

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

Отчет по лабораторной работе №4
по курсу «Алгоритмы и структуры данных»
Тема: Стек, очередь, связанный список
Вариант 7

Выполнила:
Пожидаева Е.Р.

Санкт-Петербург
2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту.....	3
Задание №1. Стек [N баллов]	3
Задание №4. Скобочная последовательность [N баллов]	6
Задание №5. Стек с максимумом [N баллов]	10
Задание №7. Максимум в движущейся последовательности [N баллов]	14
Вывод по лабораторной работе №4.....	16

Задачи по варианту

Задание №1. Стек [N баллов]

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо “+ N”, либо “-”. Команда “+ N” означает добавление в стек числа N , по модулю не превышающего 10^9 . Команда “-” означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 10^6 элементов.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится M ($1 \leq M \leq 10^6$) – число команд. Каждая последующая строка исходного файла содержит ровно одну команду.
- **Формат выходного файла (output.txt).** Выведите числа, которые удаляются из стека с помощью команды “-”, по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
6	10
+ 1	1234
+ 10	
-	
+ 2	
+ 1234	
-	

```
def stack_operations(commands):
    stack = []
    deleted_elements = []

    for command in commands:
        command = command.strip()
        if command == '-':
            deleted_elements.append(stack.pop())
        else:
```

```

        stack.append(int(command.split()[1]))

    return deleted_elements

# считываем число команд
with open('input.txt', 'r') as file:
    m = int(file.readline())

# считываем команды
with open('input.txt', 'r') as file:
    commands = file.readlines()[1:]

deleted_elements = stack_operations(commands)

# выводим изъятые элементы
with open('output.txt', 'w') as file:
    for element in deleted_elements:
        file.write(str(element) + '\n')

```

Текстовое объяснение решения.

Данный код реализует функцию `stack_operations`, которая выполняет операции со стеком на основе переданных команд.

Стек представлен в виде списка `stack`, а переменная `deleted_elements` будет хранить значения удаленных элементов.

Цикл `for` перебирает каждую команду из списка `commands`. Команда осуществляет операции - (удаление элемента из стека) или добавление элемента в стек. Если команда начинается с `-`, то с помощью `pop` извлекается последний элемент из стека и добавляется в список `deleted_elements`. В остальных случаях команда разделяется на две части: `'-'` и число, которое нужно добавить в стек. Цикл завершается после выполнения всех команд.

Для проверки корректности работы функции, в начале считывается число команд `m`, а затем команды. Значения удаленных элементов записываются в файл `'output.txt'`.


 first.py


 four.py

```
1 6
2 + 1
3 + 10
4 -
5 + 2
6 + 1234
7 -|
```

 first.py

 four.py

 five.py

 7

```
1 10
2 1234|
3
```

	Время выполнения, с	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.012844085693359375	6328320
Пример из задачи	0.000341892242431640 6	6127616

Верхняя граница диапазона значений входных данных из текста задачи(1000000 элементов)	0.03581047058105469	6438208
---	---------------------	---------

Задание №4. Скобочная последовательность [N баллов]

Определение правильной скобочной последовательности такое же, как и в задаче 3, но теперь у нас больше набор скобок: `[]{}()`.

Нужно написать функцию для проверки наличия ошибок при использовании разных типов скобок в текстовом редакторе типа LaTeX.

Для удобства, текстовый редактор должен не только информировать о наличии ошибки в использовании скобок, но также указать точное место в коде (тексте) с ошибочной скобочкой.

В первую очередь объявляется ошибка при наличии первой несовпадающей закрывающей скобки, перед которой отсутствует открывающая скобка, или которая не соответствует открывающей, например, `()[]` - здесь ошибка укажет на `}`.

Во вторую очередь, если описанной выше ошибки не было найдено, нужно указать на первую несовпадающую открывающую скобку, у которой отсутствует закрывающая, например, `([[`.

Если не найдено ни одной из указанных выше ошибок, нужно сообщить, что использование скобок корректно.

Помимо скобок, код может содержать большие и маленькие латинские буквы, цифры и знаки препинания.

