

ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ АЛГОРИТМА

Подготовил: ученик 9 класса Хайновский И.Г.

- Вычислительная сложность алгоритма Евклида изучена полностью.[17] Эта сложность может быть описана произведением количества шагов деления, требуемых алгоритмом, на вычислительную сложность одного шага. Первый известный анализ алгоритма Евклида был предложен Рейнаудом в 1811.[18] Он показал, что число шагов алгоритма для пары чисел (u, v) ограничено v . Позже он улучшил оценку до $v/2 + 2$. Эмиль Леже, в 1837 году, изучил наихудший случай, когда для вычисления НОД подаются последовательные числа Фибоначчи.[19] Затем, в 1841 году, Пьер Жозеф Финк показал,[20] что количество шагов алгоритма не превышает $2 \log_2 v + 1$. Следовательно алгоритм работает за полиномиальное время от размера меньшего из пары чисел (u, v) . [19] Анализ Финка был уточнён Габриэлем Ламе в 1844 году.[21] Он показал, что количество шагов, необходимых для завершения алгоритма, не более чем в пять раз превышает n — количество цифр в десятичном представлении меньшего из пары чисел (u, v) . [22][23]
- Когда НОД вычисляется для чисел, которые вписываются в одно машинное слово, каждый шаг алгоритма занимает постоянное время. В данном случае анализ Ламе предполагает, что вычислительная сложность оценивается как $O(n)$. Однако в модели расчёта, подходящей для вычислений с числами больше одного машинного слова, оценка сложности вычисления одного остатка может быть $O(n^2)$. [24] В этом случае общее время для всех этапов алгоритма можно проанализировать с помощью телескопического ряда, показав что это также $O(n^2)$. Для ускорения алгоритма Евклида могут быть использованы современные алгоритмические методы, основанные на методе Шёнхаге — Штрассена для быстрого целочисленного умножения. Это приводит к квазиполиномиальному времени. [25]

КОЛИЧЕСТВО ШАГОВ

- Число шагов для вычисления НОД двух натуральных чисел a и b обозначим как $T(a, b)$. Если g — это наибольший общий делитель a и b , тогда $a = mg$ и $b = ng$ для двух взаимно простых чисел m и n . Тогда $T(a, b) = T(m, n)$, что можно заметить, если разделить уравнения, полученные при вычислении НОД для пары (a, b) , на g . [26] Используя тот же принцип, число шагов алгоритма остаётся неизменным, если a и b умножаются на общий множитель w , что эквивалентно равенству $T(a, b) = T(wa, wb)$. Следовательно, количество шагов T может сильно различаться между соседними парами чисел, такими как (a, b) и $(a, b+1)$, так как данная величина зависит от НОД.
- Рекурсивный характер алгоритма Евклида даёт следующее уравнение $T(a, b) = 1 + T(b, r_0) = 2 + T(r_0, r_1) = \dots = N + T(r_{N-2}, r_{N-1}) = N + 1$, где $T(x, 0) = 0$ по предположению. [17]

НАИХУДШИЙ СЛУЧАЙ

- Если для алгоритма Евклида требуются N шагов для пары натуральных чисел $a > b > 0$, наименьшие значения a и b , для которых это выполняется — числа Фибоначчи F_{N+2} и F_{N+1} соответственно.[27] Тогда, если алгоритм Евклида требует N шагов для пары чисел (a,b) , где $a > b$, выполняются следующие неравенства $a \geq F_{N+2}$ и $b \geq F_{N+1}$. Доказать это можно по математической индукции. Если $N = 1$, тогда a делится на b без остатка. Наименьшие натуральные числа, для которых это верно, равны $b = 1$ и $a = 2$, соответственно F_2 и F_3 . Предположим теперь, что результат выполняется для всех значений N до $M - 1$. Первый шаг алгоритма Евклида с M шагами $a = q_0b + r_0$, и алгоритм Евклида для пары чисел (b,r_0) , где $b > r_0$, требует $M - 1$ шагов. По предположению индукции имеем $b \geq F_{M+1}$ и $r_0 \geq F_M$. Следовательно, $a = q_0b + r_0 \geq b + r_0 \geq F_{M+1} + F_M = F_{M+2}$, что является искомым неравенством. Это доказательство, опубликованное Габриэлем Ламе в 1844 году, представляет собой начало теории сложности вычислений,[28] а также первое практическое применение чисел Фибоначчи.[27]

ТЕОРЕМА ЛАМЕ

- Число делений с остатком в процессе применения алгоритма Евклида не превосходит упятеренного количества цифр меньшего числа $\{ \text{DISPLAYSTYLE } b \}$, записанного в десятичной системе.[29]

