Отчёт по лабораторной работе №4.

Вычисление наибольшего общего делителя

Коне Сирики

22 октября 2024

Российский университет дружбы народов, Москва, Россия

Объединённый институт ядерных исследований, Дубна, Россия

Информация

Докладчик

- Коне Сирики
- Студент физмат
- профессор кафедры прикладной информатики и теории вероятностей
- Российский университет дружбы народов
- konesirisil@yandex.ru
- https://github.com/skone19

Целью данной лабораторной работы является ознакомление с двумя алгоритмами нахождения наибольшего общего делителя их расширениями для нахождения его линейного представления, а также их последующая программная реализация.

Задачи: Рассмотреть и реализовать на языке программирования Python:

1. Алгоритм Евклида;
2. Бинарный алгоритм Евклида;
3. Расширенный алгоритм Евклида;
4. Расширенный бинарный алгоритм Евклида.

Теоретическое введение

Пусть \$x\$ и \$y\$ -- целые числа. Говорят, что \$x\$ \textit{делит} \$y\$, если с

Пусть \$a, b \in \mathbb{Z}\$. Целое число \$d\$ называется \textit{наибольшим

\item \$d \; | \; a\$ и \$d \; | \; b\$ (т.е. \$d\$ -- общий делитель \$a\$ и \$b\$);

\item если \$d'\$ -- общий делитель \$a\$ и \$b\$, то \$d' \; | \; d\$.

Линейное представление НОД

Наибольший общий делитель двух целых чисел \$a, b\$ существует и представляет

Алгоритм Евклида (1 / 2)

Алгоритм Евклида для нахождения НОД(a, b) при $a \geq b > 0$ основывается на следующем результате:

Если $a = bq + r$, то НОД(a, b) = НОД(b, r).

Строится последовательность чисел $a > b > r_1 > r_2 > ... > r_{n-1} > r_n \geq 0$, где r_k – остаток от деления двух предыдущих чисел, т.е. $r_{k-2} = r_{k-1}q_{k-1} + r_k$. Тогда НОД$(a, b)$ равен последнему ненулевому члену последовательности.

Вход. Целые числа a, b; $0 < b \leq a$.

Выход. $d = \text{НОД}(a, b)$

1. Положить $r_0 \leftarrow a, r_1 \leftarrow b, i \leftarrow 1$.
2. Найти остаток r_{i+1} от деления r_{i-1} на r_i.
3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i$. В противном случае положить $i \leftarrow i + 1$ и вернуться на шаг 2.
4. Результат: d.

Рис. 1: Алгоритм Евклида

Бинарный алгоритм Евклида основан на следующих свойствах наибольшего общего делителя ($0 < b \leq a$):

- если оба числа a и b чётные, то НОД$(a, b) = 2 \cdot$ НОД$(\frac{a}{2}, \frac{b}{2})$;
- если число a – нечётное, число b – чётное, то НОД$(a, b) =$ НОД$(a, \frac{b}{2})$;
- если оба числа a и b нечётные, то НОД$(a, b) =$ НОД$(a - b, b)$;
- если $a = b$, то НОД$(a, b) = a$.

Вход. Целые числа a, b; $0 < b \leq a$.

Выход. $d = \text{НОД}(a, b)$

1. Положить $g \leftarrow 1$.
2. Пока оба числа a и b чётные, выполнять $a \leftarrow \frac{a}{2}$, $b \leftarrow \frac{b}{2}$, $g \leftarrow 2g$ до получения хотя бы одного нечётного значения a или b.
3. Положить $u \leftarrow a$, $v \leftarrow b$.
4. Пока $u \neq 0$ выполнять следующие действия:
   4.1. Пока u чётное, полагать $u \leftarrow \frac{u}{2}$.
   4.2. Пока v чётное, полагать $v \leftarrow \frac{v}{2}$.
   4.3. При $u >= v$ положить $u \leftarrow u - v$. В противном случае положить $v \leftarrow v - u$.
5. Положить $d \leftarrow gv$.
6. Результат: d.

Вход. Целые числа a, b; $0 < b \le a$.

Выход. $d = \text{НОД}(a, b)$; такие целые числа x, y, что $ax + by = d$.

1. Положить $r_0 \leftarrow a, r_1 \leftarrow b, x_0 \leftarrow 1, x_1 \leftarrow 0, y_0 \leftarrow 0, y_1 \leftarrow 1, i \leftarrow 1$.
2. Разделить с остатком r_{i-1} на r_i: $r_{i-1} = q_i r_i + r_{i+1}$.
3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i, x \leftarrow x_i, y \leftarrow y_i$. В противном случае положить $x_{i+1} \leftarrow x_{i-1} - q_i x_i, y_{i+1} \leftarrow y_{i-1} - q_i y_i, i \leftarrow i + 1$ и вернуться на шаг 2.
4. Результат: d, x, y.