Формально, все скобки в коде (тексте) должны быть разделены на пары совпадающих скобок, так что в каждой паре открывающая скобка идет перед закрывающей скобкой, а для любых двух пар скобок одна из них вложена внутри другой, как в `(foo[bar])` или они разделены, как в `f(a,b)-g[c]`. Скобка `[` соответствует скобке `]`, соответствует `(` и `(` соответствует `)`.

```
def check_brackets(string):
    stack = []      # Используется стек для отслеживания
открытых скобок

    for i, char in enumerate(string, start=1):
        if char in '([{': # Если символ - открывающая
скобка
            stack.append((char, i))
        elif char in ')]}': # Если символ - закрывающая
скобка
            if not stack: # Если стек пустой
                return i
            top_char, _ = stack.pop() # Извлекаем
последнюю открывающую скобку из стека
            if (char == ')' and top_char != '(') or \
                (char == ']' and top_char != '[') or \
                (char == '}' and top_char != '{'):
```

```

        return i

    if stack: # Если в стеке остались открытые скобки
        _, index = stack.pop()
        return index

    return -1 # Если все скобки сбалансированы

# Чтение скобок из входного файла
with open("input.txt", "r") as file:
    expression = file.read()

# Проверка правильности расстановки скобок
result = check_brackets(expression)

# Запись результата в выходной файл
with open("output.txt", "w") as file:
    if result == -1:
        file.write("Success.")
    else:
        file.write(str(result))

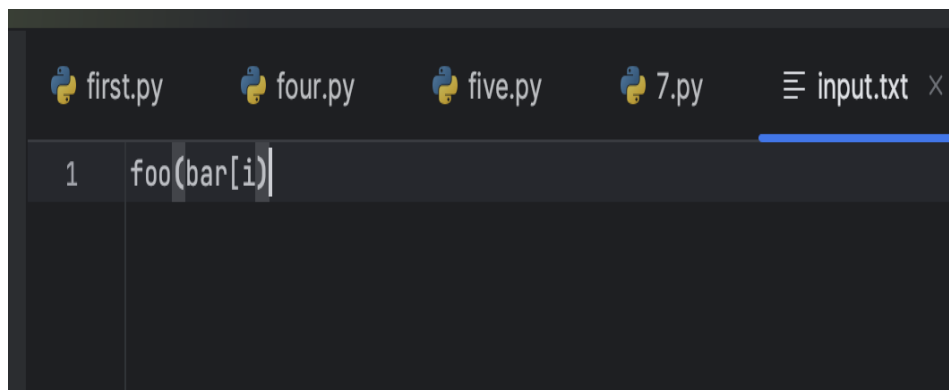
```

Текстовое объяснение решения.

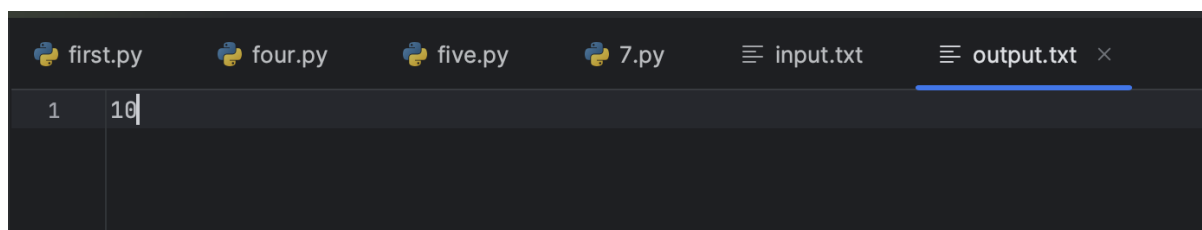
Функция `check_brackets` принимает строку `string` в качестве аргумента и проверяет правильность расстановки скобок в этой строке. Создается пустой стек `stack`, который будет использоваться для отслеживания открытых скобок. В цикле проходит каждый символ `char` из строки `string`, начиная с 1-й позиции (параметр `start=1`). Если `char` является открывающей скобкой ('(', '[', '{'), то она добавляется в стек вместе с его индексом (`i`) в формате `(char, i)`. Если `char` является закрывающей скобкой (')', ']', '}'), то выполняется следующая логика: Если стек пустой, значит скобки неправильно расставлены, поэтому функция возвращает текущий индекс `i`. Извлекается последняя открывающая скобка из стека и сохраняется в переменную `top_char`. Индекс не используется (`_`). Если `char` является закрывающей скобкой, но `top_char` не соответствует соответствующей

открывающей скобке, значит скобки неправильно расставлены и функция возвращает текущий индекс `i`. После прохода по всем символам строки, проверяется, остались ли открытые скобки в стеке. Если да, то извлекается последний элемент стека и его индекс возвращается. Если стек пустой после проверки всех скобок, значит все скобки сбалансированы и функция возвращает `-1`. Чтение строки со скобками из входного файла `"input.txt"`. Вызов функции `check_brackets` с аргументом `expression` и сохранение результата в переменную `result`. Запись результата в выходной файл `"output.txt"`. Если результат равен `-1`, то записывается строка `"Success."`, иначе записывается результат преобразованный в строку.

Результат работы кода на примере из текста задачи:



```
1 foo(bar[i])
```



```
1 10
```

Выводы по задаче: Я научилась проверять символы с помощью стека.

Задание №5. Стек с максимумом [N баллов]

Стек - это абстрактный тип данных, поддерживающий операции `Push()` и `Pop()`. Нетрудно реализовать его таким образом, чтобы обе эти операции работали за константное время. В этой задаче ваша цель - реализовать стек, который также поддерживает поиск максимального значения и гарантирует, что все операции по-прежнему работают за константное время.

