Четвёртая домашняя работа по алгоритмам

Ситдиков Руслан

1 Задача

1.Изначально $y = 1$. Двигаясь поэлементно по SX-записи числа 5, получим: $y = 3 \to y = 9 \to y = 81 \to y = 243$.
2.Функция возводит x в степень целого числа m.
3.На каждом шаге мы работаем с целым числом x. Мы, либо умножаем его на начальное x, либо на само себя(уже измененное). Мы последовательно уменьшаем число m, проделывая с ним всего 2 операции, либо отнимаем единицу, либо уменьшаем его в 2 раза,обе операции только уменьшают наше m. Алгоритм прекратится, так как двоичная запись числа с каждым шагом всегда уменьшается. Следовательно, наш алгоритм корректен.
4. Алгоритм аналогичен работе с бинарным возведением в степень - m делится пополам в случае четности и уменьшается на 1 в случае нечетности. Следовательно, алгоритм работает за $O(\log n)$.

2 Задача

Алгоритм

Для начала отсортируем массив отрезков по возрастанию левой границы. Таким образом, мы получим систему вложенных отрезков. Тогда, самый широкий отрезок будет первым, а самый узкий - последним в массиве отрезков. Обозначим левую и правую границы самого широкого отрезка l и r соответственно. Выберем точку $\left\lfloor \frac{l+r}{2} \right\rfloor$. Пройдясь по массиву отрезков, посчитаем, какое количество k из них содержат выбранную точку, и, заодно, запомним, в какой из двух половин лежит середина самого узкого отрезка q.

Если выбранная точка не совпадает с q, сравним k с $\frac{2n}{3}$. Если k меньше, выберем половину, содержащую q, переприсвоим l и r её границы и рекурсивно вызовем функцию от этой половины. Если k больше, выберем половину, не содержащую q, переприсвоим l и r её границы и рекурсивно вызовем функцию от этой половины. Если $k = \frac{2n}{3}$, то выводим точку и таким же образом рекурсивно вызовем обе половины. Если выбранная точка совпадает с q, рекурсивно вызовем любую из половин(без ограничения общности - правую). Рекурсия закончится, когда l и r совпадут.

Корректность

Так как наш отрезок конечен и в конце концов мы придем к тому, что, вследствии округления, наши границы отрезка совпадут. Получаем, что между ними не может находиться еще каких-то ненайденных точек. А так как в процессе выполнения алгоритма будут рассмотрены все части массива, подходящие под наше условие, то мы не "упустим"никакую точку "по дороге". Следовательно, наш алгоритм корректен.

Асимптотика

Сортировка исходного массива происходит за $O(n \log n)$, рекурсия по основной теореме о рекуррентных соотношениях работает за $O(n \log n)$. Общая асимптотика — $O(n \log n)$

3 Задача

Проделав всё то же самое как и в Кормене 9.3 мы получим, что количество элементов, значение которых превышает значение медианы медиан, удовлетворяет следующему равенству:

$$4 \cdot \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{7} \right\rceil \right\rceil - 3 \right) \geq \frac{2 \cdot n}{7} - 12$$

Аналогично, имееется не менее $\frac{2 \cdot n}{7} - 12$ элементов, значение которых меньше медианы медиан. Получаем, что процедура рекурсивно вызывается на седьмом шаге не менее чем для $\frac{5 \cdot n}{7} + 12$ элементов. Теперь выведем рекуррентное соотношение

$$T(n) \leq T\left(\frac{n}{7}\right) + T\left(\frac{5 \cdot n}{7}\right) + C \cdot n$$

Допустим, что для некоторой константы c выполняется неравенство $T(n) \leq c \cdot n$

$$T(n) \leq c \cdot \frac{n}{7} + c \cdot \frac{5 \cdot n}{7} + C \cdot n = n \cdot \left(\frac{6 \cdot c}{7} + C\right)$$

$$T(n) \leq c \cdot n \Leftrightarrow C \leq \frac{c}{7}$$

Отсюда следует, что $T(n) = O(n)$

4 Задача

Первый алгоритм(не знаю как доказать корректность, поэтому написал ещё второй алгоритм) Пробежимся циклом по нашему массиву от $i = 0$ до $i = n - 1$,если на нашем пути будем встречать случай когда $a[i] > a[i + 1]$ будем менять наши элементы местами. На выходе выведем

наш отсортированный массив.
Второй алгоритм
Заведём переменную $l = 0$. Пробежимся циклом по нашему массиву, и каждый раз встречая 0, будем увеличивать наше l на один. Далее создадим новый массив с l нулями и $n - l$ единицами.
Корректность второго алгоритма
Наш алгоритм корректен, так как при наших проходах он найдёт количество нулей в нашем массиве. Если же их не будет, то $l = 0$ и на выходе мы получим массив полностью состоящий из единиц. Если же весь массив состоит из нулей, то $l = n$, на выходе мы получим l нулей Следовательно, наш алгоритм корректен.

