

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

Отчет по лабораторной работе №5
по курсу «Алгоритмы и структуры данных»
Тема: Деревья. Пирамида, пирамидальная сортировка.
Очередь с приоритетами.

Вариант 11

Выполнил:
Кузнецов А.Г.
К3140

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задача №1 Множество.	3
Задача №2 Телефонная книга.	5
Задача №4 Прошитый ассоциативный массив.	7
Задача №6 Фибоначчи возвращается.	9
Вывод	11

Задачи по варианту

Задача №1 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 1018.

- 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

```
1 def add(num):
2     if not (num in set):
3         set.append(num)
4     else:
5         pass
6
7
8 def remove(num):
9     if num in set:
10        set.remove(num)
11    else:
12        pass
13
14
15 def contains(num):
16     if num in set:
17         with open('output.txt', 'a') as f:
18             f.write("Y" + "\n")
19     else:
20         with open('output.txt', 'a') as f:
21             f.write("N" + "\n")
22
```

```

23
24 with open('input.txt', 'r') as f:
25     n = int(f.readline())
26     commands = f.read().splitlines()
27     set = []
28     if 1 <= n <= 5 * 10 ** 5:
29         for command in commands:
30             cmd = command.split(' ')[0]
31             num = int(command.split(' ')[1])
32             if abs(num) <= 10 ** 18:
33                 if cmd == "A":
34                     add(num)
35
36                 elif cmd == "D":
37                     remove(num)
38
39                 else:
40                     contains(num)
41             else:
42                 with open('output.txt', 'w') as f:
43                     f.write("ERROR")
44                 break

```

Из файла input.txt получаем количество команд и сами команды, а также создаем массив для хранения команд и множество для хранения будущих значений. Проверяем удовлетворяют ли значения условиям, если да, то приступаем к чтению команд. При команде “A” добавляем значение в множество, если элемента x нет в нём. При команде “D” удаляем значение в множестве, если элемент x есть в множестве. При команде “?” проверяем есть ли элемент x в множестве, если да, то выводим Y, в противном случае выводим N. Выведенные ответы записываем в файл output.txt

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

Вывод по задаче: в ходе выполнения первой задачи было реализовано множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

Задача №2 Телефонная книга.

В этой задаче ваша цель - реализовать простой менеджер телефонной книги. Он должен уметь обрабатывать следующие типы пользовательских запросов:

- `add number name` – это команда означает, что пользователь добавляет в телефонную книгу человека с именем `name` и номером телефона `number`. Если пользователь с таким номером уже существует, то ваш менеджер должен перезаписать соответствующее имя.
- `del number` – означает, что менеджер должен удалить человека с номером из телефонной книги. Если такого человека нет, то он должен просто игнорировать запрос.
- `find number` – означает, что пользователь ищет человека с номером телефона `number`. Менеджер должен ответить соответствующим именем или строкой «not found» (без кавычек), если такого человека в книге нет.
- Формат ввода / входного файла (`input.txt`). В первой строке входного файла содержится число N ($1 \leq N \leq 105$) - количество запросов. Далее следуют N строк, каждая из которых содержит один запрос в формате, описанном выше.

Все номера телефонов состоят из десятичных цифр, в них нет нулей в начале номера, и каждый состоит не более чем из 7 цифр. Все имена представляют собой непустые строки из латинских букв, каждая из которых имеет длину не более 15. Гарантируется при проверке, что не будет человека с именем «not found».

- Формат вывода / выходного файла (`output.txt`). Выведите результат каждого поискового запроса `find` – имя, соответствующее номеру телефона, или «not found» (без кавычек), если в телефонной книге нет человека с таким номером телефона. Выведите по одному результату в каждой строке в том же порядке, как были заданы запросы типа `find` во входных данных.
- Ограничение по времени. 6 сек.
- Ограничение по памяти. 512 мб.

```
1 def add(number, name):
2     book[number] = name
3
4
5 def delete(number):
6     try:
7         return book.pop(number)
8
9     except KeyError:
```

```

10         return None
11
12
13 def find(number):
14     if book.get(number) is not None:
15         return book.get(number)
16     else:
17         return "not found"
18
19
20 with open('input.txt', 'r') as f:
21     n = int(f.readline())
22     commands = f.read().splitlines()
23     book = {}
24 if 1 <= n <= 10 ** 5:
25     for command in commands:
26         cmd = command.split(' ')[0]
27         number = int(command.split(' ')[1])
28         if cmd == "add":
29             name = command.split(' ')[2]
30             add(number, name)
31
32         elif cmd == "del":
33             delete(number)
34
35         else:
36             with open('output.txt', 'a') as f:
37                 f.write(find(number) + '\n')

