Ймовірнісні структури даних: фільтр Блума

Виконав Пристайчук Дмитро, ММШІ-1

Задача: Реалізація фільтра Блума

Структури даних

```
class BloomFilter:
    capacity: Integer           # Очікувана кількість елементів
    error_rate: Float           # Бажана ймовірність хибного спрацьовування
    size: Integer               # Розмір бітового масиву
    hash_count: Integer         # Кількість хеш-функцій
    bit_array: bitarray         # Бітовий масив
```

Математичні основи

Оптимальний розмір бітового масиву (m):

```
m = -n * ln(p) / (ln(2)²)
```

де:

- n - очікувана кількість елементів
- p - бажана ймовірність хибного спрацьовування

Оптимальна кількість хеш-функцій (k):

```
k = (m/n) * ln(2)
```

Алгоритм ініціалізації

```
def __init__(capacity, error_rate):
    # Обчислення оптимального розміру масиву
    size = -capacity * math.log(error_rate) / (math.log(2) ** 2)
    size = math.ceil(size)

    # Обчислення оптимальної кількості хеш-функцій
    hash_count = size / capacity * math.log(2)
    hash_count = math.ceil(hash_count)

    # Створення порожнього бітового масиву
```

```
    bit_array = bitarray(size)
    bit_array.setall(0)
```

Алгоритм додавання елемента

```
def add(item):
    # Обчислення позицій в бітовому масиві за допомогою k хеш-функцій
    for i in range(hash_count):
        index = mmh3.hash(str(item), i) % size
        bit_array[index] = 1
```

Алгоритм перевірки наявності елемента

```
def contains(item):
    # Перевірка чи встановлені всі відповідні біти
    for i in range(hash_count):
        index = mmh3.hash(str(item), i) % size
        if bit_array[index] == 0:
            return False  # Елемент точно відсутній

    return True  # Елемент можливо присутній
```

Аналіз ймовірності хибного спрацьовування

Ймовірність хибного спрацьовування після вставки n елементів:

```
p = (1 - e^(-kn/m))^k
```

Коли k = (m/n)ln(2), ймовірність помилки мінімізується до:

```
p = (1/2)^k = (0.6185)^(m/n)
```

Приклад виконання

```
Створено фільтр Блума:
- Розмір бітового масиву: 14378 біт
- Кількість хеш-функцій: 10
- Розрахована місткість: 1000 елементів
- Очікувана частота помилок: 0.001

Додано елементи: "apple", "banana", "orange"
```

```
Перевірка:
- "apple" присутній: Так
- "banana" присутній: Так
- "grape" присутній: Ні

Додано 1000 додаткових елементів

Тест хибного спрацьовування:
- "missing_item" присутній: Ні (але може бути "Так" через хибне спрацьовування)
```