МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра теории вероятностей и компьютерного моделирования

Лабораторная работа № 4
По спецкурсу «Теория сложности алгоритмов»

Исследование временной сложности

Выполнил: Покхарел П.К.
Группа: М8О-101М-22, Вариант 6
Преподаватель: Рассказова В.А.

Москва, 2023

Задание. Для заданного языка

PRIMES = {m: m – бинарное простое число};

1. построить описание МТ с полиномиальной временной сложностью, решающей его;

2. построить оценку сложности данной МТ;

3. реализовать данную МТ в виде программы;

4. провести тестовые исследования, демонстрирующие совпадение фактической временной сложности с теоретической.

Описание МТ:

Он основан на малой теореме Ферма, которая гласит, что если *n* — простое число, то для любого *a*, которое не делится на *n*, выполняется равенство

$$a^{n-1} \equiv 1 \pmod{n}.$$

Тест простоты Ферма — вероятностный тест, который заключается в переборе нескольких значений *a*, если хотя бы для одного из них выполняется неравенство

$$a^{n-1} \not\equiv 1 \pmod{n},$$

то число *n* — составное. В противном случае, *n* — вероятно простое. Чем больше значений *a* использовано в тесте, тем выше вероятность того, что *n* — простое.

Алгоритм

Проверяем, что число n = 2 или n = 3:
    Если n <= 3, то:
        Если n > 1:
            Вернуть True
        Иначе:

Вернуть False

Выбираем случайное число a < n

Проверяем условие a^(n-1) mod n == 1:
Если a^(n-1) mod n != 1:
Вернуть False
Иначе:
Вернуть True

Сложность

$O(\log^2 n \times \log\log n \times \log\log\log n)$

Программная реализация

```
import random

def fermat_test(n):
   # Проверяем случай с n = 2 или 3
   if n <= 3:
       return True if n > 1 else False

   # Выбираем случайное число a меньше n
   a = random.randint(2, n - 2)

   # Проверяем условие a^(n-1) mod n == 1
   if pow(a, n - 1, n) != 1:
       return False

   return True

def is_prime(binary_number):
   # Преобразуем бинарное число в десятичное
   decimal_number = int(binary_number, 2)

   # Проверяем простоту числа с помощью теста Ферма
   if fermat_test(decimal_number):
      return "Бинарное число " + binary_number + " является простым."
   else:
      return "Бинарное число " + binary_number + " не является простым."

binary_number = input("Введите бинарное число для проверки на простоту: ")

print(is_prime(binary_number))

print()

print('Проверим простые числа')
test_prime = ['11', '10001', '110101', '11010011', '111110011', '1111010111']
for x in test_prime:
   print(is_prime(x))
```

```
print()

print('Проверим составные числа')
test_composite = ['110', '11011', '1001011', '10011100', '1000110000',
'1101111000']
for x in test_composite:
   print(is_prime(x))
```

Сравнение теоретической и практической временной сложности

Логарифм берем по основанию 2, n - длина входа

Тест 1

Вход 1: число 111 (вход длины 3):

теоретическая сложность O(0.98) ≈ O(1), практическая сложность 1.491816 сек

Вход 2: число 11011 (вход длины 5):

теоретическая сложность O(1.84), практическая сложность 1.370107 сек

Вывод: Практическая сложность увеличивается ~ в 1 раз при увеличении размера входа в 1,67 раза, что не превышает теоретической оценки сложности (~ в 1.84 раз)

Тест 2

Вход 1: число 111

теоретическая сложность O(1), практическая сложность 1.49 сек

Вход 2: число 110101

теоретическая сложность O(4.16), практическая сложность 1.92 сек

Вывод: Практическая сложность увеличивается ~ в 1.29 раз при увеличении размера входа в 2 раза, что не превышает теоретической оценки сложности (~ в 4.16 раз)

Тест 3

Вход 1: число 111

теоретическая сложность O(1), практическая сложность 1.49 сек

Вход 2: число 1101011

теоретическая сложность O(6.74), практическая сложность 2.10 сек

Вывод: Практическая сложность увеличивается ~ в 1.41 раз при увеличении размера входа в 2.33 раза, что не превышает теоретической оценки сложности (~ в 6.74 раз)

Тест 4

Вход 1: число 111

теоретическая сложность O(1), практическая сложность 1.49 сек

Вход 2: число 10011100

теоретическая сложность O(9.48), практическая сложность 2.17 сек

Вывод: Практическая сложность увеличивается ~ в 1.45 раз при увеличении размера входа в 2.67 раза, что не превышает теоретической оценки сложности (~ в 9.48 раз)

Тест 5

Вход 1: число 111: теоретическая сложность O(1), практическая сложность 1.491816 сек

Вход 2: число 111110011

теоретическая сложность O(12.29), практическая сложность 2.492344 сек

Вывод: Практическая сложность увеличивается ~ в 1.67 раз при увеличении размера входа в 3 раза, что не превышает теоретической оценки сложности (~ в 12.29 раз)