```

На вход получаем из файла input.txt количество команд и сами команды, а также создаем словарь для работы с операциями. Проверяем удовлетворяет ли количество команд условию, если да, то приступаем к выполнению задания. Команда add добавляет в словарь ключ (номер телефона) и значение (имя), если же ключ повторяется, то значение переписывается. Команда del удаляет элементы по ключу. Команда find выводит значение по ключу. Ответы команды find записываем в файл output.txt

input.txt	output.txt
1 12	1 Mom
2 add 911 police	2 not found
3 add 76213 Mom	3 police
4 add 17239 Bob	4 not found
5 find 76213	5 Mom
6 find 910	6 daddy
7 find 911	7
8 del 910	
9 del 911	
10 find 911	
11 find 76213	
12 add 76213 daddy	
13 find 76213	
14	

Вывод по задаче: в ходе выполнения второй задачи была реализована простая телефонная книга, которая с помощью словаря.

Задача №4 Прошитый ассоциативный массив.

Реализуйте прошитый ассоциативный массив. Ваш алгоритм должен поддерживать следующие типы операций:

- `get x` – если ключ `x` есть в множестве, выведите соответствующее ему значение, если нет, то выведите `<none>`.
- `prev x` – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до `x`, или `<none>`, если такого нет или в массиве нет `x`.
- `next x` – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после `x`, или `<none>`, если такого нет или в массиве нет `x`.
- `put x y` – поставить в соответствие ключу `x` значение `y`. При этом следует учесть, что
 - если, независимо от предыстории, этого ключа на момент вставки в массиве не было, то он считается только что вставленным и оказывается самым последним среди добавленных элементов – то есть, вызов `next` с этим же ключом сразу после выполнения текущей операции `put` должен вернуть `<none>`;
 - если этот ключ уже есть в массиве, то значение необходимо изменить, и в этом случае ключ не считается вставленным еще раз, то есть, не меняет своего положения в порядке добавленных элементов.
- `delete x` – удалить ключ `x`. Если ключа в ассоциативном массиве нет, то ничего делать не надо.
- 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 приведенных выше операций. Ключи и значения операций - строки из латинских букв длиной не менее одного и не более 20 символов.
- Формат выходного файла (`output.txt`). Выведите последовательно результат выполнения всех операций `get`, `prev`, `next`. Следуйте формату выходного файла из примера.
- Ограничение по времени. 4 сек.
- Ограничение по памяти. 256 мб.

```

1 def put(key, data):
2     a[key] = data
3
4
5 def get(key):
6     if a.get(key) is not None:
7         return a.get(key)
8     else:
9         return "<none>"
10
11
12 def delete(number):
13     try:
14         return a.pop(number)
15     except KeyError:
16         return None
17
18
19 def prev(key):
20     keys = list(a.keys())
21     for k in keys:
22         if k == key:
23             if keys.index(k) != 0:
24                 prev = keys[keys.index(k) - 1]
25                 return a.get(prev)
26             else:
27                 return "<none>"
28
29
30 def next(key):
31     keys = list(a.keys())
32     for k in keys:
33         if k == key:
34             if keys.index(k) != (len(keys) - 1):
35                 next = keys[keys.index(k) + 1]
36                 return a.get(next)
37             else:
38                 return "<none>"
39
40
41 with open('input.txt', 'r') as f:
42     n = int(f.readline())
43     commands = f.read().splitlines()
44     a = {}
45     for command in commands:
46         cmd = command.split(' ')[0]
47         key = command.split(' ')[1]
48         if cmd == "put":
49             data = command.split(' ')[2]
50             put(key, data)
51
52         elif cmd == "delete":
53             delete(key)
54
55         elif cmd == "get":