5 Задача

Приведем наше уравнение к диофантовому виду:

$$a \cdot x + M \cdot y = b$$

Алгоритм: Внести вычисление этих коэффициентов в алгоритм Евклида несложно, достаточно вывести формулы, по которым они меняются при переходе от пары (a, b) к паре $(b\%a, a)$
Итак, пусть мы нашли решение (x_1, y_1) задачи для новой пары $(b\%a, a)$:

$$(b\%a) \cdot x_1 + a \cdot y_1 = g.$$

Для этого преобразуем величину b% a:

$$b\%a = b - \lfloor b/a \rfloor \cdot a;$$

Подставим это в приведённое выше выражение с x_1 и y_1 и получим:

$$g = (b\%a) \cdot x_1 + a \cdot y_1 = (b - \lfloor b/a \rfloor \cdot a) \cdot x_1 + a \cdot y_1,$$

и, выполняя перегруппировку слагаемых, получаем:

$$g = b \cdot x_1 + a \cdot (y_1 - \lfloor b/a \rfloor \cdot x_1).$$

Сравнивая это с исходным выражением над неизвестными x и y, получаем требуемые выражения:

$$x = y_1 - \lfloor b/a \rfloor \cdot x_1, y = x_1.$$

Корректность Сначала заметим, что при каждой итерации алгоритма Евклида его второй аргумент строго убывает, следовательно, поскольку он неотрицательный, то алгоритм Евклида всегда завершается. Для доказательства корректности нам необходимо показать, что $gcd(a, b) = gcd(b, a \ mod b) \ \forall a \geq 0, b > 0$ Покажем, что величина, стоящая в левой части равенства, делится на настоящую в правой, а стоящая в правой — делится на стоящую в левой. Очевидно, это будет означать, что левая и правая части совпадают, что и докажет корректность алгоритма Евклида. Обозначим $d = \gcd(a, b)$. Тогда, по определению,$d|a$ и $d|b$. Далее, разложим остаток от деления a на b через их частное:

$$a \ mod b = a - b\lfloor a/b \rfloor$$

Но тогда следует:

$$d|(a \ mod b)$$

Итак, вспоминая утверждение $d|b$, получаем систему:

$$\begin{cases} d|b \\ d|(a \ mod b) \end{cases}$$

Воспользуемся теперь следующим простым фактом: если для каких-то трёх чисел p, q, r выполнено: $p|q$ и $p|r$, то выполняется и: $p \mid \gcd(q, r)$. В нашей ситуации получаем:

$$d|gcd(b, a \ mod b)$$

Или, подставляя вместо d его определение как $\gcd(a, b)$, получаем:

$$gcd(a, b)|gcd(b, a \ mod b)$$

Итак, мы провели половину доказательства: показали, что левая часть делит правую. Вторая половина доказательства производится аналогично.

Сложность Время работы алгоритма оценивается теоремой Ламе, которая устанавливает удивительную связь алгоритма Евклида и последовательности Фибоначчи:

Если $a > b \geq 1$ и $b < F_n$ для некоторого n, то алгоритм Евклида выполнит не более $n - 2$ рекурсивных вызовов.

Более того, можно показать, что верхняя граница этой теоремы — оптимальная. При $a = F_n, \ b = F_{n-1}$ будет выполнено именно $n - 2$ рекурсивных вызова. Иными словами, последовательные числа Фибоначчи — наихудшие входные данные для алгоритма Евклида.

Учитывая, что числа Фибоначчи растут экспоненциально (как константа в степени n), получаем, что алгоритм Евклида выполняется за $O(\log \min(a, b))$ операций умножения. А так как наши входные данные записаны в двоичной системе, то наш алгоритм будет работать за линейное время.

6 Задача

1.В самом несбалансированном варианте каждое разделение даёт два подмассива размерами 1 и $n-1$, то есть при каждом рекурсивном вызове больший массив будет на 1 короче, чем в предыдущий раз. Такое может произойти, если в качестве опорного на каждом этапе будет выбран элемент либо наименьший, либо наибольший из всех обрабатываемых. При простейшем выборе опорного элемента — первого или последнего в массиве, — такой эффект даст уже отсортированный (в прямом или обратном порядке) массив, для среднего или любого другого фиксированного элемента «массив худшего случая» также может быть специально подобран. В этом случае потребуется $\sum_{i=0}^{n}(n-i) = O(n^2)$ операций, то есть сортировка будет выполняться за квадратичное время.
2. Вместо того, чтобы после разделения массива вызывать рекурсивно процедуру разделения для обоих найденных подмассивов, рекурсивный вызов делается только для меньшего подмассива, а больший обрабатывается в цикле в пределах этого же вызова процедуры. Глубина рекурсии ни при каких обстоятельствах не превысит $\log_2 n$, а в худшем случае вырожденного разделения она вообще будет не более 2 — вся обработка пройдёт в цикле первого уровня рекурсии. Решение задачи нашёл в википедии, зашарил ;)