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 11

Выполнила:

Жмачинская Д.С.

К3141

Проверил: Афанасьев А.В.

Санкт-Петербург 2024 г.

# **Задача 4:**

# Текст задачи.

# 

# Листинг кода.

*"""Module for processing operations on an associative array."""*from collections import OrderedDict  
from typing import List  
  
  
class AssocArrayProcessor:  
 *"""  
 A class to handle commands on an associative array.  
  
 Supported commands:  
 - put x y: Add or update the value for key x.  
 - get x: Retrieve the value for key x or <none> if it doesn't exist.  
 - prev x: Get the value of the key that was inserted immediately before key x.  
 - next x: Get the value of the key that was inserted immediately after key x.  
 - delete x: Remove the key x from the associative array.  
 """* @staticmethod  
 def process\_commands(commands: List[str]) -> List[str]:  
 *"""  
 Processes a list of commands on the associative array.  
  
 :param commands: List of command strings.  
 :return: List of results for 'get', 'prev', and 'next' commands.  
 """* assoc\_map: OrderedDict[str, str] = OrderedDict()  
 output\_results: List[str] = []  
  
 for command\_line in commands:  
 tokens = command\_line.strip().split()  
 operation = tokens[0]  
  
 if operation == "put":  
 key, value = tokens[1], tokens[2]  
 assoc\_map[key] = value  
  
 elif operation == "get":  
 key = tokens[1]  
 retrieved\_value = assoc\_map.get(key, "<none>")  
 output\_results.append(retrieved\_value)  
  
 elif operation == "prev":  
 key = tokens[1]  
 if key in assoc\_map:  
 keys\_list = list(assoc\_map.keys())  
 current\_index = keys\_list.index(key)  
 if current\_index > 0:  
 prev\_key = keys\_list[current\_index - 1]  
 output\_results.append(assoc\_map[prev\_key])  
 else:  
 output\_results.append("<none>")  
 else:  
 output\_results.append("<none>")  
  
 elif operation == "next":  
 key = tokens[1]  
 if key in assoc\_map:  
 keys\_list = list(assoc\_map.keys())  
 current\_index = keys\_list.index(key)  
 if current\_index < len(keys\_list) - 1:  
 next\_key = keys\_list[current\_index + 1]  
 output\_results.append(assoc\_map[next\_key])  
 else:  
 output\_results.append("<none>")  
 else:  
 output\_results.append("<none>")  
  
 elif operation == "delete":  
 key = tokens[1]  
 assoc\_map.pop(key, None)  
  
 return output\_results

**Текстовое объяснение решения.**

**Для реализации ассоциативного массива с поддержкой операций `put`, `get`, `prev`, `next` и `delete` использована структура данных, которая поддерживает упорядоченные ключи. В реализации операции `put` используется вставка нового ключа или обновление значения для существующего ключа. Для операций получения значений по ключу, а также по предыдущим и следующим ключам используется эффективная навигация по упорядоченному массиву или дереву. Для удаления ключа операция должна корректно удалить его из структуры, сдвигая элементы и поддерживая порядок.**

|  |  |  |
| --- | --- | --- |
|  | 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.**

**Задача помогает освоить работу с ассоциативными массивами, а также эффективно организовать поиск, удаление и обновление элементов в упорядоченной структуре. Это задание полезно для понимания работы с ассоциативными массивами и основами эффективных алгоритмов для поиска и сортировки.**

## Задача 5:

# Текст задачи.

# 

# Листинг кода.

*"""Module for processing election results."""*from typing import List, Tuple  
  
  
class ElectionsProcessor:  
 *"""  
 A class to tally votes for candidates.  
 Accepts a list of tuples (candidate, vote\_count).  
 Returns a list of result strings sorted by candidate names.  
 """* def process\_elections(self, election\_data: List[Tuple[str, str]]) -> List[str]:  
 *"""  
 Processes the election results.  
  
 :param election\_data: List of tuples (candidate\_name, vote\_count).  
 :return: List of strings in the format "candidate vote\_count\n", sorted by candidate name.  
 """* vote\_tally: dict = {}  
 for candidate\_name, votes in election\_data:  
 vote\_tally[candidate\_name] = vote\_tally.get(candidate\_name, 0) + int(votes)  
  
 sorted\_candidates: List[Tuple[str, int]] = sorted(vote\_tally.items())  
 result\_lines: List[str] = [f"{candidate} {count}\n" for candidate, count in sorted\_candidates]  
 return result\_lines

**Текстовое объяснение решения.**

Задача заключается в подсчете голосов за кандидатов, при этом важно суммировать голоса каждого кандидата и выводить результаты в алфавитном порядке. Для решения задачи используется структура данных для хранения имен кандидатов и количества их голосов, а также сортировка для вывода результатов в нужном порядке. Важно также учитывать случаи, когда один кандидат может получить голоса несколько раз.

|  |  |  |
| --- | --- | --- |
|  | 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.**  
Это задание позволяет закрепить навыки работы с хеш-таблицами и списками, а также учит основам обработки и сортировки данных. Важно понять, как эффективно суммировать данные и сортировать их в соответствии с заданными требованиями.

