Основные алгоритмы 3

Ковалев Алексей

1. Основываясь на алгоритме составим реккурентное соотношение для f:

$$f(n) = 3f\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + c,\ c = 2020$$

При малых значениях аргумента ($n \leqslant 2020$)

$$f(n) = O(1)$$

Воспользуемся мастер-теоремой:

$$f(n) = 3f\left(\frac{n}{4}\right) + c$$

$$a = 3,\ b = 4,\ \log_b a = \log_4 3$$

$$\exists \varepsilon = 0.1 \colon c = O\left(n^{\log_b a - \varepsilon}\right) = O\left(n^{\log_4 3 - 0.1}\right)$$

$$f(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_4 3}\right)$$

Ответ: $f(n) = \Theta\left(n^{\log_4 3}\right)$.

2. Пусть исходный массив – $n_1, n_2, \ldots, n_k$, полученный в конце массив – $m, m, \ldots, m$, $\gcd(n_1, n_2, \ldots, n_k) = n$.

1. Мы всегда вычитаем из большего числа меньшее, значит числа на доске будут оставаться положительными целыми. В какой-то момент все числа станут одинаковыми, так как мы не можем бесконечно вычитать положительные целые числа, получая положительное целое число, тогда процесс остановится. □

2. Когда процесс остановится на доске будут записаны числа, все равные $\gcd(n_1, n_2, \ldots, n_k)$.
   Предположим обратное и рассмотрим два варианта:

   - Числа в конце не делятся n:
     тогда при последнем вычитании получено $m \colon n \nmid m$, а значит среди чисел, разность которых брали, хотя бы одно не делилось на n. Продолжая это рассуждение по индукции приходим к выводу, что среди $n_1, n_2, \ldots, n_k$ было число, которое не делится на n, что невозможно. Значит все полученные числа делятся на n.
   - Числа в конце делятся на n, но не равны ему:
     тогда $m = qn,\ q \in \mathbb{N},\ q \neq 1$. Сумма любого набора этих чисел будет делиться на qn, но среди всевозможных массивов, элементы которых состоят из каких-то сумм чисел итогового массива и только их, есть исходный, значит все числа в нем делятся на qn, что невозможно, поскольку n – наибольший общий делитель.

   Значит все полученные в конце числа равны n.

3. Приведем алгоритм умножение некоторых чисел a и b, длина максимального из которых в двоичной записи равна n. Воспользуемся следующим соотношением:

$$(a + b)^2 = a^2 + 2ab + b^2$$

Выразив отсюда ab имеем:

$$ab = \frac{1}{2} \cdot \left((a + b)^2 - a^2 - b^2\right)$$

Тогда произведение ab ищется за линейное время, в предположении того, что возможно возвести любое число в квадрат за $O(n)$. На возведение каждого из чисел $a + b,\ a,\ b$ в квадрат потребуется линейное время. Также $O(n)$ операций потребуется на вычитание чисел и деление (для деления на 2 будем использовать битовый сдвиг). Итого два любых числа можно перемножить за $O(n)$. □

4. Будем вычислять НОК следующим способом: известно, что $\gcd(n,m)\cdot \operatorname{lcm}(n,m) = nm$. Тогда $\operatorname{lcm}(n,m) = \frac{nm}{\gcd(n,m)}$. Обозначим $\min(\log n, \log m) = a$. Так как в этой задаче мы используем модель вычислений с атомарными битовыми операциями, сложность умножения двух чисел $O\big(a^2\big)$. В такой модели сложность вычисление НОД при помощи алгоритма Евклида составляеят $O\big(a^3\big)$. Тогда время работы алгоритма нахождения НОК составляет $O\big(a^2+a^3\big) = O\big(a^3\big)$.

5. Воспользуемся идеей, уже использованной в задаче 3: рассмотрим выражение

$$(a_1 + a_2 + \ldots + a_n)^2 = \sum_{i=1}^{n} a_i^2 + 2\sum_{i\neq j} a_i a_j$$

Тогда искомая сумма

$$\sum_{i\neq j} a_i a_j = \frac{1}{2}\cdot\left((a_1 + a_2 + \ldots + a_n)^2 - a_1^2 - a_2^2 - \ldots - a_n^2\right)$$

Она может быть вычислена за линейное от n количество арифметических операций: $O(n)$ операций на вычисление суммы всех элементов массивы, $O(n)$ операций на вычисление квадратов всех чисел, $O(n)$ операций на вычитание и еще $O(1)$ арифеметических операций на деление и возведение суммы всех элементов в квадрат. Всего $O(n)$ арифметических операций.

