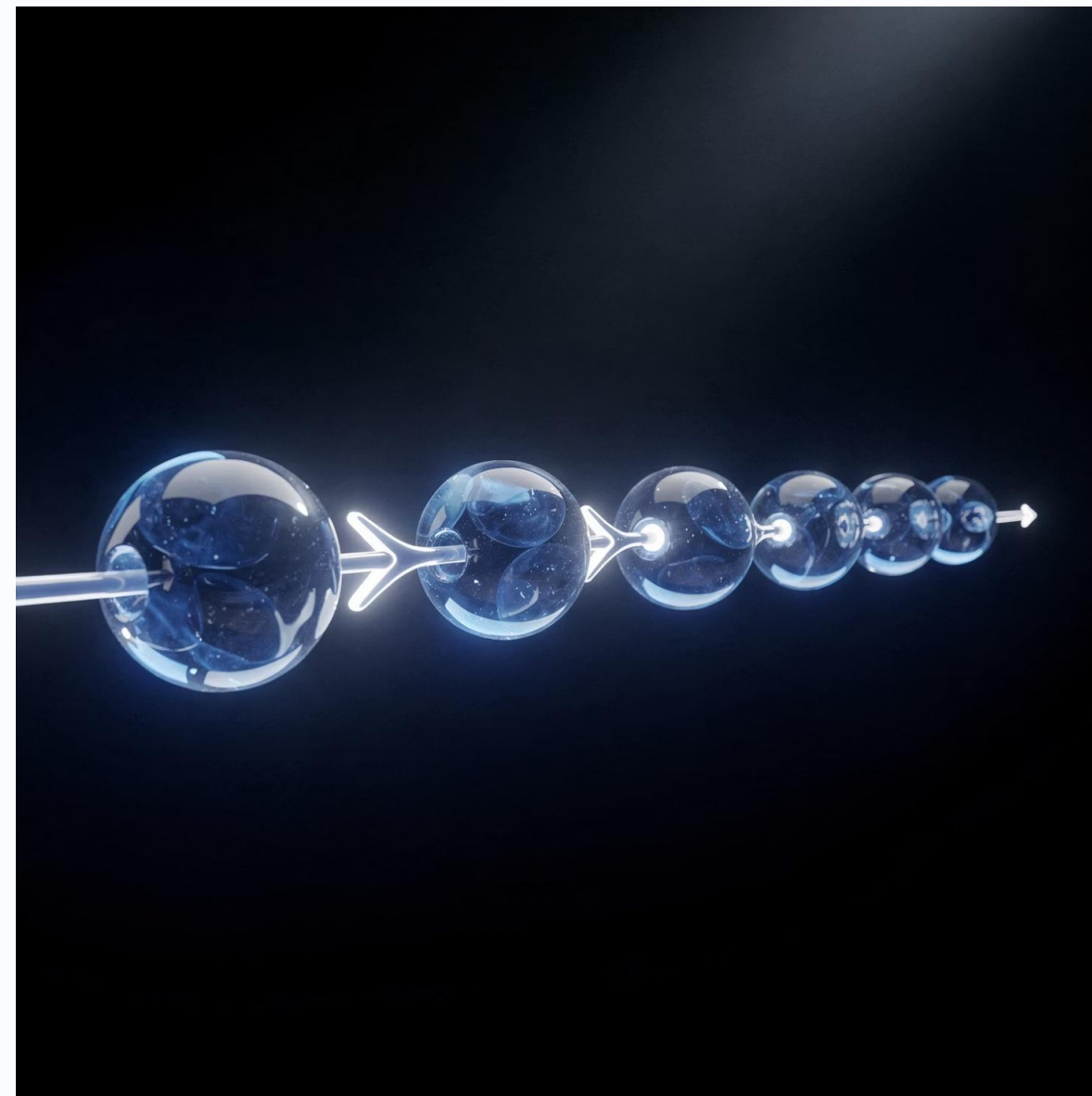
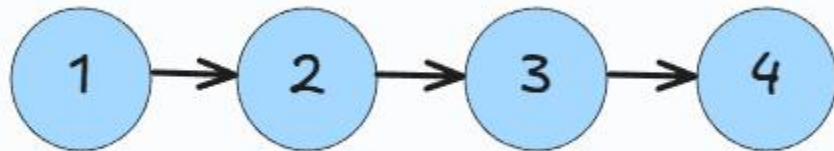


