Отчёт по лабораторной работе №4. Вычисление наибольшего общего делителя

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Леонова Алина Дмитриевна, 1032212306
Группа: НФИмд-01-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

4 декабря, 2021, Москва

Цель и задание работы

Цель работы

Целью данной работы является ознакомление с четырьмя алгоритмами вычисления наибольшего общего делителя и их реализация на выбранном языке программирования.

Задание

Реализовать программно алгоритмы вычисления наибольшего общего делителя:

- алгоритм Евклида
- бинарный алгоритм Евклида
- расширенный алгоритм Евклида
- расширенный бинарный алгоритм Евклида

Теоретическое введение

Алгоритм Евклида

- НОД(A,0) = A
- НОД(0,B) = B
- Если A = B*Q + R и B≠0, то НОД(A, B) = НОД(B, R),

где Q — целое число, а R — целое число от 0 до B-1

Бинарный алгоритм Евклида

Основан на использовании следующих свойств НОД:

- НОД(2m, 2n) = 2 НОД(m, n),
- НОД(2m, 2n+1) = НОД(m, 2n+1),
- НОД(-m, n) = НОД(m, n)

Выполнение лабораторной работы

Функция для проверки

```python
# Функция для проверки разных реализаций вычисления НОД(a,b)
def check(nod_func):
    print(nod_func(0, 105))
    print(nod_func(1, 105))
    print(nod_func(91, 105))
    print(nod_func(100000, 100))
    print(nod_func(12345, 678))
    print(nod_func(12345, 24690))

```

Figure 1: Функция check

Функция для проверки функций вычисления НОД(a,b) на 6 парах целых чисел (см. рис. 1).

Алгоритм Евклида

```
10
11  # 1. Алгоритм Евклида
12  def nod1(a, b):
13      if a == 0 or b == 0:
14          return max(a, b)
15      if a == 1 or b == 1:
16          return 1
17      if a < b:
18          a, b = b, a
19
20      d = nod1(a % b, b)
21      return d
22
```

Figure 2: Функция nod1

Функция, реализующая вычисление НОД(a,b) с помощью алгоритма Евклида (см. рис. 2).

Бинарный алгоритм Евклида

```
23
24  # 2. Бинарный алгоритм Евклида
25  def nod2(a, b):
26      if a == 0 or b == 0:
27          return max(a, b)
28      if a == 1 or b == 1:
29          return 1
30      if a < b:
31          a, b = b, a
32
33      g = 1
34      if a % 2 == 0 and b % 2 == 0:
35          a /= 2
36          b /= 2
37          g *= 2
38
39      d = int( g * nod2(a - b, b) )
40      return d
41
```

Figure 3: Функция nod2

Функция, реализующая вычисление НОД(a,b) с помощью бинарного алгоритма Евклида (см. рис. 3).

Расширенный алгоритм Евклида

```
42
43  # 3. Расширенный алгоритм Евклида
44  # d = НОД(a,b) = ax + by
45  def nod3(a, b):
46      if a == 0 or b == 0:
47          return max(a, b)
48      if a == 1 or b == 1:
49          return 1
50      if a < b:
51          a, b = b, a
52
53      x, y = [1,0], [0,1]
54      a_, b_ = a, b
55
56      while b_ != 0:
57          a_, b_, p = b_, a_ % b_, a_ // b_
58
59          if b_ != 0:
60              x[0], x[1] = x[1], x[0] - p*x[1]
61              y[0], y[1] = y[1], y[0] - p*y[1]
62
63      d = a_
64      print(a,'*',x[1],' + ',b,'*',y[1],' = ',d)
65      return d
66
```

Figure 4: Функция nod3

Функция, реализующая вычисление НОД(a,b) с помощью расширенного алгоритма Евклида (см. рис. 4).

Расширенный бинарный алгоритм Евклида

```
77      g = 1
78      while a % 2 == 0 and b % 2 == 0:
79          a /= 2
80          b /= 2
81          g *= 2
82
83      a, b, g = int(a), int(b), int(g)
84      u, v, A, B, C, D = a, b, 1 , 0, 0, 1
85
86      while u != 0:
87          while u % 2 == 0:
88              u /= 2
89              if A % 2 == 0 and B % 2 == 0:
90                  A /= 2
91                  B /= 2
92              else:
93                  A = (A + b) / 2
94                  B = (B - a) / 2
95
96          while v % 2 == 0:
97              v /= 2
98              if C % 2 == 0 and D % 2 == 0:
99                  C /= 2
100                 D /= 2
101             else:
102                 C = (C + b) / 2
103                 D = (D - a) / 2
104
105         if u >= v:
106             u, A, B = u-v, A-C, B-D
107         else:
108             v, C, D = v-u, C-A, D-B
109
110     A, B, C, D = int(A), int(B), int(C), int(D)
111     d = int( g * v )
112     print(a,'*',C,' + ',b,'*',D,' = ',d)
113     return d
```

Figure 5: Часть функции nod4

Проверка работы

```
115
116  print('Алгоритм Евклида')
117  check(nod1)
118
119  print('\nБинарный алгоритм Евклида')
120  check(nod2)
121
122  print('\nРасширенный алгоритм Евклида')
123  check(nod3)
124
125  print('\nРасширенный бинарный алгоритм Евклида')
126  check(nod4)
127
```

Figure 6: Проверка работы

Вызов проверок работы всех реализованных функций на шести разных вариантах входных параметров, задаваемых в функции check (см. рис. 6).

Результат работы реализованных вариаций алгоритма Евклида

```
In [1]: runfile('E:/GitHub/1.2-IS/Lab_4/
L4_Leonova.py', wdir='E:/GitHub/1.2-IS/Lab_4')
Алгоритм Евклида
105
1
7
100
3
12345

Бинарный алгоритм Евклида
105
1
7
100
3
12345

Расширенный алгоритм Евклида
105
1
105 * -6  +  91 * 7  =  7
7
100000 * 0  +  100 * 1  =  100
100
12345 * 101  +  678 * -1839  =  3
3
24690 * 0  +  12345 * 1  =  12345
12345

Расширенный бинарный алгоритм Евклида
105
1
105 * -6  +  91 * 7  =  7
7
25000 * 0  +  25 * 1  =  100
100
12345 * -125  +  678 * 2276  =  3
3
24690 * 0  +  12345 * 1  =  12345
12345

In [2]:
```

Figure 7: Результат выполнения L3_Leonova.py

Выводы

Цель лабораторной работы была достигнута, четыре алгоритмама вычисления наибольшего общего делителя были реализованы на языке программирования Python.