6.

1.
$$T(n) = 36T\left(\frac{n}{6}\right) + n^2$$
$$a = 36,\ b = 6,\ \log_b a = \log_6 36 = 2$$
$$f(n) = n^2 = \Theta\big(n^{\log_b a}\big)$$

Пользуясь мастер-теоремой получаем:

$$T(n) = \Theta\big(n^{\log_b a}\log n\big) = \Theta\big(n^2 \log n\big)$$

Ответ: $T(n) = \Theta\big(n^2\log n\big)$.

2.
$$T(n) = 3T\left(\frac{n}{3}\right) + n^2$$
$$a = 3,\ b = 3, \log_b a = \log_3 3 = 1$$
$$f(n) = n^2$$
$$\exists\varepsilon = \frac{1}{2}:\ f(n) = n^2 = \Omega\big(n^{\log_b a+\varepsilon}\big) = \Omega\left(n^{1+\frac{1}{2}}\right) = \Omega\left(n^{\frac{3}{2}}\right)$$
$$\exists c = \frac{2}{3}:\ af\left(\frac{n}{b}\right) = 3f\left(\frac{n}{3}\right) = 3\cdot\frac{n^2}{9} = \frac{n^2}{3} \leqslant cf(n) = \frac{2}{3}f(n) = \frac{2}{3}n^2$$

Пользуясь мастер-теоремой получаем:

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

Ответ: $T(n) = \Theta\big(n^2\big)$.

3.
$$T(n) = 4T\left(\frac{n}{2}\right) + \frac{n}{\log n}$$
$$a = 4,\ b = 2,\ \log_b a = \log_2 4 = 2$$
$$f(n) = \frac{n}{\log n}$$
$$\exists\varepsilon = \frac{1}{2}: f(n) = \frac{n}{\log n} = O\big(n^{\log_b a-\varepsilon}\big) = O\left(n^{2-\frac{1}{2}}\right) = O\left(n^{\frac{3}{2}}\right)$$

Пользуясь мастер-теоремой получаем:

$$T(n) = \Theta\big(n^{\log_b a}\big) = \Theta\big(n^2\big)$$

Ответ: $T(n) = \Theta\big(n^2\big)$.

7. Будем сортировать массив слиянием, дополнив алгоритм следующим образом: при очередном слиянии двух массивов a и b будем счиать, сколько элементов из a мы уже прошли (`i`), и помнить, сколько инверсий уже найдено (`inv`). Тогда, если мы добавляем в объединение массивов элемент из a, увеличиваем `i` на 1, если же добавляем в объединение элеменет из b, увеличиваем `inv` на `i`.
Корректность: алгоритм корректен, так как он сортирует массив слиянием, но при перестановке элементов он учитывает, являлась ли данная пара инверсией, то есть непосредственно считает их количество.
Сложность: временная сложность алгоритма – $O(n \log n)$, как и у MergeSort, так как мы увеличили количество действий не более чем на константу. Также алгоритм требует $O(n)$ памяти, так как мы использовали только $O(1)$ дополнительной по отношению к MergeSort памяти.

8.

$$T_1(n) = aT_1\left(\frac{n}{b}\right) + f(n)$$

$$T_2(n) = aT_2\left(\frac{n}{b}\right) + g(n)$$

$$f(n) = \Theta(g(n))$$

Из дерева рекурсии для обеих функций получаем следующие соотношения:

$$T_1(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n^i}{b^i}\right) + \alpha n^{\log_b a}$$

$$T_2(n) = \sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right) + \beta n^{\log_b a}$$

