Отчёт по лабораторной работе №8

Целочисленная арифметика многократной точности

Федюшина Ярослава Андреевна

Содержание

Список иллюстраций

Цель работы

Получить навыки написания программного кода для реализации алгоритмов для выполнения арифметических операций

Задание

Реализовать программмно 5 алгоритмов для выполнения арифметических операций с большими целыми числами

Выполнение лабораторной работы

Python

def to_digits(num_str, b=10): return [int(d) for d in reversed(num_str)]

def from_digits(digits, b=10): return ''.join(str(d) for d in reversed(digits))

def normalize(num_digits): while len(num_digits) > 1 and num_digits[-1] == 0: num_digits.pop() return num_digits

def bigint_add(u_str, v_str, b=10):

```
u = to_digits(u_str, b)
v = to_digits(v_str, b)

# Делаем списки одинаковой длины
n = max(len(u), len(v))
u = u + [0] * (n - len(u))
v = v + [0] * (n - len(v))

w = [0] * (n + 1)
k = 0

for j in range(n):
    total = u[j] + v[j] + k
    w[j] = total % b
    k = total // b
```

```
w[n] = k

result_digits = normalize(w)
return from_digits(result_digits, b)
```

def bigint_subtract(u_str, v_str, b=10):

```
u = to_digits(u_str, b)
v = to_digits(v_str, b)

# Делаем списки одинаковой длины
n = max(len(u), len(v))
u = u + [0] * (n - len(u))
v = v + [0] * (n - len(v))

w = [0] * n
k = 0  # Заем

for j in range(n):
    diff = u[j] - v[j] + k
    if diff < 0:
        diff += b
        k = -1
    else:
        k = 0
    w[j] = diff

result_digits = normalize(w)
return from_digits(result_digits, b)
```

def bigint_multiply_column(u_str, v_str, b=10):

```
u = to_digits(u_str, b)
v = to_digits(v_str, b)

n = len(u)
m = len(v)
w = [0] * (m + n)

for j in range(m):
    if v[j] == 0:
        w[j] = 0
        continue

    k = 0
    for i in range(n):
        t = u[i] * v[j] + w[i + j] + k
        w[i + j] = t % b
        k = t // b

    w[n + j] = k

result_digits = normalize(w)
return from_digits(result_digits, b)
```

def bigint_multiply_fast(u_str, v_str, b=10):

```
u = to_digits(u_str, b)
v = to_digits(v_str, b)

n = len(u)
```

```
m = len(v)
w = [0] * (m + n)
t = 0

for s in range(m + n):

    sum_diag = 0
    for i in range(max(0, s - m + 1), min(n, s + 1)):
        j = s - i
        if 0 <= j < m:
            sum_diag += u[i] * v[j]

    sum_diag += t
    w[s] = sum_diag % b
    t = sum_diag // b

result_digits = normalize(w)
return from_digits(result_digits, b)
```

def bigint_divide(u_str, v_str, b=10):

```
u = [int(d) for d in u_str]
v = [int(d) for d in v_str]

n = len(u) - 1  # Индекс старшего разряда u
t = len(v) - 1  # Индекс старшего разряда v

# Проверка условий
if n < t:
    return "0", u_str
```

```
# Шаг 1: Инициализация частного
q = [0] * (n - t + 1)

# Шаг 2: Корректировка
# Создаем v * b^(n-t)
v_padded = v + [0] * (n - t)

# Пока u >= v_padded
while True:
    # Функция сравнения двух чисел в списках
    def compare_lists(a, b):
        # Удаляем ведущие нули
        a_norm = normalize(a.copy())
        b_norm = normalize(b.copy())

        if len(a_norm) > len(b_norm):
            return 1
        elif len(a_norm) < len(b_norm):
            return -1
        else:
            for i in range(len(a_norm)-1, -1, -1):
                if a_norm[i] > b_norm[i]:
                    return 1
                elif a_norm[i] < b_norm[i]:
                    return -1
            return 0

    if compare_lists(u, v_padded) >= 0:
```

```
            q[n - t] += 1
        u = to_digits(from_digits(to_digits(''.join(map(str, u))), b), b)
        v_padded_digits = to_digits(from_digits(''.join(map(str, v_padded)), b), b)
            u_digits = to_digits(from_digits(''.join(map(str, u)), b), b)

            # Делаем одинаковой длины
            max_len = max(len(u_digits), len(v_padded_digits))
            u_digits = u_digits + [0] * (max_len - len(u_digits))
                v_padded_digits = v_padded_digits + [0] * (max_len -
len(v_padded_digits))

            # Вычитаем
            borrow = 0
            result = []
            for i in range(max_len):
                diff = u_digits[i] - v_padded_digits[i] + borrow
                if diff < 0:
                    diff += b
                    borrow = -1
                else:
                    borrow = 0
                result.append(diff)

        u = [int(d) for d in reversed(from_digits(normalize(result), b))]
    else:
            break

# Шаг 3: Основной цикл
for i in range(n, t, -1):
```

```
    if i - t - 1 >= 0 and i - t - 1 < len(q):
        # Шаг 3.1
        if u[0] >= v[0]:
            q[i - t - 1] = b - 1
        else:
          q[i - t - 1] = (u[0] * b + (u[1] if len(u) > 1 else 0)) // v[0]

        # Шаг 3.2
       while q[i - t - 1] * (v[0] * b + (v[1] if len(v) > 1 else 0)) > \
         u[0] * b * b + (u[1] if len(u) > 1 else 0) * b + (u[2] if len(u) > 2 else 0):
            q[i - t - 1] -= 1

# Шаг 4: Остаток
r = ''.join(map(str, u))

# Формируем частное
quotient = ''.join(map(str, q))
quotient = quotient.lstrip('0') or '0'

return quotient, r
```

