САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №6
по курсу «Алгоритмы и структуры данных»
Тема: Хеширование. Хеш-таблицы

Выполнил:
Мезенцев Богдан
К 3139
Проверил:
Афанасьев Антон Владимирович

Санкт-Петербург
2024 г.

Содержание отчёта

Задачи по варианту

Задание 1. Множество

Реализация функции set_commands(), которая выполняет операции для множества и проверяет наличие элемента в этом множестве.

```python
def set_commands(A):
    """Выполняет операции для множества и проверяет наличие элемента в этом множестве"""
    set_num = set()
    res = []

    for operation, x in A:
        if operation == 'A':
            set_num.add(x)
        elif operation == 'D':
            set_num.discard(x)
        elif operation == '?':
            res.append('Y' if x in set_num else 'N')

    return res
```

Данная функция создает множество, в котором будут храниться все элементы, и список, куда будут добавляться элементы по команде “?”. После обработки всех полученных операций функция возвращает список res.

Задание 2. Телефонная книга

Реализация функции phone_book_commands(), которая выполняет операции для полученных контактов.

```python
def phone_book_commands(A):
    """Выполняет операции с 'телефонной книгой' и возвращает """
    phonebook = {}
    results = []

    for query in A:
        parts = query.split()
        command = parts[0]
        number = parts[1]

        if command == "add":
            name = parts[2]
            phonebook[number] = name  # Добавление номера
        elif command == "del":
            phonebook.pop(number, None)  # Удаление номера
        elif command == "find":
            results.append(phonebook.get(number, "not found"))  # Поиск номера

    return results
```

Данная функция использует словарь phonebook для удобного хранения контактов в формате: ключ - Имя контакта, значение - номер контакта. Полученные строки из файла с входными данными разделяются на части, чтобы отдельно работать с командами и данными контактов(имя и номер). Далее функция в зависимости от команды реализует соответствующее действие.

Задание 5. Выборы в США

Реализация функции count_votes(), которая подсчитывает голоса для каждого кандидата.

```python
def count_votes(A):
    """Подсчитывает количество голосов для каждого кандидата"""
    votes = {}

    for entry in A:
        candidate, votes_count = entry.rsplit(maxsplit=1)
        votes_count = int(votes_count)

        # Добавляем голоса к уже существующему кандидату или создаем новую запись
        if candidate in votes:
            votes[candidate] += votes_count
        else:
            votes[candidate] = votes_count

    return votes
```

Данная функция создает словарь, в котором будут хранится имена кандидатов и их голоса.
Считывая каждую строку из входного файла, функция суммирует количество голосов для каждого кандидата.
В результате возвращает суммарное количество голосов для кандидатов.

Задание 6. Фибоначчи возвращается

Реализация функции, которая проверяет являются ли полученные числа числами последовательности Фибоначчи или нет:

```
def fibonacci(A):

    def is_fibonacci_number(number):
        """Проверка, является ли число x числом Фибоначчи"""
        x = int(number)
        # Если хотя бы одно из этих выражений является полным квадратом, то это число Фибоначчи
        return math.isqrt(5 * x * x + 4) ** 2 == 5 * x * x + 4 or math.isqrt(5 * x * x - 4) ** 2 == 5 * x * x - 4

    result = []
    for number in A:
        if is_fibonacci_number(number):
            result.append("Yes")
        else:
            result.append("No")

    return result
```

Полученное передается в функцию is_fibonacci_number(), где оно подставляется в два выражения, чтобы проверить, можно ли получить с помощью этого числа полный квадрат. Так как, если хотя бы одно из выражений будет являться полным квадратом, то проверяемое число это число Фибоначчи.
Данная функция вызывается для всех полученных чисел.