для некоторых $\alpha,\ \beta$. В силу того что $f(n) = \Theta(g(n))$ имеем $\exists c_1,\ c_2,\ N \colon \forall n \geqslant N$

$$c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)$$

$$c_1 \sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right) \leqslant \sum_{i=0}^{\log_b n} a^i f\left(\frac{n^i}{b^i}\right) \leqslant c_2 \sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right)$$

$$\sum_{i=0}^{\log_b n} a^i f\left(\frac{n^i}{b^i}\right) = \Theta\left(\sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right)\right)$$

$$T_1(n) = \Theta\left(\sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right)\right) + \alpha n^{\log_b a}$$

Причем $\alpha n^{\log_b a} = \Theta\left(n^{\log_b a}\right),\ \beta n^{\log_b a} = \Theta\left(n^{\log_b a}\right)$. Тогда

$$T_1(n) = \Theta\left(\sum_{i=0}^{\log_b n} a^i g\left(\frac{n^i}{b^i}\right)\right) + \Theta\left(n^{\log_b a}\right) = \Theta(T_2(n))$$

$$T_1(n) = \Theta(T_2(n))$$

□

9.

1. $$T(n) = 3T\left(\left\lfloor\frac{n}{4}\right\rfloor\right) + T\left(\left\lceil\frac{n}{6}\right\rceil\right) + n$$

   Так как на каждом уровне дерево ветвится не более чем в 4 раза, а n делится как минимум на 4, количество листьев будет составлять $O\left(n^{\log_4 4}\right) = O(n)$.

   $$T(n) = \sum_{i=0}^{\log_4 n} \frac{3^i n}{4^i} + \sum_{i=0}^{\log_6 n} \frac{n}{6^i} + O(n)$$

$$T(n) = n \cdot \frac{1-(3/4)^{\log_4 n+1}}{1-3/4} + n \cdot \frac{1-(1/6)^{\log_6 n+1}}{1-1/6} + O(n)$$

$$T(n) = 4n - 3 \cdot \left(\frac{3}{4}\right)^{\log_4 n} + \frac{1}{5}n - \frac{6}{5} \cdot \left(\frac{1}{6}\right)^{\log_6 n} + O(n)$$

$$T(n) = 4n - \frac{3n^{\log_4 3}}{n} + \frac{6}{5}n - \frac{1}{5} \cdot \frac{1}{n} + O(n) = \Theta(n)$$

Ответ: $T(n) = \Theta(n)$.

2.

$$T(n) = T(\lfloor \alpha n \rfloor) + T(\lfloor (1-\alpha)n \rfloor) + \Theta(n)\,,\ 0 < \alpha < 1$$

Так как на каждом уровне дерево ветвится не более чем в 2 раза, а n умножается на α и $1-\alpha$, количество листьев будет составлять $O\big(n^{-\log_\beta 2}\big)$, где $\beta = \min(1-\alpha, \alpha) \leqslant \frac{1}{2}$. Тогда, начиная с некоторого номера, для некоторых $c_1,\ c_2$ выполняется

$$c_1 \left(\sum_{i=0}^{-\log_\alpha n} \alpha n + \sum_{i=0}^{-\log_{1-\alpha} n} (1-\alpha)n \right) + O\big(n^{-\log_\beta 2}\big) \leqslant T(n) \leqslant c_2 \left(\sum_{i=0}^{-\log_\alpha n} \alpha n + \sum_{i=0}^{-\log_{1-\alpha} n} (1-\alpha)n \right) + O\big(n^{-\log_\beta 2}\big)$$

$$-c_1' n(\log_\alpha n + \log_{1-\alpha} n) + O\big(n^{-\log_\beta 2}\big) \leqslant T(n) \leqslant -c_2' n(\log_\alpha n + \log_{1-\alpha} n) + O\big(n^{-\log_\beta 2}\big)$$

$$c_1' n \log n + O\big(n^{-\log_\beta 2}\big) \leqslant T(n) \leqslant c_2' n \log n + O\big(n^{-\log_\beta 2}\big)$$

Причем $-\log_\beta 2 \leqslant 1$, в силу $\beta = \min(1-\alpha, \alpha) \leqslant \frac{1}{2}$, а значит $n^{-\log_\beta 2} = O(n)$. Тогда

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

Ответ: $T(n) = \Theta(n \log n)$.

3.

