

Билет 17. Быстрая сортировка

Основная идея: Выбрали pivot, навели бардак, поставили pivot на место, повторили для частей. Profit!

Три вида разбиения

1. Разбиение Ломуто (простое)

Алгоритм:

1. Выбираем pivot (обычно последний элемент)
2. i = начало
3. Идем j от начала до предпоследнего:
 - Если $arr[j] < pivot$, меняем $arr[i]$ и $arr[j]$, $i++$
4. Меняем $arr[i]$ и pivot
5. Возвращаем i

Пример: [3, 1, 4, 1, 5, 9, 2, 6] с pivot = 6

После разбиения: [3, 1, 4, 1, 5, 2, 6, 9]

↑
↑
все < 6
pivot все > 6

2. Разбиение Хоара

Алгоритм:

1. Выбираем pivot (обычно первый или средний)
2. l идет слева, r идет справа
3. Пока $l \leq r$:
 - Ищем слева элемент $\geq pivot$
 - Ищем справа элемент $\leq pivot$
 - Если нашли оба - меняем их местами
4. Возвращаем r (граница между частями)

Пример: [5, 3, 8, 4, 2, 7, 1, 6] с pivot = 5

l находит 8 (≥ 5), r находит 1 (≤ 5) → swap
[5, 3, 1, 4, 2, 7, 8, 6]

l находит 7 (≥ 5), r находит 2 (≤ 5) \rightarrow swap
[5, 3, 1, 4, 2, 7, 8, 6] - упс, $l > r$, стоп

Итог: [2, 3, 1, 4, 5, 7, 8, 6]
 \uparrow \uparrow
 все ≤ 5 все ≥ 5

3. Толстое разбиение (Проблема флага Нидерландов)

Проблема: что если много одинаковых элементов?

Решение: делим на ТРИ части!

Алгоритм:

1. Выбираем pivot
2. l = начало, r = конец, mid = начало
3. Пока mid \leq r:
 - Если $\text{arr}[\text{mid}] < \text{pivot}$: swap(arr[l], arr[mid]), l++, mid++
 - Если $\text{arr}[\text{mid}] = \text{pivot}$: mid++
 - Если $\text{arr}[\text{mid}] > \text{pivot}$: swap(arr[mid], arr[r]), r--

Пример: [3, 5, 2, 1, 5, 4, 5, 2] с pivot = 5

Итог: [3, 2, 1, 4, 2, 5, 5, 5]
 \uparrow \uparrow \uparrow
 $< \text{pivot}$ $> \text{pivot}$ = pivot

Время работы

Худший случай:

$O(n^2)$ - когда pivot всегда минимальный или максимальный

Лучший случай:

$O(n \log n)$ - когда pivot всегда медиана

Средний случай (при случайном выборе pivot):

$O(n \log n)$ - и это то, ради чего всё затевалось!

Почему в среднем $O(n \log n)$?

Математика:

$$T(n) = \frac{1}{n} \sum_{k=0}^{n-1} [T(k) + T(n-1-k)] + O(n)$$

$$T(n) = O(n \log n)$$

Интуитивно:

- В среднем pivot делит массив пополам
- Глубина рекурсии: $O(\log n)$
- На каждом уровне: $O(n)$ работы
- Итого: $O(n \log n)$

Выбор pivot

Способ	Плюсы	Минусы
Первый элемент	Просто	$O(n^2)$ на отсортированных
Последний элемент	Просто	$O(n^2)$ на отсортированных
Средний элемент	Лучше на отсортированных	Все еще можно подобрать плохой случай
Случайный элемент	$O(n \log n)$ в среднем	Нужен ГСЧ
Медиана медиан	Гарантия $O(n \log n)$	Сложная реализация

Практические советы

- Для общего случая: случайный pivot
- Для избежания $O(n^2)$: медиана из трех
- При многочисленных дубликатах: толстое разбиение
- Для маленьких массивов: переключаться на сортировку вставками

Итог: Быстрая сортировка - это как демократия: в худшем случае бардак, но в среднем работает лучше всех! Главное - хорошо выбирать "лидера"(pivot).