Mini 1

```
def divide(a: str, b: str) -> (int, int):
    a, b = a.lstrip("0"), b.lstrip("0")
    if b == '':
        raise ZeroDivisionError
    n, m = len(a), len(b)
    r = 0
    s = []
    for i in range(n):
        r = r * 10 + int(a[i])
        q = 0
        while q < 10 and q * int(b) <= r:
            q += 1
        q -= 1
        s.append(str(q))
        r -= q * int(b)
    if not s:
        s = 0
    else:
        s = int(''.join(s))
    return s, r


def test_divide():
    assert divide("12345", "12") == (1028, 9)

    assert divide("987654321", "1") == (987654321, 0)

    assert divide("12345678910111213141516171819202122232425262728293O", "31323334353637383
     divmod(12345678910111213141516171819202122232425262728293O, 31323334353637383940)

    assert divide("12345", "12345") == (1, 0)

    assert divide("100", "10") == (10, 0)

    assert divide("123", "12") == (10, 3)

    assert divide("123", "1000") == (0, 123)

    assert divide("0", "123") == (0, 0)

    try:
        divide("123", "0")
    except ZeroDivisionError:
```

```
        print("ZeroDivisionError caught")

    assert divide("00123", "012") == (10, 3)


test_divide()
```

Алгоритм требует по порядку O(m*n), так как:

1. Существует внешний цикл по длине делимого
2. Операция вычитания требует m элементарных операций
3. Существует коэффициент, возникающий при поиске q, но на порядок это не влияет

Лучший и Худший случаи

- Худший: делитель значительно меньше делимого. Оценка остается все той же - O(n * m)
- Лучший: делитель больше делимого, тогда на каждом шаге q будет определяться за одну итерацию и = 0. Оценка - O(n)