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

Вариант 14

Выполнил: Мурашов Н.А.

К3141

Проверил: Афанасьев А.В.

Санкт-Петербург 2024 г.

# Задача 4:

# Текст задачи.

# 

# Листинг кода.

*"""Модуль для обработки операций над ассоциативным массивом."""*from collections import OrderedDict  
  
  
class AssocArrayProcessor:  
 *"""  
 Класс для обработки команд над ассоциативным массивом.  
  
 Поддерживаемые команды:  
 - put x y: добавить или обновить значение по ключу x  
 - get x: получить значение по ключу x или <none>  
 - prev x: получить значение по предыдущему ключу относительно x  
 - next x: получить значение по следующему ключу относительно x  
 - delete x: удалить ключ x  
 """* @staticmethod  
 def process\_commands(commands):  
 *"""  
 Обрабатывает список команд.  
  
 :param commands: Список строк команд.  
 :return: Список результатов для команд get, prev, next.  
 """* assoc\_array = OrderedDict()  
 results = []  
  
 for line in commands:  
 parts = line.strip().split()  
 command = parts[0]  
  
 if command == "put":  
 x, y = parts[1], parts[2]  
 assoc\_array[x] = y  
  
 elif command == "get":  
 x = parts[1]  
 results.append(assoc\_array.get(x, "<none>"))  
  
 elif command == "prev":  
 x = parts[1]  
 if x in assoc\_array:  
 keys = list(assoc\_array.keys())  
 idx = keys.index(x)  
 if idx > 0:  
 results.append(assoc\_array[keys[idx - 1]])  
 else:  
 results.append("<none>")  
 else:  
 results.append("<none>")  
  
 elif command == "next":  
 x = parts[1]  
 if x in assoc\_array:  
 keys = list(assoc\_array.keys())  
 idx = keys.index(x)  
 if idx < len(keys) - 1:  
 results.append(assoc\_array[keys[idx + 1]])  
 else:  
 results.append("<none>")  
 else:  
 results.append("<none>")  
  
 elif command == "delete":  
 x = parts[1]  
 assoc\_array.pop(x, None)  
  
 return results

**Текстовое объяснение решения.**

**Для реализации ассоциативного массива использован класс** OrderedDict **из библиотеки** collections**, который поддерживает порядок добавления элементов. В классе** AssocArrayProcessor **реализованы методы для обработки команд: -** put x y **— добавляет или обновляет элемент по ключу. -** get x **— возвращает значение по ключу или** <none>**, если ключ отсутствует. -** prev x **и** next x **— возвращают значения по соседним ключам. Если такого ключа нет, возвращается** <none>**. -** delete x **— удаляет ключ.**

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.004003 сек | 0.30МБ |
| Пример из задачи | 0.05832 сек | 0.52 МБ |
| Пример из задачи | 0.09345 сек | 1.45 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.42455 сек | 1.99 МБ |

**Вывод по задаче.**

**Решение эффективно обрабатывает команды для ассоциативного массива с сохранением порядка вставки. Использование** OrderedDict **обеспечивает быструю обработку операций с ключами.**

## Задача 5:

# Текст задачи.

# 

# Листинг кода.

*"""Модуль для обработки результатов выборов."""*class ElectionsProcessor:  
 *"""  
 Класс для подсчёта голосов кандидатов.  
 Принимает список пар (кандидат, количество\_голосов).  
 Возвращает список строк с результатами, отсортированных по имени кандидата.  
 """* def process\_elections(self, data):  
 *"""  
 Обрабатывает результаты выборов.  
  
 :param data: Список кортежей (имя\_кандидата, количество\_голосов).  
 :return: Список строк вида "кандидат голосов\n".  
 """* votes = {}  
 for candidate, vote\_count in data:  
 votes[candidate] = votes.get(candidate, 0) + int(vote\_count)  
  
 sorted\_candidates = sorted(votes.items())  
 res = [f"{candidate} {count}\n" for candidate, count in sorted\_candidates]  
 return res

**Текстовое объяснение решения.**

Задача сводится к подсчету голосов для каждого кандидата, что можно решить с помощью словаря, где ключами будут имена кандидатов, а значениями — количество голосов. После обработки всех данных, список кандидатов сортируется по имени, и выводится результат в нужном формате.

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.003465 сек | 0.35 МБ |
| Пример из задачи | 0.03958 сек | 0.56 МБ |
| Пример из задачи | 0.10345 сек | 1.59 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.4124 сек | 1.89 МБ |

**Вывод по задаче.**  
Решение успешно справляется с задачей подсчета голосов и их сортировки. Преимущество использования словаря — это возможность быстро обновлять количество голосов для каждого кандидата, а сортировка по имени производится стандартными средствами Python.