Расширенный бинарный алгоритм Евклида

Вход. Целые числа a, b; $0 < b \le a$.

Выход. $d = \text{НОД}(a, b)$; такие целые числа x, y, что $ax + by = d$.

1. Положить $g \leftarrow 1$.
2. Пока числа a и b чётные, выполнять $a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, g \leftarrow 2g$ до получения хотя бы одного нечётного значения a или b.
3. Положить $u \leftarrow a, v \leftarrow b, A \leftarrow 1, B \leftarrow 0, C \leftarrow 0, D \leftarrow 1$.
4. Пока $u \ne 0$ выполнять следующие действия:

   4.1. Пока u чётное:

   4.1.1. Положить $u \leftarrow \frac{u}{2}$.

   4.1.2. Если оба числа A и B чётные, то положить $A \leftarrow \frac{A}{2}$, $B \leftarrow \frac{B}{2}$. В противном случае положить $A \leftarrow \frac{A+b}{2}, B \leftarrow \frac{B-a}{2}$.

   4.2. Пока v чётное:

   4.2.1. Положить $v \leftarrow \frac{v}{2}$.

   4.2.2. Если оба числа C и D чётные, то положить $C \leftarrow \frac{C}{2}$, $D \leftarrow \frac{D}{2}$. В противном случае положить $C \leftarrow \frac{C+b}{2}, D \leftarrow \frac{D-a}{2}$.

   4.3. При $u \ge v$ положить $u \leftarrow u - v, A \leftarrow A - C, B \leftarrow B - D$. В противном случае положить $v \leftarrow v - u, C \leftarrow C - A, D \leftarrow D - B$.
5. Положить $d \leftarrow gv, x \leftarrow C, y \leftarrow D$.
6. Результат: d, x, y.

Рис. 4: Расширенный бинарный алгоритм Евклида

Ход выполнения и результаты

Алгоритм Евклида. Реализация

```python
def is_even(a):
    return (True if a % 2 == 0 else False)

def euclidean_algorithm(a, b):
    (a, b) = (abs(int(a)), abs(int(b)))
    if b > a:
        (a, b) = (b, a)
    r = [a, b] # шаг 1
    while r[1] != 0: # шаги 2-3
        (r[0], r[1]) = (r[1], r[0] % r[1])
    return r[0] # шаг 4
```

Алгоритм Евклида. Результаты

```
print("НОД({}, {}) = {}".format(12345, 24690, euclidean_algorithm(12345, 24690)))
print("НОД({}, {}) = {}".format(12345, 54321, euclidean_algorithm(12345, 54321)))
print("НОД({}, {}) = {}".format(12345, 12541, euclidean_algorithm(12345, 12541)))
print("НОД({}, {}) = {}".format(99, 121, euclidean_algorithm(99, 121)))
```

[2] ✓ 0.3s

```
НОД(12345, 24690) = 12345
НОД(12345, 54321) = 3
НОД(12345, 12541) = 1
НОД(99, 121) = 11
```

Рис. 5: Примеры нахождения НОД двух чисел с помощью программной реализации алгоритма Евклида

Бинарный алгоритм Евклида. Реализация

```
def euclidean_algorithm_binary(a, b): <...>
    g = 1 # шаг 1
    while is_even(a) and is_even(b): # шаг 2
        (a, b, g) = (int(a / 2), int(b / 2), 2 * g)
    (u, v) = (a, b) # шаг 3
    while u != 0: # шаг 4
        while is_even(u):
            u = int(u / 2)
        while is_even(v):
            v = int(v / 2)
        if u >= v:
            u -= v
        else:
            v -= u
```

Бинарный алгоритм Евклида. Результаты

```python
print("НОД({}, {}) = {}".format(12345, 24690, euclidean_algorithm_binary(12345, 24690)))
print("НОД({}, {}) = {}".format(12345, 54321, euclidean_algorithm_binary(12345, 54321)))
print("НОД({}, {}) = {}".format(12345, 12541, euclidean_algorithm_binary(12345, 12541)))
print("НОД({}, {}) = {}".format(24, 56, euclidean_algorithm_binary(24, 56)))
```