```

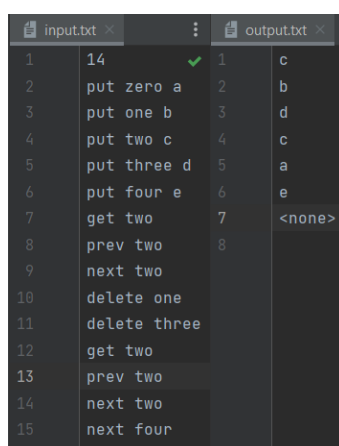


```

56         with open('output.txt', 'a') as f:
57             f.write(get(key) + '\n')
58
59     elif cmd == "prev":
60         with open('output.txt', 'a') as f:
61             f.write(prev(key) + '\n')
62
63     else:
64         with open('output.txt', 'a') as f:
65             f.write(next(key) + '\n')

```

На вход из файла input.txt получаем количество команд и сами команды, а также создаём словарь для хранения данных. Команды выполняются следующим образом: put добавляет в словарь ключ и значение в словарь при повторении ключа значение будет переписано, get выводит значение из словаря основываясь на ключ, delete удаляет элемент по ключу, prev и next рассматривают набор ключей и в зависимости от запроса выведут предыдущее значение или следующее. Ответ записываем в файл output.txt



input.txt	output.txt
1 14	1 c
2 put zero a	2 b
3 put one b	3 d
4 put two c	4 c
5 put three d	5 a
6 put four e	6 e
7 get two	7 <none>
8 prev two	8
9 next two	
10 delete one	
11 delete three	
12 get two	
13 prev two	
14 next two	
15 next four	

Вывод по задаче: в ходе выполнения четвертой задачи был реализован прошитый ассоциативный массив, который поддерживает операции put, delete, get, prev и next

Задача №6 Фибоначчи возвращается.

Вам дается последовательность чисел. Для каждого числа определите, является ли оно числом Фибоначчи. Напомним, что числа Фибоначчи определяются, например, так:

$$F_0 = F_1 = 1$$

$F_i = F_{i-1} + F_{i-2}$ для $i \geq 2$.

- Формат ввода / входного файла (input.txt). Первая строка содержит одно число N ($1 \leq N \leq 106$) - количество запросов. Следующие N строк содержат по одному целому числу. При этом соблюдаются следующие ограничения при проверке:

1. Размер каждого числа не превосходит 5000 цифр в десятичном представлении.

2. Размер входа не превышает 1 Мб.

- Формат вывода / выходного файла (output.txt). Для каждого числа, данного во входном файле, выведите «Yes», если оно является числом Фибоначчи, и «No» в противном случае.

- Ограничение по времени. 2 сек.

- Ограничение по памяти. 128 мб. Внимание: есть вероятность превышения по памяти, т.к. сами по себе числа Фибоначчи большие. Делайте проверку на память!

```
1 import math
2 t_start = time.perf_counter()
3 with open('input.txt', 'r') as f:
4     n = int(f.readline())
5     nums = f.read().splitlines()
6 for num in nums:
7     if math.sqrt(5 * (int(num) ** 2) - 4) % 1 == 0 or math.sqrt(5 *
8 (int(num) ** 2) + 4) % 1 == 0:
9         with open('output.txt', 'a') as f:
10             f.write("Yes" + '\n')
11     else:
12         with open('output.txt', 'a') as f:
13             f.write("No" + '\n')
```

Получаем на вход количество чисел и сами числа. Проверка на число Фибоначчи происходит с помощью формулы. Выходные данные записываем в output.txt

input.txt	output.txt
1 8	1 Yes
2 1	2 Yes
3 2	3 Yes
4 3	4 No
5 4	5 Yes
6 5	6 No
7 6	7 No
8 7	8 Yes
9 8	9

Вывод по задаче: в ходе выполнения шестой задачи был реализован алгоритм по вычислению является ли число числом Фибоначчи или же нет

Вывод

В ходе лабораторной работы была проведена работа с множествами, словарями хэш-таблицами и хэш-функциями.