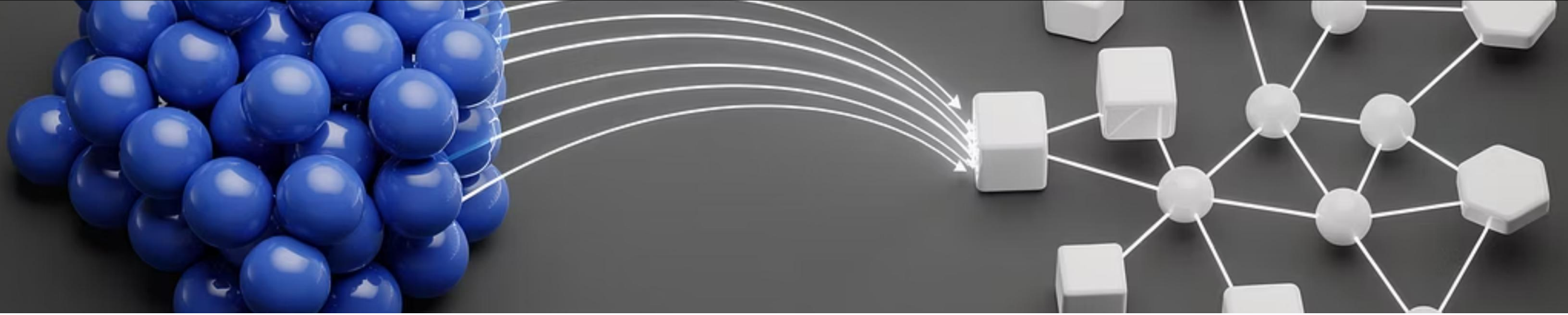
# Что такое связный список?

Это динамическая структура данных, состоящая из узлов. Каждый узел содержит данные и ссылку на следующий узел (односвязный список) или на следующий и предыдущий узлы (двусвязный список).

Основные преимущества:

- Эффективная вставка и удаление элементов без сдвига остальных данных
- Динамическое изменение размера структуры
- Гибкость при работе с памятью





# Формулировка задачи

## Условие

Дан **head** односвязного списка. Необходимо вернуть голову развёрнутого списка.

## Требования к решению:

Временная сложность:  **$O(n)$**  — один проход по списку

Пространственная сложность:  **$O(1)$**  — разворот **in-place** без дополнительной памяти

- Нельзя создавать новые узлы — только переставляем ссылки

## Пример работы

**Вход:**  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \text{None}$

**Выход:**  $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow \text{None}$



ЧАСТО НА СОБЕСАХ

БАЗОВАЯ ЗАДАЧА

## Зачем это знать?

### Проверка фундамента

Задача показывает ваше понимание ссылок, указателей и управления памятью. Без этого невозможно решать более сложные алгоритмические проблемы.

### База для сложных задач

Разворот списка — это фундамент для задач на палиндромы, разворот подписков, работу с K-группами и циклическими списками.

# Интуиция решения

## **Метафора моста**

Представьте, что вы идёте по мосту из досок — и переворачиваете каждую доску за собой, меняя её направление. Именно так мы и будем работать со списком!



