Домашнее задание №3

Томинин Ярослав, 778 группа

3 марта 2018 г.

1.
Поймем, что на доске останется НОД этих чисел. Докажем сначала, что если все числа делятся на какое то число n, то разность их тоже будет делиться на n и сколько мы их не будем вычитать друг из друга все числа будут кратны n. Так же поймем, что наши числа, если они не делились на k, не будут делиться на k. Если мы представим все числа в кольце k, то мы поймем, что за одну операцию мы можем изменять остаток при делении только у одного числа. Если же все числа стали кратны k, то это означает, что за шаг до этого только одно число было не кратно k. Теперь поймем, что мы не можем вычесть из числа, не кратного k число кратное k и получить число, кратное k. Противоречие. Так же мы будем продолжать операции до тех пор, пока все числа не станут равны. А из доказанного выше это и означает, что останется их НОД.
2.
Поймем, что ранее мы уже реализовывали алгоритм Евклида за $O(n^2)$.(Так как у нас получается сумма i^2 , где i изменяется от 1 до n) Поэтому мы просто можем произведение наших чисел разделить на их НОД. Так как все арифметческие операции линейны, то наша асимтотика не изменится.
Корректность алгоритма. После перемножения чисел мы получаем число, в котором общие делители наших чисел попали два раза, а остальные по одному. Следовательно, если мы наше число разделим на НОД, то мы действительно получим НОК.
Сложность по времени. $\Theta = O(n^2)$
3.
Посчитаем сумму чисел и возведем ее в квадрат. После этого вычтем квадраты наших чисел и мы получим в точности то, что и хотели.

$$(a_1 + \cdots + a_n)^2 - \sum_{i=1}^{n} a_i^2 = \sum_{i \neq j} a_i \times a_j$$

Корректость алгоритма. В нашем алгоритме мы пользуемся формулой раскрытия скобок и сокращаем элементы в степени 2.
Сложность по времени.Операции суммирований займут O(n), а умножение займет O(1). После этого мы возведем наши числа во вторую степень за O(n), вычтем их n раз за O(n). Следовательно алгоритм работает за $\Theta = O(n)$
4.
a)

Из основной теореиы о рекуррентных оценках следует, что так как

$$f = n^{\log_b a} = n^2$$

То

$$T(n) = \Theta(n^2 \lg n)$$

b)
Поймем, что

$$3\frac{n^2}{9} \leq n^2$$

$$n^2 = n^{\log_b a + \epsilon}, \epsilon = 1$$

Следовательно

$$T(n) = \Theta(n^2)$$

c)
Поймем, что

$$f(n) = \frac{n}{\log n} = O(n^{\log_2 4 - \epsilon}), \epsilon = 1$$

Отсюда следует, что

$$T(n) = \Theta(n^2)$$

5.
Запишем рекурсивную формулу

$$T(n) = nT(\lceil \frac{n}{2} \rceil) + \alpha n$$

То есть наша сложность будет равна

$$\sum_{i=1}^{\log n} \frac{n^{i+1}}{2^i}$$

Поймем, что каждый i+1 элемент является O для i, а именно

$$\frac{n^i}{2^{i-1}} = O(\frac{n^{i+1}}{2^i})$$

Следовательно наша сложность будет равна сложности последнего элемента

$$T(n) = \Theta(\frac{n^{\log n}}{\log n})$$

6.
a) $T(n) = T(\lfloor \alpha n \rfloor) + T(\lfloor (1-\alpha)n \rfloor) + \Theta(n) \quad (0 < \alpha < 1)$

Выберем из α и $1-\alpha$ большее. В нашем случае на нарушая общности мы будем считать, что $\alpha < 1-\alpha$. Тогда будем считать что путь прекратится, когда наше текущее n станет равно 1. Тогда в нашей ситуации самый короткий путь будет равен

$$n\alpha^k$$

$$k = \log_\alpha \frac{1}{n}$$

Так как на каждом уровне сумма элементов равна n, то общая сложность $T(n) = \Omega(n \log_\alpha \frac{1}{n})$ Если же мы возьмем самый большой путь, то его длина будет равна

$$n = (1-\alpha)^m$$

$$k = \log_{1-\alpha} \frac{1}{m}$$

Следовательно $T(n) = O(n \log_{1-\alpha} \frac{1}{n})$
Тогда понятно, что $T(n) = \Theta(n \log_\alpha \frac{1}{n})$
b) $T(n) = T(\lfloor n/2 \rfloor) + 2 \cdot T(\lfloor n/4 \rfloor) + \Theta(n)$;
Аналогично пункту а возьмем самый маленький и самый длинный пути

$$k = \log_4 n$$

$$T(n) = \Omega(n \log_4 n)$$

Самый длинный путь

$$m = \log_2 n$$

$$T(n) = \Omega(n \log_2 n)$$

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

c)

Найдем длину нашего пути

$$k = \log_3 n$$

Тогда выведем формулу сложности

$$\sum_{i=1}^{\log_3 n} \frac{n^3}{(\log^2 \frac{n}{3^i})}$$

Поймем, что первые $\frac{\log n}{2}$ членов будут больше, чем

7.
a)
Для начала поймем, что $i!(i!)^{-1} \equiv 1(modp) \implies ((i-1)!)^{-1}(i-1)!i(i!)^{-1} \equiv ((i-1)!)^{-1}(modp) \implies i(i!)^{-1} \equiv ((i-1)!)^{-1}(modp)$ Теперь найдем n! mod p за O(n) операций, а потом вычислим его обратный элемент с помощью алгоритма из прошлого задания за O(n). А потом с помощью полученной ормулы найдем оставшиеся обратные элементы. Поэтому наш алгоритм займет O(n) операций.
b)
Рассмотрим следующие преобразования по mod p $0 \equiv \lfloor \frac{p}{i} \rfloor i + (p\%i) \implies 0 \equiv \lfloor \frac{p}{i} \rfloor + (p\%i)i^{-1} \implies 0 \equiv \lfloor \frac{p}{i} \rfloor (p\%i)^{-1} + i^{-1} \implies 0 \equiv \lfloor \frac{p}{i} \rfloor (p\%i)^{-1} + i^{-1} \implies \lfloor \frac{p}{i} \rfloor (p\%i)^{-1} \equiv -i^{-1}$ Теперь будем вычислять inv[1], inv[2] и т.д. Теперь поймем что на каждый из них будет затрачено O(1) операций, потому что каждый следующий моно выразить через предыдущие за O(1), а inv[1]=1.