СРЕДНЕЕ

- Существуют различные способы вычисления среднего количества шагов алгоритма. Первый способ вычисления — среднее время $T(a)$, необходимое для вычисления НОД заданного числа a и меньшего натурального числа b , выбранного с равной вероятностью из целых чисел от 0 до $a - 1$. [17]
- $$T(a) = \frac{1}{a} \sum_{0 \leq b < a} T(a, b)$$
- Однако, поскольку $T(a, b)$ сильно колеблется в зависимости от НОД двух чисел, усреднённая функция $T(a)$ также является «шумной». [30] Для того, чтобы уменьшить этот шум, второе среднее $t(a)$ берётся по всем взаимно простым числам c a .
- $$\tau(a) = \frac{1}{\varphi(a)} \sum_{\substack{0 \leq b < a \\ \gcd(a, b) = 1}} T(a, b)$$
- где $\varphi(a)$ функция Эйлера. Это среднее плавно растёт с ростом a . [31]
- $$\tau(a) = \frac{12}{\pi^2} \ln^2 \ln a + C + O(a^{-1/6 - \varepsilon})$$
- Константа (константа Портера [32]) в этой формуле $C \approx 1.467$, ε является бесконечно малым.
- Третье среднее значение $Y(n)$ определяется как среднее число шагов, требуемых, когда a и b выбираются случайным образом (с равномерным распределением) от 1 до n .
- $$Y(n) = \frac{1}{n^2} \sum_{a=1}^n \sum_{b=1}^n T(a, b) = \frac{1}{n} \sum_{a=1}^n T(a)$$

ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ ШАГА

- На каждом шаге алгоритма Евклида вычисляется коэффициент q_k и остаток r_k для заданной пары целых чисел r_{k-2} и r_{k-1} . Эти величины связаны следующим соотношением:

$$r_{k-2} = q_k r_{k-1} + r_k$$

- Вычислительная сложность каждого шага связана главным образом с нахождением q_k , так как остаток r_k можно быстро вычислить используя r_{k-2} , r_{k-1} , и q_k

$$r_k = r_{k-2} - q_k r_{k-1}$$

- Вычислительная сложность операции деления чисел размером n бит оценивается как $O(n^{\epsilon+1})$, где ϵ — размер частного.[24]

- Для сравнения, исходный алгоритм Евклида, с использованием вычитания, может быть намного медленнее. В большинстве случаев коэффициент q_k является малым числом. Вероятность данного частного q примерно равна $\ln|u|/(u-1)$, где $u = (q+1)^2$. [33] Для иллюстрации вероятность частного 1, 2, 3 или 4 составляет примерно 41,5%, 17,0%, 9,3% и 5,9% соответственно. Так как операция вычитания быстрее, чем деление, особенно для чисел больше одного машинного слова, [34] алгоритм Евклида с использованием вычитания может быть более конкурентоспособным в сравнении с алгоритмом, использующим деление. [35] Это используется в бинарном алгоритме вычисления НОД. [36]

- Оценка сложности алгоритма вычисляется как произведение количества шагов на время выполнения одного шага. Она показывает, что алгоритм Евклида растёт квадратично $O(n^2)$, где n — среднее число цифр в двух начальных числах a и b в десятичной записи. Пусть n_0, n_1, \dots, n_{N-1} представляют число цифр в последовательных остатках r_0, r_1, \dots, r_{N-1} . Так как число шагов N растёт линейно с n , время работы ограничено следующим выражением:

$$\{ \text{\textit{DISPLAYSTYLE}} O \{ \text{\textit{BIG}} \} \sum_{i < N} n_i (n_{i-1} - n_{i+1} + 2) \{ \text{\textit{BIG}} \} \} \text{\textit{SUBSETEQ}} O \{ \text{\textit{BIG}} \} \sum_{i < N} (n_{i-1} - n_{i+1} + 2) \{ \text{\textit{BIG}} \} \} \text{\textit{SUBSETEQ}} O(n \{ \text{\textit{BIG}} \} + 2N) \text{\textit{SUBSETEQ}} O(n^2 \{ \text{\textit{BIG}} \} \} \{ \text{\textit{DISPLAYSTYLE}} O \{ \text{\textit{BIG}} \} \sum_{i < N} n_i (n_{i-1} - n_{i+1} + 2) \{ \text{\textit{BIG}} \} \} \text{\textit{SUBSETEQ}} O \{ \text{\textit{BIG}} \} \sum_{i < N} (n_{i-1} - n_{i+1} + 2) \{ \text{\textit{BIG}} \} \} \text{\textit{SUBSETEQ}} O(n \{ \text{\textit{BIG}} \} + 2N) \text{\textit{SUBSETEQ}} O(n^2 \{ \text{\textit{BIG}} \} \} \}$$