

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

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

Вариант 21

Выполнила:  
Савченко А.С.

Санкт-Петербург  
2024 г.

## **Содержание отчета**

<b>Содержание отчета</b>	<b>2</b>
<b>Задачи по варианту</b>	<b>3</b>
<b>ВЫЧИСЛЕНИЕ ВАРИАНТА</b>	<b>3</b>
Задача №1. Множество	5
Задача №6. Выборы в США	10
Задача №8. Почти интерактивная хеш-таблица	12
<b>Вывод (по всей лабораторной)</b>	<b>16</b>

## Задачи по варианту

## ВЫЧИСЛЕНИЕ ВАРИАНТА

Текст задачи:

**Базовый уровень.** Решается 3 задачи, причем должно выполняться два условия:

- Все три задачи не могут стоять *подряд* в задании. Т.е. решать набор (1,2,3) - нельзя, (1,2,4) - можно
- Номер одной из задач находится по следующей хэш-функции:

$$H(v) = (A \cdot v \bmod p) \bmod 9,$$

где  $A$  - последние 2 числа номера вашей группы,  $v$  - ваш номер в списке группы (вариант),  $p$  - следующее простое число, которое больше чем количество человек в вашей группе.

- (опционально) Номер второй задачи можете найти по принципу:

$$H(v) = ((A \cdot v + B) \bmod p) \bmod 9,$$

где  $B$  - это сумма кодов ASCII всех букв вашей фамилии ☺

В этом случае максимум за защиту можно получить 4 балла. В сумме с самой работой (0,5 балла) и отчетом (1 балл) получается 5,5 балла, что достаточно для зачета.

Листинг кода:

```
def var_generator(group_num, group_size, surname, student_number):
    A = int(str(group_num)[-2:]) # Последние 2 цифры номера группы

def next_prime(n):
    def is_prime(num):
        if num < 2:
            return False
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                return False
        return True
```

```

    prime = n + 1
    while not is_prime(prime):
        prime += 1
    return prime

    p = next_prime(group_size)    # Следующее простое
число большее, чем количество человек в группе

    # Сумма кодов ASCII всех букв фамилии
    B = sum(ord(char) for char in surname)

    task1 = (A * student_number % p) % 9
    task2 = ((A * student_number + B) % p) % 9

    # Проверка на подряд идущие задачи
    if task1 == task2 or task1 + 1 == task2:
        task2 = (task2 + 2) % 9

    task3 = (task1 + 2) % 9
    if task3 == task2 or task3 + 1 == task2 or task3 -
1 == task2 or task3 + 1 == task1 or task3 - 1 ==
task1:
        task3 = (task3 + 2) % 9

    return task1, task2, task3

group_num = 3241
group_size = 35
surname = "Савченко"
student_number = 25

task1, task2, task3 = var_generator(group_num,
group_size, surname, student_number)
print(f"Номер первой задачи: {task1}")

```

```
print(f"Номер второй задачи: {task2}")  
print(f"Номер третьей задачи: {task3}")
```

ВАРИАНТ:

```
C:\Users\Anastasia\PycharmProjects\1sem_labs\.venv  
Номер первой задачи: 8  
Номер второй задачи: 6  
Номер третьей задачи: 1
```

### **Задача №1. Множество**

Текст задачи:

## 1 задача. Множество

Реализуйте множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

- **Формат входного файла (input.txt).** В первой строке входного файла находится строго положительное целое число операций  $N$ , не превышающее  $5 \cdot 10^5$ . В каждой из последующих  $N$  строк находится одна из следующих операций:
  - $A\ x$  – добавить элемент  $x$  в множество. Если элемент уже есть в множестве, то ничего делать не надо.
  - $D\ x$  – удалить элемент  $x$ . Если элемента  $x$  нет, то ничего делать не надо.
  - $?\ x$  – если ключ  $x$  есть в множестве, выведите «Y», если нет, то выведите «N».

Аргументы указанных выше операций – **целые числа**, не превышающие по модулю  $10^{18}$ .

- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
8	Y
A 2	N
A 5	N
A 3	
? 2	
? 4	
A 2	
D 2	
? 2	

- Примечание.

Эту задачу можно решить совершенно разными способами, включая использование различных средств стандартных библиотек (правда, не всех - в стандартных библиотеках некоторых языков программирования используются слишком предсказуемые методы хеширования). Именно по этой причине все разумно использовать для проверки реализаций хеш-таблиц, которые понадобятся в следующих задачах этой работы.

