Алгоритмы НОД - Лабораторная реализация

Этот документ описывает реализацию нескольких алгоритмов для вычисления наибольшего общего делителя (НОД) на Python. Включены следующие алгоритмы:

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

1. Алгоритм Евклида

Этот алгоритм использует деление с остатком для нахождения НОД двух целых чисел.

```
def gcd_euclid(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

2. Бинарный алгоритм Евклида

Бинарный алгоритм Евклида использует двоичные (побитовые) операции для нахождения НОД и более эффективен на компьютерах, так как быстро обрабатывает четные числа.

```
def gcd_binary(a, b):
    if a == 0:
        return b
    if b == 0:
        return a

    shift = 0
    while ((a | b) & 1) == 0:
        a >>= 1
        b >>= 1
        shift += 1

    while (a & 1) == 0:
        a >>= 1

    while b != 0:
        while (b & 1) == 0:
            b >>= 1

        if a > b:
            a, b = b, a

        b -= a
```

```
    return a << shift
```

3. Расширенный алгоритм Евклида

Этот алгоритм не только находит НОД двух чисел, но и коэффициенты x и y, такие, что ax + by= НОД(a,b)

```
def extended_gcd_euclid(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = extended_gcd_euclid(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y
```

4. Расширенный бинарный алгоритм Евклида

Расширенный бинарный алгоритм Евклида сочетает эффективность двоичных операций с расширенным НОД для нахождения коэффициентов x и y.

```
def extended_binary_gcd(a, b):
    if a == 0:
        return b, 0, 1

    if b == 0:
        return a, 1, 0

    shift = 0
    while ((a | b) & 1) == 0:
        a >>= 1
        b >>= 1
        shift += 1

    u, v = a, b
    x1, x2, y1, y2 = 1, 0, 0, 1

        while (u & 1) == 0:
            u >>= 1
            if (x1 | y1) & 1:
                x1 += b
                y1 -= a
            x1 >>= 1
            y1 >>= 1

        while (v & 1) == 0:
            v >>= 1
            if (x2 | y2) & 1:
```

```
                x2 += b
                y2 -= a
            x2 >>= 1
            y2 >>= 1

        if u >= v:
            u -= v
            x1 -= x2
            y1 -= y2
        else:
            v -= u
            x2 -= x1
            y2 -= y1

    return v << shift, x2, y2
```

Взаимодействие с пользователем

Ниже приведен скрипт на Python, который позволяет пользователю вводить значения для a и b, выбирать алгоритм и видеть результат.

```
def main():
    print("Введите два целых числа для нахождения НОД.")
    a = int(input("Введите число a: "))
    b = int(input("Введите число b: "))

    print("\nВыберите алгоритм для нахождения НОД:")
    print("1 - Алгоритм Евклида")
    print("2 - Бинарный алгоритм Евклида")
    print("3 - Расширенный алгоритм Евклида")
    print("4 - Расширенный бинарный алгоритм Евклида")
    print("5 - Все алгоритмы")

    choice = int(input("Введите номер выбранного алгоритма: "))

    if choice == 1:
        print("НОД (алгоритм Евклида):", gcd_euclid(a, b))
    elif choice == 2:
        print("НОД (бинарный алгоритм Евклида):", gcd_binary(a, b))
    elif choice == 3:
        gcd, x, y = extended_gcd_euclid(a, b)
        print(f"НОД (расширенный алгоритм Евклида): {gcd}, x = {x}, y = {y}")
    elif choice == 4:
        gcd, x, y = extended_binary_gcd(a, b)
        print(f"НОД (расширенный бинарный алгоритм Евклида): {gcd}, x = {x}, y =
{y}")
    elif choice == 5:
        print("НОД (алгоритм Евклида):", gcd_euclid(a, b))
        print("НОД (бинарный алгоритм Евклида):", gcd_binary(a, b))
        gcd, x, y = extended_gcd_euclid(a, b)
        print(f"НОД (расширенный алгоритм Евклида): {gcd}, x = {x}, y = {y}")
```

```
        gcd, x, y = extended_binary_gcd(a, b)
        print(f"НОД (расширенный бинарный алгоритм Евклида): {gcd}, x = {x}, y =
{y}")
    else:
        print("Неверный выбор. Пожалуйста, выберите число от 1 до 5.")

if __name__ == "__main__":
    main()
```