Студент: Александр Никулин
Дата: 28 сентября 2020 г.

Задача 1. *Оцените время работы детерминированного алгоритма поиска порядковой статистики, если вместо пятерок разбивать элементы на*

(a) *семерки.*
(b) *тройки.*

Решение. Мы знаем, что первый, второй и четверный шаг алгоритма при любом разбиение отработают за время сравниемое с $\mathcal{O}(n)$, т.к сортировка занимает константное время (разбиваем на фиксированное количество элементов), выполнение partition также линейно. Третий шаг, нахождение медианы от медиан займет $T(\frac{n}{k})$, где k - размер группы, на которую мы разбиваем. Остается разобраться с последним шагом, где для анализа мы выбираем худший случай, то есть $T(n-c)$, где c - нижняя граница количества элементов $\geq$ или $\leq$ медианы от медиан.

(a) Аналогично док-ву с семинара (не знаю как рисовать кружочки в латехе), найдем, что при разбиении на группы размера 7, у нас получится не меньше $\frac{4n}{14} - 4$ элементов, которые $\leq$ медианы медиан, и столько же $\geq$. Константу можно не учитывать, либо сделав неравенство строгим, либо предположив, что n делится на 7. Значит, рекурсивный вызов в худшем случае будет работать за $T(\frac{10n}{14})$. Тогда:

$$T(n) = \mathcal{O}(n) + T(\frac{n}{7}) + T(\frac{5n}{7})$$

(b) При разбиении на группы по 3 у нас получится, что не меньше $\frac{2n}{6}$ элементов $\leq$ и $\geq$ медианы медиан. Тогда,

$$T(n) = \mathcal{O}(n) + T(\frac{n}{3}) + T(\frac{2n}{3})$$

Далее, для решения этих соотношений можно воспользоваться общей формулой, которую я нашел в учебнике Кормена, ибо не смог решить (b) сам, а как додуматься до предположения, что это все же $\mathcal{O}(n \log n)$ тоже непонятно. Формула выглядит так (одно из обобщений мастер теоремы):

$$T(n) \leq \sum_{i=1}^{k} T(a_i n) + O(n), \text{ где } a_i > 0 \text{ for } i = 1, \ldots, k \text{ и } \sum_{i=1}^{k} a_i \leq 1$$

$$T(n) = \begin{cases} \mathcal{O}(n) & \text{if } \sum_{i=1}^{k} a_i < 1 \\ \mathcal{O}(n \log n) & \text{if } \sum_{i=1}^{k} a_i = 1 \end{cases}$$

Убедимся, что $\frac{1}{7} + \frac{5}{7} < 1$, а $\frac{1}{3} + \frac{2}{3} = 1$, а значит при разбиении на группы по 7 мы все еще получаем $\mathcal{O}(n)$ в худшем случае, а при разбиении по 3 уже получаем $\mathcal{O}(n \log n)$.

Задача 2.

Решение.

Задача 3.

Решение.

Задача 4. *Докажите, что для поиска максимума в массиве различных чисел потребуется как минимум $n - 1$ сравнение.*

Решение. Максимальным может быть только один элемент из n, а остальные $n - 1$ элементов максимальными не являются. Каждое сравнение увеличивает количество не максимальных элементов на один (один из элементов оказался меньше, а значит не может быть максимальным). Таким образом, нам понадобится как минимум $n - 1$ сравнение, чтобы получить максимальный элемент (оставшийся один).

Задача 5. *Дана последовательность из n чисел, нужно за один проход и $\mathcal{O}(n)$ времени в среднем найти в ней k минимумов, используя $\mathcal{O}(k)$ дополнительной памяти.*

Решение. Без ограничения на память задача решается просто (и в "среднем" намекает) с помощью процедуры partition. Найдем k-ый элемент с помощью QuickSelect, после чего запустим partition для этого элемента - получим все элементы меньше k-ого, но не упорядоченные.

Чтобы уложиться в ограничение по памяти, заведем массив размера $2k$ и найдем там k-ю порядковую статистику (за $\mathcal{O}(k)$ в среднем). Слева от нее все элементы меньше (k - 1 штука), справа больше. Теперь заполним вторую половину массива k элементами из изначального и повторим первые шаги, пока изначальный массив не пуст. На каждом шаге мы отбрасываем элементы, которые не могут быть первыми k минимумами т.к. больше k-той статистики. Всего таких шагов будет $\frac{n-2k}{k}$, каждый из которых занимает $\mathcal{O}(k)$. Итоговое время можно оценить как $\mathcal{O}(n-2k) = \mathcal{O}(n)$ в среднем.

Задача 6. *Дан массив из $2n$ различных чисел. Найдите минимальное и максимальное за $3n-2$ сравнения и докажите, что это точная нижняя оценка, то есть меньшего количества сравнений может не хватить.*

Решение. Можно отдельно найти миниальный и максимальный элементы, получив $(2n-1)+(2n-1) = 4n-2$ сравнений. Можно быстрее, если обрабатывать элементы по парам. Зафиксируем максимальный и минимальный элемент. Тогда, приняв очередную пару, вначале сравним внутри пары, а потом больший элемент будем сравнивать с максимальным, а меньший с минимальным. Каждая пара, таким образом, добавляет три сравнения в общее количество. Всего пар n. Общее количество сравнений: $\frac{3(2n-2)}{2} + 1 = 3n-2$. Одно сравнение на выбор минимума и максимума и еще $\frac{2n-2}{2}$ пар осталось. Осталось доказать, что лучше нельзя.

Всего кандидатов на минимум и максимум $2n$, а останется только один. Значит, мы должны проверить $2(2n-1) = 4n-2$ кандидатов, которые не окажутся минимум & максимумом. Рассмотрим возможные случаи, возникающие при сравнении двух чисел. Если оба числа являются кандидатами на минимум и максимум, то после сравнения мы уменьшим количество кандидатов на два. Если только один является кандидатом (на минимум или на максимум), то после сравнения мы уменьшим количество кандидатов на один. Любые другие сравнения ничего не меняют (например те кандидаты, что уже участвовали в сравнении). Сравнений убирающих по два может быть максимум $\lfloor \frac{2n}{2} \rfloor$. Тогда всего:

$$((4n-2) - 2\lfloor \frac{2n}{2} \rfloor) - \lfloor \frac{2n}{2} \rfloor) = 4n - 2 - 2n + n = 3n - 2$$