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 2T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n)$$

Заметим, что на каждом слое дерева рекурсии для $T(n)$ сумма элементов в вершинах равна n. Глебина дерева не больше $\log_2 n$, так как $T(n) \geqslant T\big(\lfloor \frac{n}{2} \rfloor\big)$ и не меньше $\log_4 n$, так как $T(n) \geqslant T\big(\lfloor \frac{n}{4} \rfloor\big)$, то есть глубина дерева логарифмическая. Значит $\exists c_1, c_2, N : \forall n \geqslant N\ c_1 n \log_4 n \leqslant T(n) \leqslant c_2 n \log_2 n$. Отсюда $T(n) = \Theta(n \log n)$.
Ответ: $T(n) = \Theta(n \log n)$.

4.

$$T(n) = 27T\left(\frac{n}{3}\right) + \frac{n^3}{\log^2 n}$$

Для некоторого α выполняется

$$T(n) = \sum_{i=0}^{\log_3 n} \frac{27^i (n/3^i)^3}{\log^2(n/3^i)} + \alpha n^{\log_3 27}$$

$$T(n) = \sum_{i=0}^{\log_3 n} \frac{n^3}{\log^2(n/3^i)} + \alpha n^3$$

$$T(n) = n^3 \sum_{i=0}^{\log_3 n} \frac{1}{(\log n - i \log 3)^2} + \alpha n^3$$

$$n^3 \sum_{i=0}^{\log_3 n} \frac{1}{(\log n - i \log 3)^2} \leqslant n^3 \Rightarrow \alpha n^3 \leqslant T(n) \leqslant (1+\alpha) n^3$$

$$T(n) = \Theta\big(n^3\big)$$

Ответ: $T(n) = \Theta\big(n^3\big)$.

10. Для начала покажем, как за $O(\log p)$ вычислять $a^{-1} \pmod p$.
По малой теореме Ферма

$$a^{p-1} \equiv 1 \pmod p, \ \gcd(a,p) = 1$$
$$a^{p-2} \equiv a^{-1} \pmod p$$

То есть мы можем найти a^{-1}, посчитав a^{p-2}, для чего воспользуемся быстрым возведенеим в степень, сложность которого $O(\log p)$.

1. Сначала вычислим $(n!)^{-1}$. Для этого необходимо сначала найти $n! \pmod p$, что делается за $O(n)$, а затем воспользоваться приведенном выше алгоритмом. Значение $(n!)^{-1}$ сразу сохраним в `invfac[`n`]`. Далее будем считать `invfac[`i`]` для $i = n-1, \ldots, 1$ домножая `invfac[`i`]` на $i-1$. Здесь мы пользуемся тем, что
$$(i!)^{-1} \equiv 1 \cdot 2^{-1} \cdot \ldots \cdot i^{-1} \pmod p$$
$$((i-1)!)^{-1} \equiv 1 \cdot 2^{-1} \cdot \ldots \cdot (i-1)^{-1} \equiv (i!)^{-1} \cdot i \pmod p$$
Итак, сложность предложенного алгоритма составляет $O(n + \log p)$ – $O(n + \log p)$ на вычисление $(n!)^{-1}$ и затем $O(n)$ операций на вычисление всех предыдущих значений `invfac[`i`]`.

2. Сначала вычислим массив `invfac[`i`]` для всех i от 1 до n из предыдыдущего пункта задачи. Затем вычислим массив `fac[`i`]` для всех i от 0 до $n-1$, на что нам потребуется $O(n)$ операций (для вычисление `fac[`i`]` умножаем `fac[`$i-1$`]` на i, `fac[0]` $= 0$). Теперь для вычисления `inv[`i`]` нужно умножить `invfac[`i`]` на `fac[`$i-1$`]` для всех i от 1 до n. Это потребует еще $O(n)$ арифметических операций, значит всего для вычисления `inv[`i`]` необходимо $O(n + \log p)$ арифметических операций.
Также понятно, что алгориитм можно оптимизировать по памяти, не храня целиком какой-то из массивов `invfac[]` или `fac[]`, а храня лишь переменную, в которой на шаге i хранятся либо $(i!)^{-1}$, либо $(i-1)!$ соответственно.