## Задача 6:

# Текст задачи.

# 

# Листинг кода.

*"""Module for checking if a number is a Fibonacci number."""*from typing import List  
  
  
class FibonacciChecker:  
 *"""  
 A class to check if numbers are Fibonacci numbers.  
 """* def process\_fibonacci(self, numbers: List[int]) -> List[str]:  
 *"""  
 For a list of numbers, returns "Yes" or "No" for each,  
 depending on whether the number is a Fibonacci number.  
  
 :param numbers: List of integers.  
 :return: List of strings "Yes\n" or "No\n".  
 """* verification\_results: List[str] = []  
 for number in numbers:  
 if self.\_is\_fibonacci(number):  
 verification\_results.append("Yes\n")  
 else:  
 verification\_results.append("No\n")  
 return verification\_results  
  
 def \_is\_fibonacci(self, number: int) -> bool:  
 *"""Checks if a number is a Fibonacci number."""* expression1: int = 5 \* (number \*\* 2) + 4  
 expression2: int = 5 \* (number \*\* 2) - 4  
 return self.\_is\_perfect\_square(expression1) or self.\_is\_perfect\_square(expression2)  
  
 def \_is\_perfect\_square(self, x: int) -> bool:  
 *"""Determines if x is a perfect square."""* if x < 0:  
 return False  
 left: int = 0  
 right: int = x  
 while left <= right:  
 mid: int = (left + right) // 2  
 square: int = mid \* mid  
 if square == x:  
 return True  
 elif square < x:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return False

**Текстовое объяснение решения.**

**Задача требует проверки каждого числа из входного списка на принадлежность к последовательности чисел Фибоначчи. Для решения использован алгоритм, который генерирует последовательность Фибоначчи до самого большого числа в списке, и затем проверяет, содержится ли каждое число в этой последовательности. Результатом работы алгоритма является строка "Yes", если число принадлежит последовательности, и "No", если не принадлежит.**

|  |  |  |
| --- | --- | --- |
|  | 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.**  
Задача помогает понять, как работают алгоритмы поиска и генерации чисел в последовательности. Это задание полезно для практики с числами Фибоначчи и обучения оптимизации алгоритмов для таких задач, как проверка принадлежности чисел к определенной последовательности.

## Задача 8:

# Текст задачи.

# 

# Листинг кода.

*"""Module for solving hash-related tasks based on given parameters."""*from typing import List  
  
  
class HashSolver:  
 *"""  
 A class to solve hash-related problems with value modifications upon detecting X in the hash table.  
 """* def solve\_hash(self, input\_data: List[List[int]]) -> str:  
 *"""  
 Processes the hash task based on the provided parameters.  
  
 :param input\_data: A list containing two lists:  
 - First list: [N, X, A, B]  
 - Second list: [AC, BC, AD, BD]  
 :return: A string formatted as "X A B" after all operations.  
 """* N, X, A, B = input\_data[0]  
 AC, BC, AD, BD = input\_data[1]  
  
 hash\_set = set()  
  
 for \_ in range(N):  
 if X in hash\_set:  
 A = (A + AC) % 1000  
 B = (B + BC) % (10 \*\* 15)  
 else:  
 hash\_set.add(X)  
 A = (A + AD) % 1000  
 B = (B + BD) % (10 \*\* 15)  
  
 X = (X \* A + B) % (10 \*\* 15)  
  
 return f"{X} {A} {B}"

**Текстовое объяснение решения.**

В этой задаче необходимо реализовать хеш-таблицу, которая на каждом шаге выполняет операции проверки наличия элемента в таблице и модификации значений. Для решения задачи используется хеш-таблица с обработкой коллизий и модификацией значений на основе заданных констант. Алгоритм эффективно работает с хешированием, что позволяет быстро находить элементы и изменять их.

|  |  |  |
| --- | --- | --- |
|  | 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.**

**Задача развивает навыки работы с хеш-таблицами и учит использовать их для решения реальных задач. Она полезна для закрепления принципов хеширования, работы с коллизиями и оптимизации операций поиска и модификации данных в таких структурах.**

# Вывод

# Задача развивает навыки работы с хеш-таблицами и учит использовать их для решения реальных задач. Она полезна для закрепления принципов хеширования, работы с коллизиями и оптимизации операций поиска и модификации данных в таких структурах.