[4] ✓ 0.4s

```
НОД(12345, 24690) = 12345
НОД(12345, 54321) = 3
НОД(12345, 12541) = 1
НОД(24, 56) = 8
```

Рис. 6: Примеры нахождения НОД двух чисел с помощью программной реализации бинарного алгоритма Евклида

Расширенный алгоритм Евклида. Реализация

```
def euclidean_algorithm_extended(a, b):
    (a, b) = (abs(int(a)), abs(int(b)))
    reversed = True if b > a else False
    (a, b) = (b, a) if reversed else (a, b)
    (r, x, y) = ([a, b], [1, 0], [0, 1]) # шаг 1
    while r[1] != 0: # шаги 2-3
        (r[0], r[1], q) = (r[1], r[0] % r[1], r[0] // r[1])
        if r[1] != 0: # если остаток ещё не нулевой..
            (x[0], x[1]) = (x[1], x[0] - q * x[1])
            (y[0], y[1]) = (y[1], y[0] - q * y[1])
    (d, x_r, y_r) = (r[0], x[1], y[1])
    if reversed:
        (x_r, y_r) = (y_r, x_r)
    return (d, x_r, y_r)
```

```python
(d, x, y) = euclidean_algorithm_extended(12345, 24690)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 24690, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_extended(12345, 54321)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 54321, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_extended(12345, 12541)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 12541, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_extended(39, 169)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 39, b = 169, d = d, x = x, y = y))
```

[6] ✓ 0.4s

```
НОД(12345, 24690) = 12345 = 12345 * 1 + 24690 * 0
НОД(12345, 54321) = 3 = 12345 * 3617 + 54321 * -822
НОД(12345, 12541) = 1 = 12345 * 4159 + 12541 * -4094
НОД(39, 169) = 13 = 39 * -4 + 169 * 1
```

Рис. 7: Примеры нахождения НОД двух чисел и его линейного представления с помощью программной реализации расширенного алгоритма Евклида

Расширенный бинарный алгоритм Евклида. Реализация

```python
def euclidean_algorithm_binary_extended(a, b):
    <...>
    g = 1 # шаг 1
    while is_even(a) and is_even(b): # шаг 2
        (a, b, g) = (int(a / 2), int(b / 2), 2 * g)
    (u, v, A, B, C, D) = (a, b, 1, 0, 0, 1) # шаг 3
    while u != 0: # шаг 4
        while is_even(u): # шаг 4.1
            u = int(u / 2) # шаг 4.1.1
            if is_even(A) and is_even(B): # шаг 4.1.2
                (A, B) = (int(A / 2), int(B / 2))
            else:
                (A, B) = (int((A + b) / 2), int((B - a) / 2))
```

```
        while is_even(v): # шаг 4.2
            v = int(v / 2) # шаг 4.2.1
            if is_even(C) and is_even(D): # шаг 4.2.2
                (C, D) = (int(C / 2), int(D / 2))
            else:
                (C, D) = (int((C + b) / 2), int((D - a) / 2))
        if u >= v: # шаг 4.3
            (u, A, B) = (u - v, A - C, B - D)
        else:
            (v, C, D) = (v - u, C - A, D - B)
    (d, x, y) = (g * v, C, D) # шаг 5
    if reversed:
        (x, y) = (y, x)
    return (d, x, y)
```

```
(d, x, y) = euclidean_algorithm_binary_extended(12345, 24690)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 24690, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_binary_extended(12345, 54321)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 54321, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_binary_extended(12345, 12541)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 12345, b = 12541, d = d, x = x, y = y))

(d, x, y) = euclidean_algorithm_binary_extended(190, 342)
print("НОД({a}, {b}) = {d} = {a} * {x} + {b} * {y}".format(a = 190, b = 342, d = d, x = x, y = y))
```

[8] ✓ 0.3s

```
...
НОД(12345, 24690) = 12345 = 12345 * 1 + 24690 * 0
НОД(12345, 54321) = 3 = 12345 * -32597 + 54321 * 7408
НОД(12345, 12541) = 1 = 12345 * -8382 + 12541 * 8251
НОД(190, 342) = 38 = 190 * 11 + 342 * -6
```

Рис. 8: Примеры нахождения НОД двух чисел и его линейного представления с помощью программной реализации расширенного бинарного алгоритма Евклида

Заключение

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с двумя алгоритмами нахождения наибольшего общего делителя – алгоритмом Евклида, бинарным алгоритмом Евклида, – и их расширенными версиями для нахождения линейного представления этого делителя, после чего все четыре алгоритма были успешно реализованы на языке программирования **Python**.

Спасибо за внимание