## Задача 6:

# Текст задачи.

# 

# Листинг кода.

*"""Модуль для проверки, является ли число числом Фибоначчи."""*class FibonacciChecker:  
 *"""  
 Класс для проверки, является ли число числом Фибоначчи.  
 """* def process\_fibonacci(self, numbers):  
 *"""  
 Для списка чисел возвращает "Yes" или "No" для каждого,  
 в зависимости от того, является ли число числом Фибоначчи.  
  
 :param numbers: Список чисел.  
 :return: Список строк "Yes\n" или "No\n".  
 """* results = []  
 for num in numbers:  
 if self.\_is\_fibonacci(num):  
 results.append("Yes\n")  
 else:  
 results.append("No\n")  
 return results  
  
 def \_is\_fibonacci(self, num):  
 *"""Проверяет, является ли число Фибоначчи."""* x1 = 5 \* (num\*\*2) + 4  
 x2 = 5 \* (num\*\*2) - 4  
 return self.\_is\_perfect\_square(x1) or self.\_is\_perfect\_square(x2)  
  
 def \_is\_perfect\_square(self, x):  
 if x < 0:  
 return False  
 left, right = 0, x  
 while left <= right:  
 mid = (left + right) // 2  
 square = mid \* mid  
 if square == x:  
 return True  
 elif square < x:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return False

**Текстовое объяснение решения.**

**Для решения задачи используется математическая характеристика чисел Фибоначчи: число является числом Фибоначчи, если одно из выражений** 5\*n^2 + 4 **или** 5\*n^2 - 4 **является полным квадратом. Это позволяет зафиксировать числовую принадлежность к последовательности Фибоначчи без необходимости генерировать сами числа. Внутри решающего класса проверяется каждое число на соответствие этой формуле.**

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.01134 сек | 1.67 МБ |
| Пример из задачи | 0.02689 сек | 1.77 МБ |
| Пример из задачи | 0.07581 сек | 1.87 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 1.02345 сек | 1.90 МБ |

**Вывод по задаче.**  
Решение использует эффективную математическую проверку, что позволяет быстро определить, является ли число числом Фибоначчи. Это решение работает за постоянное время для каждого числа и эффективно решает задачу.

## Задача 8:

# Текст задачи.

# 

# Листинг кода.

*"""Модуль для вычисления хеш-задачи по заданным параметрам."""*class HashSolver:  
 *"""  
 Класс для решения задачи с модификацией значений при обнаружении X в хеш-таблице.  
 """* def solve\_hash(self, data):  
 *"""  
 Принимает данные в формате:  
 Первая строка: N, X, A, B  
 Вторая строка: AC, BC, AD, BD  
  
 :param data: Список списков чисел [[N,X,A,B],[AC,BC,AD,BD]]  
 :return: Строка "X A B" после всех операций.  
 """* N, X, A, B = data[0]  
 AC, BC, AD, BD = data[1]  
  
 hash\_table = set()  
  
 for \_ in range(N):  
 if X in hash\_table:  
 A = (A + AC) % (10 \*\* 3)  
 B = (B + BC) % (10 \*\* 15)  
 else:  
 hash\_table.add(X)  
 A = (A + AD) % (10 \*\* 3)  
 B = (B + BD) % (10 \*\* 15)  
  
 X = (X \* A + B) % (10 \*\* 15)  
  
 return f"{X} {A} {B}"

**Текстовое объяснение решения.**

**Для решения задачи используется хеш-таблица, которая представлена как набор (`set`). На каждой итерации проверяется наличие числа** X **в таблице. В зависимости от этого выполняется модификация значений** A **и** B**. Модификации осуществляются по заданным формулам, и результат выводится после всех операций.**

|  |  |  |
| --- | --- | --- |
|  | Время выполнения | Затраты памяти |
| Нижняя граница диапазона значений | 0.00782 сек | 0.48 МБ |
| входных данных из текста задачи |  |  |
| Пример из задачи | 0.06548 сек | 1.48 МБ |
| Пример из задачи | 0.25642 сек | 1.60 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 1.10294 сек | 1.91 МБ |

**Вывод по задаче.  
Решение корректно выполняет необходимые модификации и выводит результаты в формате, указанном в задаче. Использование хеш-таблицы позволяет эффективно проверять наличие элемента и быстро модифицировать другие значения, что делает решение достаточно производительным.**

# Вывод

Все решения в задачах демонстрируют использование базовых структур данных и алгоритмов, что позволяет эффективно решать поставленные задачи в рамках ограничений по времени и памяти.