k-ая порядковая статистика. Поиск k-ой порядковой статистики. Среднее время работы алгоритма при случайном выборе pivot

ceagest

1 Стратегии выбора элементов

Рассмотрим стратегию, при котором выбирается центральный элемент

- Заметим, что если при каждом partition в качестве опорного брать самый маленький элемент, то у нас подзадачи будут иметь размеры 0 и $n-1$.
- Это ужасно плохо, так как при такой работе алгоритм скатится до $\mathcal{O}(n^2)$.
- Таким образом, от того, как мы выбираем опорный элемент зависит время работы нашего алгоритма.
- Неплохой стратегией является выбор случайного элемента.
- Также можно брать медиану трех случайных элементов (или даже 5-ти или 7-ми).
- Стратегия, при которой выбирается случайный элемент сводит на нет возможность подобрать такой массив, на котором наш алгоритм будет работать долго.
- Худшее, что может произойти: нам не повезет n раз подряд, вероятность чего мизерная.
- Проведем анализ работы быстрой сортировки, когда в качестве опорного выбирается случайный элемент.
- Анализ будем проводить из предположения, что все элементы различны.

Лемма 1.1. *Пусть X — число сравнений, выполняемых за время работы сортировки над n-элементным массивом. Тогда время работы быстрой сортировки составляет $\mathcal{O}(n + X)$.*

Доказательство. Быстрая сортировка делает не более n вызовов функции partition, который в свою очередь совершает некоторое количество итераций цикла, в каждой из которых происходит сравнение элементов с pivot. □

Лемма 1.2. $\mathbb{E}[X] = \mathcal{O}(n \log n)$, *где X — число сравнений, выполняемых за время работы сортировки над n-элементным массивом.*

Доказательство. Интуитивное объяснение, НЕ ЯВЛЯЕТСЯ МАТЕМАТИЧЕСКИ ТОЧНЫМ ДОКАЗАТЕЛЬСТВОМ: сделаем вид что массив всегда разбивается «примерно» пополам. Тогда рекуррентное соотношение совпадает с рекуррентным соотношением алгоритма MergeSort:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

Из мастер теоремы о рекурсии получаем, что $T(n) = \mathcal{O}(n \log n)$. □

2 Порядковые статистики

Определение 2.1. *k-порядковая статистика – это k-ый по величине элемент.*

2.1 Алгоритм поиска

Пусть мы выбрали pivot с помощью алгоритма $\mathcal{A}$ и выполним Partition. Пусть исходный pivot оказался на позиции k', тогда получается, что он больше ровно k' элементов слева и меньше $n - k'$ элементов справа. А значит, по определению, он является k' порядковой статистикой.

Рассмотрим случаи:

1. $k = k'$. Нашли искомую порядковую статистику.
2. $k < k'$. Значит порядковая статистика находится левее, чем pivot. Тогда запускаем рекурсивно действия выше, только уже от левого массива, и ищем k-ую порядковую.
3. $k > k'$. Значит порядковая статистика находится правее, чем pivot. Тогда запускаем рекурсивно действия выше, только уже от правого массива, и ищем, учитывая «смещенную нумерацию», $(k - k' - 1)$-ую порядковую статистику.

Лемма 2.1. *Алгоритм в среднем работает за* $\mathcal{O}(n)$.