Отчет по лабораторной работе №4

Дисциплина: Математические основы защиты информации и информационной безопасности

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

List of Figures

1 Цель работы

Ознакомится и реализовать алгоритмы вычисления наибольшего общего делителя.

2 Задание

Реализовать такие алгоритмы вычисления наибольшего общего делителя, как алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида и расширенный бинарный алгоритм Евклида.

3 Теоретическое введение

Пусть числа a и b целые и $b \neq 0$. Разделить a на b с остатком – значит представить a в виде $a = qb + r$,где $q, r \in Z$ и $0 \leq r \leq |b|$ Число q называется неполным частным, число r – неполным остатком от деления a на b. Целое число $d \neq 0$ называется наибольшим общим делителем целых чисел $a_1, a_2, ..., a_k$ (обозначается d = НОД($a_1, a_2, ..., a_k$)), если выполняются следующие условия: 1. каждое из чисел $a_1, a_2, ..., a_k$ делится на d;
2. если $d_1 \neq 0$ – другой общий делитель чисел □$a_1, a_2, ..., a_k$, то d делится на d_1.

Для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком, называемый алгоритмом Евклида.

1. Алгоритм Евклида.

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

   Выход. $d = (a, b)$.

   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.

2. Бинарный алгоритм Евклида.

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

   Выход. $d = (a, b)$.

   1. Положить $g \leftarrow 1$.

2. Пока оба числа a и b четные, выполнять $a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, g \leftarrow 2g$ до получения хотя бы одного нечетного значения a или b.
3. Положить $u \leftarrow a, v \leftarrow b$.
4. Пока $u \neq 0$ выполнять следующие действия:
   1. Пока u четное, полагать $u \leftarrow \frac{u}{2}$.
   2. Пока v четное, полагать $v \leftarrow \frac{v}{2}$.
   3. При $u \geq v$ положить $u ← u - v $. В противном случае положить $v ← v - u $.
5. Положить $d \leftarrow gv$.
6. Результат: d.

3. Расширенный алгоритм Евклида.

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

   Выход. $d = (a, b)$; 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.

4. Расширенный бинарный алгоритм Евклида.

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

   Выход. $d = (a, b)$.

   1. Положить $g \leftarrow 1$.
   2. Пока оба числа a и b четные, выполнять $a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, g \leftarrow 2g$ до получения хотя бы одного нечетного 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 $u \neq 0$ выполнять следующие действия:
      1. Пока u четное:

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

  2. Пока v четное:
    1. Положить $v \leftarrow \frac{v}{2}$.
    2. Если оба числа C и D четные, то положить $C \leftarrow \frac{C}{2}, D \leftarrow \frac{D}{2}$. В противном случае положить $C \leftarrow \frac{C+b}{2}, D \leftarrow \frac{D-a}{2}$.
  3. При $u \geq v$ 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.

4 Выполнение лабораторной работы

4.1 Шаг 1

Ознакомилась с предоставленными теоретическими данными. Для выполнения задания решила использовать язык Python. Написала функцию, выполняющую поиск НОД с помощью алгоритма Евклида. Код функции и результат ее использования представлен на Рисунке 1 (рис. - fig. 4.1). Функция принимает на вход числа a и b. При условии, что $0 < b \leq a$ реализуется алгоритм, представленный в теоретическом введении и функция возвращает НОД. Если условие $0 < b \leq a$ не выполняется, функция ничего не вернет, будет выведено соответствующее сообщение. Пример выполнения функции также показан на Рисунке 1 (рис. - fig. 4.1).

Алгоритм Евклида

```
In [9]: import numpy as np

In [28]: def algevc (a, b):
             if (a >= b and b > 0):
                 r=[]
                 r.append(a)
                 r.append(b)
                 i=1
                 d=0
                 while (d == 0):
                     r.append(r[i-1]%r[i])
                     if r[i+1]==0:
                         d=r[i]
                     else:
                         i=i+1
                 return d
             else:
                 print ("Ошибка, проверьте, что a больше или равно b, a b больше 0")

In [30]: algevc(100, 20)
Out[30]: 20
```

Figure 4.1: Реализация алгоритма Евклида

4.2 Шаг 2

На Рисунке 2 (рис. - fig. 4.2) представлен код функции, реализующий бинарный алгоритм Евклида, и пример выполнения.

