Третье задание по алгоритмам

Руслан Ситдиков

26 февраля 2018 г.

1 Задача

Это число будет равно наибольшему общему делителю всех этих чисел. Так как когда мы производим операцию разности, НОД этих чисел никак не изменяется ($GCD(a, b) = GCD(a - b, b)$), следовательно в конце, мы получаем их НОД.

2 Задача

Алгоритм:

Находим алгоритмом Евклида НОД двух чисел. Далее пользуемся тем фактом, что $GCD(a, b) \cdot LCM(a, b) = a \cdot b$ и отсюда следует, что $LCM = \frac{a \cdot b}{GCD}$.

Корректность:

Пусть S — какое-нибудь кратное чисел a и b. Т.е., S делится на a,и по определению делимости существует некоторое целое число k такое, что справедливо равенство $S = a \cdot k$. Но S делится и на b, тогда $a \cdot k$ делится на b. Обозначим НОД(a,b) как d. Тогда можно записать равенства $a = a_1 \cdot d$ и $b = b_1 \cdot d$, причем $a_1 = \frac{a}{d}$ и $b_1 = \frac{b}{d}$ будут взаимно простыми числами. Следовательно, полученное в предыдущем абзаце условие, что $a \cdot k$ делится на b, можно переформулировать так:$a_1 \cdot d \cdot k$ делится на $b_1 \cdot d$, а это в силу свойств делимости эквивалентно условию, что $a_1 \cdot k$ делится на b_1. В этом случае по свойству взаимно простых чисел,т.к. $a_1 \cdot k$ делится на b_1, и a_1 не делится на b_1 (a_1 и b_1 — взаимно простые числа), то на b_1 должно делиться k. Тогда должно существовать некоторое целое число t, для которого $k = b_1 \cdot t$, а т.к. $b_1 = \frac{b}{d}$, то $k = \frac{b}{d \cdot t}$. Подставив в равенство $S = a \cdot k$ вместо k его выражение вида $\frac{b}{d \cdot t}$, приходим к равенству $LCM = \frac{a \cdot b}{GCD}$. Так мы получили равенство $S = \frac{a \cdot b}{d \cdot t}$, которое дает вид всех общих кратных чисел a и b. Из того, что a и b числа положительные по условию следует, что при $t = 1$ мы получим их наименьшее положительное общее кратное, которе равно $\frac{a \cdot b}{d}$. Этим доказано, что $LCM = \frac{a \cdot b}{GCD}$.

Оценка:

$LCM(a, b) \cdot GCD(a, b) = a \cdot b$. Поэтому верхняя оценка будет $O(n^3)$. Чтобы найти НОД нам потребуется $O(n^3)$ операций,т.к. по оба числа уменьшаются в 2 раза за 2 итерации, то нам потребуется не более 2n входов, где n — длина побитовой записи наименьшего числа, а т.к. операции у нас атомарные, то деление потребует $O(n^2)$ операций. В итоге получим $O(n^3)$. Умножение двух чисел потребует $O(n^2)$ операций, где n — длина побитовой записи наименьшего числа. Поэтому верхняя оценка работы алгоритма $O(n^3)$.

3 Задача

Пробежимся циклом по нашему массиву и найдём сумму всех чисел нашего массива, заодно возведём её в квадрат. Пробежимся ещё раз по циклу, и будем возводить наши числа в квадрат, и каждый раз отнимать от нашей суммы квадраты наших чисел. На выходе выведем нашу сумму делённую на два.

4 Задача

а) Так как $f(n) = n^2 = \Theta(n^{log_6 36})$, это подходит под второй случай. Следовательно $T(n) = n^2 \log n$
б) $f(n) = n^2, a = 3, b = 3$. Так как $f(n) = \Omega(n^{log_3 3+\varepsilon})$, для $\varepsilon = 1/2$. Выполняется и второе условие $af(n/b) \leq cf(n)$, для $c = 1/2$. Следовательно $T(n) = \Theta(n^2)$
в) $T(n) = 4T(\frac{n}{2}) + \frac{n}{\log n}$; $a = 4$, $b = 2$, $f(n) = \frac{n}{\log n} = O(n^{\log_b a-\varepsilon}) \Longrightarrow T(n) = \Omega(n^2)$ по первому пункту теоремы.

5 Задача

$T(n) = nT(n) + O(n) = \frac{n^k}{2^{0.5k(k-1)}}T(\lceil \frac{n}{2^k} \rceil) + \sum\limits_{i=0}^{k} O(\frac{n}{2^{0.5i(i-1)}}) = Cn^{\frac{\log n-1}{2}} + \sum\limits_{i=0}^{\log n} O(\frac{n^i}{2^{0.5i(i-1)}})$. И т.к. $\sum\limits_{i=0}^{\log n} O(\frac{n^i}{2^{0.5i(i-1)}}) \leq O(\log n \cdot n^{\frac{\log n+1}{2}})$, то $T(n) = \Omega(\frac{\log n+1}{2})$ и $T(n) = O(\log n \cdot n^{\frac{\log n+1}{2}})$.

6 Задача

а) Предположим, что $T(n) = cn \log n$ и без ограничения общности $\alpha < \frac{1}{2}$. Тогда $T(n) = \alpha \cdot cn \log \alpha n + (1-\alpha) \cdot cn \log(1-\alpha)n + \Theta(n) = cn \log n + C + \Theta(n) \Longrightarrow T(n) = \Theta(n \log n)$. Иными словами мы можем оценить снизу наш алгоритм короткой веткой дерева, а сверху длинной и в обоих случаях будет $n \log n$, потому что на каждом уровне у нас cn операций и всего количество уровней оценивается логарифмом.
б) Предположим, что $T(n) = \Theta(n \log n)$. Тогда $T(n) = \frac{cn}{2}\log(\frac{n}{2}) + 2\frac{cn}{4}\log(\frac{n}{4}) + \Theta(n) = cn \log n + C + \Theta(n) = \Theta(n \log n)$. Иными словами мы можем оценить снизу наш алгоритм короткой веткой дерева, а сверху длинной и в обоих случаях будет $n \log n$, потому что на каждом уровне у нас cn

операций и всего количество уровней оценивается логарифмом.

в) $T(n) = 27T(\frac{n}{4}) + \frac{n^3}{\log^2 n} = 3^{3k}T(\frac{n}{3^k}) + \sum\limits_{i=0}^{k} \frac{n^3}{\log^2(\frac{n}{3^i})} = Cn^3 + n^3 \cdot \sum\limits_{i=0}^{\log_3 n} \frac{n^3}{\log^2(\frac{1}{3^i})}$

Т.к. $\sum\limits_{i=0}^{\log_3 n} \frac{1}{\log^2(\frac{n}{3^i})} \leq \sum\limits_{i=0}^{\log_3 n} \frac{1}{\log^2 n} = \frac{1}{\log n} \Longrightarrow T(n) = \Theta(n^3)$.

7 Задача

а)$invfac[i] = (i!)^{-1} (\mod p)$ $p \mod i = p - [\frac{p}{i}] \cdot i$. Возьмем $\mod p$ от двух частей и домножим на обратный остаток к $p \longrightarrow 1 = -[\frac{p}{i}] \cdot p^{-1} (\mod i) \cdot i$ по модулю р. Умножая на обратный i^{-1} получим $i^{-1} = -[\frac{p}{i}] \cdot p^{-1} (\mod i)$ по модулю р. Алгоритм работает за $O(n)$ арифметический операций,т.к. произведение в факториал по модулю $\Theta(n)$, работа цикла $\Theta(n)$ с входом $O(1)$ арифметических операций.