def test_algorithms(): print(“Тестирование алгоритмов для системы счисления с основанием 10”) print(“=” * 60)

```
# Тестовые данные
test_cases = [
    ("123", "456"),
    ("999", "1"),
    ("1000", "999"),
    ("54321", "12345"),
```

```
    ("999999", "888888"),
]

for u_str, v_str in test_cases:
    print(f"\nu = {u_str}, v = {v_str}")
    print("-" * 40)

    # Сложение
    sum_result = bigint_add(u_str, v_str)
    print(f"Сложение: {u_str} + {v_str} = {sum_result}")
    print(f"Проверка: {int(u_str) + int(v_str)}")

    # Вычитание (только если u >= v)
    if int(u_str) >= int(v_str):
        sub_result = bigint_subtract(u_str, v_str)
        print(f"Вычитание: {u_str} - {v_str} = {sub_result}")
        print(f"Проверка: {int(u_str) - int(v_str)}")

    # Умножение столбиком
    mul_col_result = bigint_multiply_column(u_str, v_str)
    print(f"Умножение столбиком: {u_str} * {v_str} = {mul_col_result}")
    print(f"Проверка: {int(u_str) * int(v_str)}")

    # Быстрое умножение
    mul_fast_result = bigint_multiply_fast(u_str, v_str)
    print(f"Быстрое умножение: {u_str} * {v_str} = {mul_fast_result}")
    print(f"Проверка: {int(u_str) * int(v_str)}")

    # Деление (только если v != 0)
```

```
    if int(v_str) != 0:
        div_result, mod_result = bigint_divide(u_str, v_str)
        print(f"Деление: {u_str} / {v_str} = {div_result}")
        print(f"Остаток: {mod_result}")
        print(f"Проверка частного: {int(u_str) // int(v_str)}")
        print(f"Проверка остатка: {int(u_str) % int(v_str)}")
```

def demo(): “” “Демонстрация работы алгоритмов” “” print(“Демонстрация работы алгоритмов”) print(“=” * 60)

```
# Пример 1
u = "123456789"
v = "987654321"
b = 10

print(f"Пример 1 (b={b}):")
print(f"u = {u}")
print(f"v = {v}")

# Сложение
result_add = bigint_add(u, v, b)
print(f"\n1. Сложение: {u} + {v} = {result_add}")

# Вычитание
result_sub = bigint_subtract(v, u, b)  # v > u
print(f"2. Вычитание: {v} - {u} = {result_sub}")

# Умножение столбиком
result_mul_col = bigint_multiply_column(u, v, b)
print(f"3. Умножение столбиком: {u} * {v} = {result_mul_col}")
```

```
# Быстрое умножение
result_mul_fast = bigint_multiply_fast(u, v, b)
print(f"4. Быстрое умножение: {u} * {v} = {result_mul_fast}")

# Деление
quotient, remainder = bigint_divide(v, u, b)
print(f"5. Деление: {v} / {u} = {quotient}")
print(f"   Остаток: {remainder}")

# Пример 2: Двоичная система (b=2)
print(f"\n{'='*60}")
print("Пример 2 (b=2):")
u_bin = "1101"  # 13 в десятичной
v_bin = "101"   # 5 в десятичной
b_bin = 2

print(f"u = {u_bin} (десятичное: {int(u_bin, 2)})")
print(f"v = {v_bin} (десятичное: {int(v_bin, 2)})")

result_add_bin = bigint_add(u_bin, v_bin, b_bin)
print(f"\n1. Сложение: {u_bin} + {v_bin} = {result_add_bin} (десятичное: {int(result

result_mul_bin = bigint_multiply_column(u_bin, v_bin, b_bin)
print(f"2. Умножение: {u_bin} * {v_bin} = {result_mul_bin} (десятичное: {int(result_
```

if **name** == “**main**”: # Запуск демонстрации demo()

```
print("Запуск тестирования всех алгоритмов:")
test_algorithms()
```

Результаты работы кода

Демонстрация работы алгоритмов

Пример 1 (b=10): u = 123456789 v = 987654321

1. Сложение: 123456789 + 987654321 = 1111111110
2. Вычитание: 987654321 - 123456789 = 864197532
3. Умножение столбиком: 123456789 * 987654321 = 121932631112635269
4. Быстрое умножение: 123456789 * 987654321 = 121932631112635269
5. Деление: 987654321 / 123456789 = 0 Остаток: 987654321

========== Пример 2 (b=2): u = 1101 (десятичное: 13) v = 101 (десятичное: 5)

1. Сложение: 1101 + 101 = 10010 (десятичное: 18)
2. Умножение: 1101 * 101 = 1000001 (десятичное: 65) Запуск тестирования всех алгоритмов: Тестирование алгоритмов для системы счисления с основанием 10 ==========

u = 123, v = 456 ... Деление: 999999 / 888888 = 1 Остаток: 111111 Проверка частного: 1 Проверка остатка: 111111

Вывод

В ходе выполнения данной лабораторной работы я приобрела практические навыки реализации алгоритмов арифметических операций с большими целыми числами