Бинарный алгоритм Евклида

```
In [31]: def binalgevc (a, b):
             if (a >= b and b > 0):
                 g=1
                 while (a%2==0 and b%2==0):
                     a=a/2
                     b=b/2
                     g=2*g
                 u=a
                 v=b
                 while (u !=0):
                     if (u%2==0):
                         u=u/2
                     if (v%2==0):
                         v=v/2
                     if (u >= v):
                         u=u-v
                     else:
                         v=v-u
                 d=g*v
                 return d
             else:
                 print ("Ошибка, проверьте, что a больше или равно b, a b больше 0")

In [35]: binalgevc(100, 6)

Out[35]: 2.0
```

Figure 4.2: Реализация бинарного алгоритма Евклида

4.3 Шаг 3

На Рисунке 3 (рис. - fig. 4.3) представлен код функции, реализующий расширенный алгоритм Евклида, и пример выполнения. Данная функция в случае нахождения НОД выводит не только сам НОД, но и числа x и y, являющиеся коэффициентами уравнения $ax + by = d$.

Расширенный алгоритм Евклида

```
In [50]: def expalgevc (a, b):
             if (a >= b and b > 0):
                 r=[]
                 r.append(a)
                 r.append(b)
                 x=[]
                 x.append(1)
                 x.append(0)
                 y=[]
                 y.append(0)
                 y.append(1)
                 i=1
                 r.append(1)
                 while (r[i+1] != 0):
                     r[i+1]=r[i-1]%r[i]
                     q=r[i-1]//r[i]
                     if (r[i+1]==0):
                         d=r[i]
                         xx=x[i]
                         yy=y[i]
                     else:
                         x.append(x[i-1]-q*x[i])
                         y.append(y[i-1]-q*y[i])
                         i=i+1
                         r.append(1)
                 print (a, "*", xx, "+", b, "*", yy, "=", d)
                 return (d, xx, yy)
             else:
                 print ("Ошибка, проверьте, что a больше или равно b, a b больше 0")

In [52]: expalgevc(100, 7)

         100 * -3 + 7 * 43 = 1

Out[52]: (1, -3, 43)
```

Figure 4.3: Реализация расширенного алгоритма Евклида

4.4 Шаг 4

На Рисунке 4 (рис. - fig. 4.4) и Рисунке 5 (рис. - fig. 4.5) представлен код функции, реализующий расширенный бинарный алгоритм Евклида, и пример выполнения. Данная функция в случае нахождения НОД выводит не только сам НОД, но и числа x и y, являющиеся коэффициентами уравнения $ax + by = d$.

Расширенный бинарный алгоритм Евклида

```
In [58]: def expbinalgevc (a, b):
             if (a >= b and b > 0):
                 a_v=a
                 b_v=b
                 g=1
                 while (a%2==0 and b%2==0):
                     a=a/2
                     b=b/2
                     g=2*g
                 u=a
                 v=b
                 A=1
                 B=0
                 C=0
                 D=1
```

Figure 4.4: Реализация расширенного бинарного алгоритма Евклида

```
        while (u !=0):
            if (u%2==0):
                u=u/2
                if (A%2==0 and B%2==0):
                    A=A/2
                    B=B/2
                else:
                    A= (A+b)/2
                    B=(B-a)/2
            if (v%2==0):
                v=v/2
                if (C%2==0 and D%2==0):
                    C=C/2
                    D=D/2
                else:
                    C=(C+b)/2
                    D=(D-a)/2
            if (u >= v):
                u=u-v
                A=A-C
                B=B-D
            else:
                v=v-u
                C=C-A
                D=D-B
        d=g*v
        x=C
        y=D
        print (a_v, "*", x, "+", b_v, "*", y, "=", d)
        return(d,x,y)
    else:
        print ("Ошибка, проверьте, что a больше или равно b, a b больше 0")
```

```
In [60]: expbinalgevc(100, 7)

         100 * -3.0 + 7 * 43.0 = 1.0

Out[60]: (1.0, -3.0, 43.0)
```

Figure 4.5: Реализация расширенного бинарного алгоритма Евклида

5 Выводы

Я ознакомилась с алгоритмами нахождения НОД и реализовала их. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.