### **Движение вперёд**

Идём по списку слева направо, посещая каждый узел по очереди



### **Разворот стрелок**

Меняем направление ссылки текущего узла на предыдущий



### **Сохранение связи**

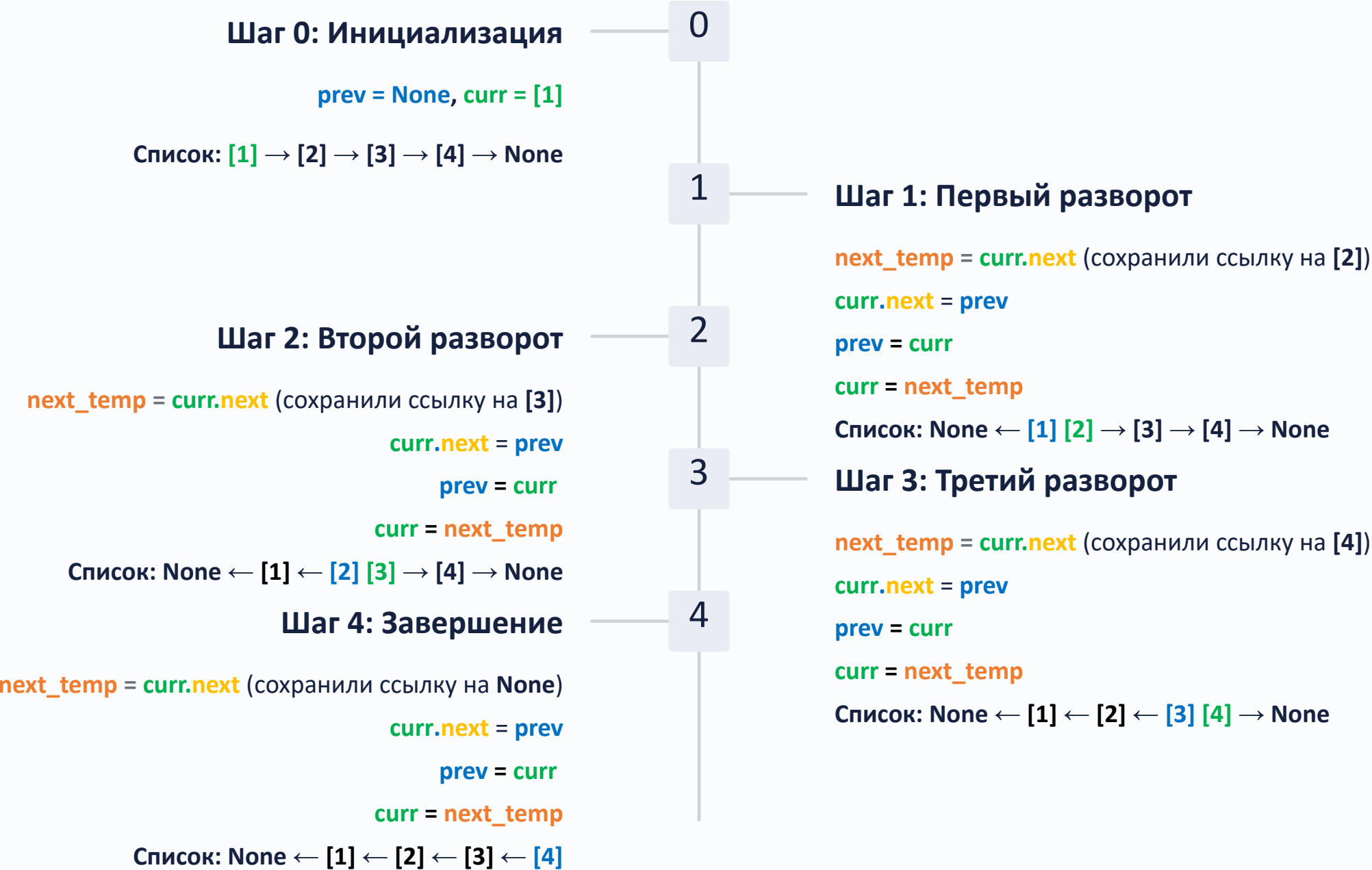
Обязательно сохраняем следующий узел перед изменением ссылки

Ключевая идея: мы меняем направление стрелок *по ходу движения*, поэтому нам не нужна дополнительная память для хранения узлов.



# Пошаговая визуализация

Давайте проследим за тремя указателями: **prev** (предыдущий), **curr** (текущий) и **next\_temp** (следующий).



# LeetCode #206: Reverse Linked List

Учитывая заголовок односвязного списка, разверните список и верните развернутый список.