Листинг кода:

```
class HashTable:
    def __init__(self):
        self.table = set()
```

```
def add_key(self, key):
    self.table.add(key)

def remove_key(self, key):
    self.table.discard(key)

def check_key(self, key):
    return key in self.table

def my_set(operations):
    hash_table = HashTable()
    results = []

    for operation in operations:
        splited = operation.split()
        op_type = splited[0]
        key = int(splited[1])

        if op_type == 'A':
            hash_table.add_key(key)
        elif op_type == 'D':
            hash_table.remove_key(key)
        elif op_type == '?':
            if hash_table.check_key(key):
                results.append('Y')
            else:
                results.append('N')
    return results

if __name__ == "__main__":
    with open('input.txt', 'r') as input_file:
        n = int(input_file.readline().strip())
        operations = [input_file.readline().strip() for
_ in range(n)]
    ans = my_set(operations)
```

```
with open('output.txt', 'w') as output_file:
    output_file.write("\n".join(ans) + "\n")
```

Текстовое объяснение решения: Из входного файла считываем список строк операции и число-ключ. В функцию передаем считанный массив строк, создаем пустой результирующий список. Для каждой операции из списка отделаем тип операции от числового ключа, в зависимости от типа операции либо добавляем число в множество, либо удаляем методом discard, если тип операции ? проверяем наличие элемента в множестве и помещаем да или нет в результирующий список, который потом выводим в выходной файл в нужном формате.

Результат работы кода на примерах из текста задачи: (скрины input output файлов):

inp	output.txt
1 8 ✓	1 Y
2 A 2	2 N
3 A 5	3 N
4 A 3	
5 ? 2	
6 ? 4	
7 A 2	
8 D 2	
9 ? 2	

Результат работы кода на минимальных и максимальных значениях: (скрины input output файлов):



input.txt

output.txt

The file size (11,19 ...

308102 A 4773773700 ✓
308103 A 4101362451846
308104 D 5530830270276
308105 D 8652406240330
308106 A 5457768157988
308107 ? -532766113274
308108 A 9676395720309
308109 D 4142448909173
308110 A -178751280282
308111 D -497917333323
308112 A -841278595252
308113 ? -151047938389
308114 D -301858851412
308115 A -203787800439
308116 ? -964900696587
308117 ? 2141910576110
308118 D 3023256142203
308119 D -364185898528

100287 N
100288 N
100289 N
100290 N
100291 N
100292 N
100293 N
100294 N
100295 N
100296 N
100297 N
100298 N
100299 N
100300 N
100301 N
100302 N
100303 N
100304 N
100305 N
100306 N

inp

output.txt

1 1 ✓
2 ? 2

1 N

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0005755 с	14.8359375 Мб
Пример из задачи	0.0009368 с	14.73046875 Мб
Верхняя граница диапазона значений входных данных из текста задачи	0.3517592 с	17.94921875 Мб

Вывод по задаче:

В данной задаче я реализовала множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа», используя встроенного типа данных `set()` в Python. Операции добавления, удаления и проверки существования выполняются в среднем за  $O(1)$  благодаря хеш-таблицам, используемым в `set`.

## Задача №6. Выборы в США

Текст задачи:

Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов — число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов. Вам известно за кого проголосовал каждый штат и сколько голосов было отдано данным штатом. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов.

- **Формат ввода / входного файла (input.txt).** Каждая строка входного файла содержит фамилию кандидата, за которого отдают голоса выборщики этого штата, затем через пробел идет количество выборщиков, отдавших голоса за этого кандидата.
- **Формат вывода / выходного файла (output.txt).** Выведите фамилии всех кандидатов в *лексикографическом* порядке, затем, через пробел, количество отданных за них голосов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.
- Примеры:

№	input.txt	output.txt
1	McCain 10	McCain 16
	McCain 5	Obama 17
	Obama 9	
	Obama 8	
	McCain 1	

№	input.txt	output.txt
2	ivanov 100	ivanov 900
	ivanov 500	petr 70
	ivanov 300	tourist 3
	petr 70	
	tourist 1	
	tourist 2	

№	input.txt	output.txt
3	bur 1	bur 1

Листинг кода:

```
cands = {}  
with open('input.txt', 'r') as input_file:  
    for line in input_file:
```

```

    name, votes = line.split()
    votes = int(votes)
    if name in cands:
        cands[name] += votes
    else:
        cands[name] = votes
sorted_cands = sorted(cands.items())
with open('output.txt', 'w') as output_file:
    for name, votes in sorted_cands:
        output_file.write(f"{name} {votes}\n")

```

Текстовое объяснение решения:

Создаем слов где будем хранить голоса за каждого из кандидатов. Чистаем в цикле строки из файла и разделяем их на имя кандидата и кол-во голосов. Если имя уже есть в словаре по ключу добавляем голоса, в противном случае добавляем имя и кол-во голосов. Сортируем кандидатов в лексикографическом порядке, вывод в файл отсортированный список кортежей в нужном формате.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):

input.txt	output.txt
McCain 10	1 McCain 16
McCain 5	2 Obama 17
Obama 9	3
Obama 8	
McCain 1	

input.txt	output.txt
1 ivanov 100	1 ivanov 900
2 ivanov 500	2 petr 70
3 ivanov 300	3 tourist 3
4 petr 70	4
5 tourist 1	
6 tourist 2	

Результат работы кода на минимальных и максимальных значениях:(скрины input output файлов):

	Время выполнения	Затраты памяти
Пример из задачи	0.0006989 с	14.9921875 Мб
Пример из задачи	0.0008013 с	14.85546875 Мб
Пример из задачи	0.000775 с	14.9765625 Мб

### **Задача №8. Почти интерактивная хеш-таблица**

Текст задачи:

## 8 задача. Почти интерактивная хеш-таблица

В данной задаче у Вас не будет проблем ни с вводом, ни с выводом. Просто реализуйте быструю хеш-таблицу.

