

Отчёт по лабораторной работе 4

МОЗИИБ

Папикян Гагик Тигранович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Алгоритм Евклида	7
3.2	Бинарный Алгоритм Евклида	8
4	Выполнение лабораторной работы	9
5	Выводы	13

List of Figures

4.1	Выполнение лабораторной работы	12
-----	--	----

List of Tables

1 Цель работы

Познакомиться с алгоритмами поиска Наибольшего Общего Делителя(НОД)

2 Задание

- 1) Реализовать алгоритм Евклида
- 2) Реализовать бинарный алгоритм Евклида
- 3) Реализовать расширенный бинарный алгоритм Евклида

3 Теоретическое введение

3.1 Алгоритм Евклида

Древнегреческие математики называли этот алгоритм $\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu$ или $\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega$ — «взаимное вычитание». Этот алгоритм не был открыт Евклидом, так как упоминание о нём имеется уже в Топике Аристотеля (IV век до н. э.). В «Началах» Евклида он описан дважды — в VII книге для нахождения наибольшего общего делителя двух натуральных чисел и в X книге для нахождения наибольшей общей меры двух однородных величин. В обоих случаях дано геометрическое описание алгоритма, для нахождения «общей меры» двух отрезков.

Историками математики было выдвинуто предположение, что именно с помощью алгоритма Евклида (процедуры последовательного взаимного вычитания) в древнегреческой математике впервые было открыто существование несоизмеримых величин (стороны и диагонали квадрата, или стороны и диагонали правильного пятиугольника). Впрочем, это предположение не имеет достаточных документальных подтверждений. Алгоритм для поиска наибольшего общего делителя двух натуральных чисел описан также в I книге древнекитайского трактата Математика в девяти книгах.

Суть алгоритма заключается в последовательном делении большего числа на меньшее. Если остаток от деления равен нулю, то делитель - это НОД. Иначе за большее числа принимается делитель, а за меньшее - остаток, а алгоритм повторяется.

3.2 Бинарный Алгоритм Евклида

Бинарный алгоритм Евклида использует сдвиги, а не деление, что дает прирост в производительности. Так же существует расширенный бинарный алгоритм Евклида, дающий значения x и y , удовлетворяющие уравнению $ax+by=d$

4 Выполнение лабораторной работы

Был написан следующий скрипт на javascript

```
// Алгоритм Евклида
const A = 106, B = 16
console.log(`A=${A} B=${B}`)
let a = A, b = B
let rem
while(rem != 0){
    rem = a%b
    a = b
    b = rem
}
console.log(`d=${a}`)
```

```
// Бинарный Алгоритм Евклида
let g = 1
a = A, b = B
while(!a%2 && !b%2){
    a /= 2
    b /= 2
    g *= 2
}
let u = a, v = b
```

```

while(u){
    if(!u%2){
        u/=2
    }
    if(!v%2){
        v/=2
    }
    if(u>=v){
        u -= v
    }else{
        v -= u
    }
}
console.log(`d=${g*v}`)

```

// Расширенный Бинарный Алгоритм Евклида

```

g = 1
a = A, b = B
while(!a%2 && !b%2){
    a /= 2
    b /= 2
    g *= 2
}
u = a, v = b
let a_ = 1, b_ = 0, c_ = 0, d_ = 1
while(u){
    if(!u%2){
        u/=2
        if(!a_%2 && !b_%2){

```

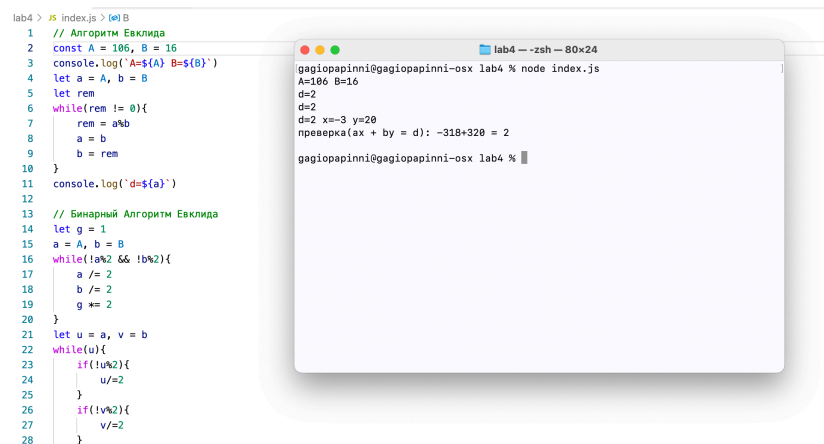
```

        a_ /= 2
        b_ /= 2
    }else{
        a_ = (a_+b) / 2
        b_ = (b_-a) / 2
    }
}
if(!v%2){
    v/=2
    if(!c_%2 && !d_%2){
        c_ /= 2
        d_ /= 2
    }else{
        c_ = (c_+b) / 2
        d_ = (d_-a) / 2
    }
}
if(u>=v){
    u -= v
    a_ -= c_
    b_ -= d_
}else{
    v -= u
    c_ -= a_
    d_ -= b_
}
}
console.log(`d=${g*v} x=${c_} y=${d_}`)
проверка(ax + by = d): ${A*c_}+${B*d_} = ${g*v}

```

`)

Результат исполнения скрипта приведен на рисунке 1 (рис. 4.1)



The image shows a code editor on the left and a terminal window on the right. The code editor contains a JavaScript script with two algorithms for finding the GCD of 106 and 16. The first algorithm is a standard Euclidean algorithm using a while loop with the remainder operation. The second algorithm is a binary version of the Euclidean algorithm. The terminal window shows the output of the script, which includes the values of A, B, d, and the result of a check.

```
lab4 > JS index.js > [9] B
1 // Алгоритм Евклида
2 const A = 106, B = 16
3 console.log(`A=${A} B=${B}`)
4 let a = A, b = B
5 let rem
6 while(rem != 0){
7   rem = a%b
8   a = b
9   b = rem
10 }
11 console.log(`d=${a}`)
12
13 // Бинарный Алгоритм Евклида
14 let g = 1
15 a = A, b = B
16 while(!a%2 || !b%2){
17   a /= 2
18   b /= 2
19   g *= 2
20 }
21 let u = a, v = b
22 while(u){
23   if(!u%2){
24     u/=2
25   }
26   if(!v%2){
27     v/=2
28   }
```

```
lab4 -- zsh -- 80x24
gagiopapinni@gagiopapinni-osx lab4 % node index.js
A=106 B=16
d=2
d=2
d=2 x=-3 y=20
проверка(ax + by = d): -318+328 = 2
gagiopapinni@gagiopapinni-osx lab4 %
```

Figure 4.1: Выполнение лабораторной работы

5 Выводы

Был реализован алгоритм Евклида, бинарный и расширенный бинарный алгоритмы Евклида. Для примера были использованы числа $A = 106$, $B = 16$, а на рис 4.1 видно, что их НОД = 2