EASY

LIKED LIST



# LeetCode #206: Reverse Linked List

```
1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
8         prev = None
9         curr = head
10        while curr:
11            next_temp = curr.next
12            curr.next = prev
13            prev = curr
14            curr = next_temp
15        return prev
16
17
18
```

Временная сложность:  $O(n)$  — один проход по всем узлам списка

Пространственная сложность:  $O(1)$  — используем только три указателя

# LeetCode #206: Reverse Linked List

```
1 # Definition for singly-linked list.
2 # class ListNode:
3 #     def __init__(self, val=0, next=None):
4 #         self.val = val
5 #         self.next = next
6 class Solution:
7     def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
8         prev = None
9         curr = head
10        while curr:
11            next_temp = curr.next
12            curr.next = prev
13            prev = curr
14            curr = next_temp
15        return prev
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

## Ошибка 1: Потеря ссылки

Изменение `curr.next` до сохранения следующего узла в `next_temp`. Результат: теряем весь оставшийся список и получаем бесконечный цикл или обрыв.

```
curr.next = prev # Ой!
next_temp = curr.next # Уже поздно!
```

## Ошибка 2: Неправильный return

Возврат `curr` вместо `prev` в конце функции. Когда цикл заканчивается, `curr` равен `None`, а `prev` указывает на новую голову списка.

```
return curr # Вернёт None!
return prev #
```



# Что решать дальше?

Освоили базовый разворот? Пора переходить к более сложным вариациям этой задачи!

01

## **LeetCode #92: Reverse Linked List II**

Разворот части списка между позициями m и n.  
Нужно развернуть только под список, сохранив остальную часть без изменений.

02

## **LeetCode #25: Reverse Nodes in k-Group**

Разворот узлов группами по K элементов. Если в последней группе меньше K элементов — оставляем их как есть.

03

## **Leetcode #234: Palindrome Linked List**

Разворот списка - ключевой шаг в решении задачи, если разворачивать с середины и сравнивать.

# Что запомнить



## Сохраняйте `next` ДО изменения

Возможная ошибка: Изменить `curr.next` до сохранения следующего узла. Потеряете весь список!



## Три указателя — золотой стандарт

`prev`, `curr`, `next_temp` — классический паттерн для разворота. Запомните эту тройку!



## Возвращайте `prev` как новый `head`

Когда цикл завершается, `curr = None`, а `prev` указывает на последний узел — новую голову списка.



# Как понять, что нужен разворот списка?

Ищи эти признаки в условии задачи:

## Односвязный список + обратный порядок

Если можно свести задачу к этому контексту.

## Сравнение начала и конца списка

Например, в задачах на палиндромы. Можно оптимально решить через разворот второй половины.

## Изменение направления связей без новых узлов

По сути, переносимся к первому варианту - есть обратный порядок, можем использовать

💡 Совет: Если видишь комбинацию «linked list» + « $O(1)$  space» + «reverse/backward» — это разворот списка!

# Подписывайся и оставайся на связи

Присоединяйся к сообществу, где разбирают не только задачи, но и контекст:

- как выбирать технологии в продакшене,
- как думать на собеседованиях,
- и как не выгорать, оставаясь профессионалом.

## Telegram-канал

[t.me/marat\\_notes](https://t.me/marat_notes)

## Репозиторий с кодом

[github.com/MaratNotes/marat\\_notes](https://github.com/MaratNotes/marat_notes)

## Видео

[youtube.com/@marat\\_notes](https://youtube.com/@marat_notes)

<https://vkvideo.ru/@club231048746>

### Не только про код:

- Алгособесы и data-инженерия (Airflow, Spark, Kafka)
- Разборы когнитивных искажений в IT («Ошибки мышления»)
- Разборы интересных выступлений с конференций и статей («ИТИнсайты»)
- Мысли о балансе работы, отдыха и профессионального роста

Код ко всем разборам — с пояснениями и production-подходом.

### Две рубрики:

- «Грокаем алгособесы» — паттерны, LeetCode, логика на интервью
- «Как работают данные» — продакшен-кейсы из реальных проектов



## В следующем выпуске:

В следующем видео поговорим про монотонный стек, простой приём, который помогает быстро решать задачи вроде:

«найти для каждого элемента ближайший больший справа»