В этой хеш-таблице будут храниться целые числа из диапазона  $[0; 10^{15} - 1]$ . Требуется поддерживать добавление числа  $x$  и проверку того, есть ли в таблице число  $x$ . Числа, с которыми будет работать таблица, генерируются следующим образом. Пусть имеется четыре целых числа  $N, X, A, B$  такие что:

- $1 \leq N \leq 10^7$
- $1 \leq X \leq 10^{15}$
- $1 \leq A \leq 10^3$
- $1 \leq B \leq 10^{15}$

Требуется  $N$  раз выполнить следующую последовательность операций:

- Если  $X$  содержится в таблице, то установить  $A \leftarrow (A + A_C) \bmod 10^3, B \leftarrow (B + B_C) \bmod 10^{15}$ .
- Если  $X$  не содержится в таблице, то добавить  $X$  в таблицу и установить  $A \leftarrow (A + A_D) \bmod 10^3, B \leftarrow (B + B_D) \bmod 10^{15}$ .
- Установить  $X \leftarrow (X \cdot A + B) \bmod 10^{15}$ .

Начальные значения  $X, A$  и  $B$ , а также  $N, A_C, B_C, A_D$  и  $B_D$  даны во входном файле. Выведите значения  $X, A$  и  $B$  после окончания работы.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится четыре целых числа  $N, X, A, B$ . Во второй строке содержится еще четыре целых числа  $A_C, B_C, A_D$  и  $B_D$  такие что  $0 \leq A_C, A_D < 10^3$   $0 \leq B_C, B_D < 10^{15}$ .
- **Формат выходного файла (output.txt).** Выведите значения  $X, A$  и  $B$  после окончания работы.

- Ограничение по времени. 5 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt
4 0 0 0
1 1 0 0
output.txt
3 1 1

Листинг кода:

```
from numba import njit
```

```

@njit
def hash_table(N, X, A, B, AC, BC, AD, BD):
    table = set()
    for _ in range(N):
        if X in table:
            A = (A + AC) % 1000
            B = (B + BC) % 10 ** 15
        else:
            table.add(X)
            A = (A + AD) % 1000
            B = (B + BD) % 10 ** 15
        X = (X * A + B) % 10 ** 15
    return X, A, B

if __name__ == "__main__":
    with open('input.txt', 'r') as infile:
        N, X, A, B = map(int,
infile.readline().split())
        AC, BC, AD, BD = map(int,
infile.readline().split())

    X, A, B = hash_table(N, X, A, B, AC, BC, AD, BD)

    with open('output.txt', 'w') as outfile:
        outfile.write(f"{X} {A} {B}")

```

Текстовое объяснение решения:

Будем использовать библиотеку `numba` чтобы уложиться в ограничения времени. `@njit` - Декоратор, который оптимизирует и ускоряет функцию с помощью JIT-компиляции. Сама функция `hash_table`:  $N$  раз выполняет операции описанные в задаче.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):

```
input.txt x v : output.txt
4 0 0 0 ✓ 1 3 1 1
1 1 0 0 2
```

Результат работы кода на минимальных и максимальных значениях:(скрины input output файлов):

```
input.txt x v : output.txt x
1 1 1 1 ✓ 1 2 1 1
0 0 0 0 2
```

```
time_test.py x input.txt x
1 10000000 9999999999999999 9999999999999999 9999999999
2 999 9999999999999999 999 9999999999999999
3
output.txt x
1 9999899999999999 999 9999999899999999
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0007799 с	15.0234375 Мб
Пример из задачи	0.0005965 с	14.9453125 Мб
Верхняя граница диапазона значений входных данных из текста задачи(без numba)	7.4637536 с	16.76953125 Мб
Верхняя граница диапазона значений входных данных из	3.026858 с	91.53125 Мб

текста задачи(с numba)		
------------------------	--	--

Вывод по задаче:

Сначала я реализовала алгоритм без сторонних библиотек, но не укладывался в ограничения времени(7.4637536 с), а это значило что либо стоит использовать другой ЯП или сторонние библиотеки, такой является numba. С ней код укладывается в ограничения, хоть и требует больше памяти.

**Вывод** (по всей лабораторной)

При выполнении лабораторной работы я поработала с хэш-таблицами, множествами, и узнала новую библиотеку numba. Самой интересной задачей оказалось вычисление задач варианта)