Реализуйте стек, поддерживающий операции `Push()`, `Pop()` и `Max()`.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится n ($1 \leq n \leq 400000$) – число команд. Последующие n строк исходного файла содержит ровно одну команду: `push V`, `pop` или `max`. $0 \leq V \leq 10^5$.
- **Формат выходного файла (output.txt).** Для каждого запроса `max` выведите (в отдельной строке) максимальное значение стека.

Листинг кода.

```
class MaxStack:
    def __init__(self):
        self.stack = [] # основной стек
        self.max_stack = [] # стек максимальных значений

    def push(self, value):
        self.stack.append(value)
        if not self.max_stack or value >= self.max_stack[-1]:
            self.max_stack.append(value)

    def pop(self):
        value = self.stack.pop()
        if value == self.max_stack[-1]:
            self.max_stack.pop()

    def max(self):
        return self.max_stack[-1]
```

```
with open('input.txt', 'r') as fr:
    n = int(fr.readline())
    commands = [fr.readline().strip().split() for _
in range(n)]

stack = MaxStack()
result = []
for command in commands:
    if command[0] == 'push':
        stack.push(int(command[1]))
    elif command[0] == 'pop':
        stack.pop()
    elif command[0] == 'max':
        result.append(str(stack.max()))

with open('output.txt', 'w') as fw:
    fw.write('\n'.join(result))
```

Текстовое объяснение решения.

В основной части кода открывается файл ввода 'input.txt', считывается количество команд *n* и сами команды. Затем создается объект класса *MaxStack* и выполняются команды из списка команд. Если команда - 'push', то вызывается метод *push()* объекта класса, если команда - 'pop', то вызывается метод *pop()*, а если команда - 'max', то вызывается метод *max()*. Результаты вызова метода *max()* добавляются в список *result*.

После обработки всех команд список *result* записывается в файл вывода 'output.txt' в формате строк, разделенных символом новой строки.

Результат работы кода на примере из текста задачи:

```
first.py four.py five.py 7.py input.txt x output.txt
1 5
2 push 2
3 push 1
4 max
5 pop
6 max
```

```
first.py four.py five.py 7.py input.txt output.txt x
1 2
2 2
```

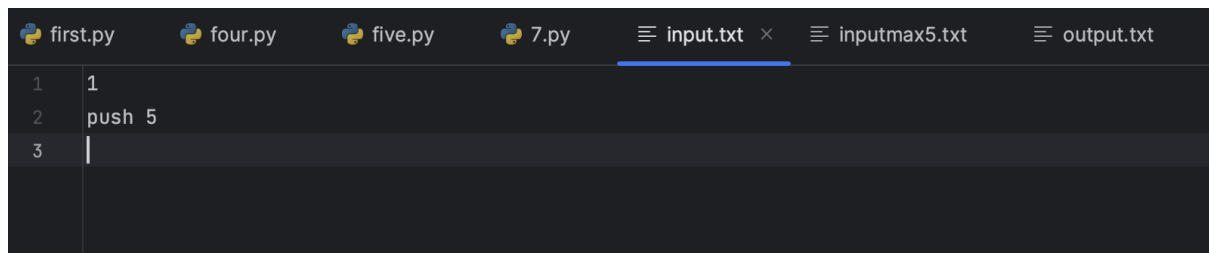
Результат работы кода на максимальных и минимальных значениях:

Максимальное:

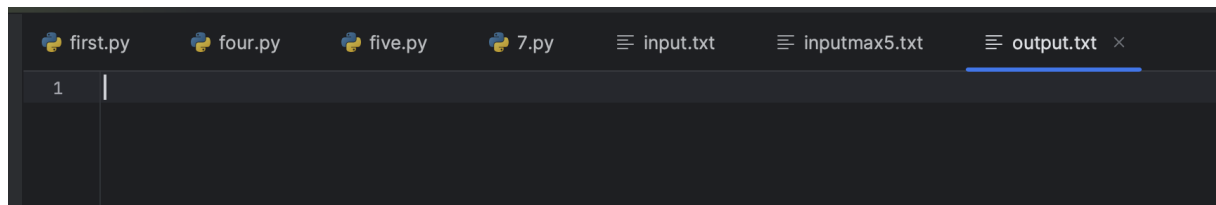
```
first.py four.py five.py 7.py input.txt inputmax5.txt x output.txt
1 400000
2 push 7
3 push 1
4 push 7
5 max
6 pop
7 max
8 push 7
9 max
10 pop
11 max
12 push 7
```

```
first.py four.py five.py 7.py input.txt inputmax5.txt output.txt x
1 7
2 7
3 7
4 7
5 7
6 7
7 7
8 7
9 7
10 7
11 7
```

Минимальное:



```
first.py four.py five.py 7.py input.txt x inputmax5.txt output.txt
1 1
2 push 5
3 |
```



```
first.py four.py five.py 7.py input.txt inputmax5.txt output.txt x
1 |
```

	Время выполнения, с	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0015120506286621094	6164480
Пример из задачи	0.0006091594696044922	6262784
Верхняя граница диапазона значений входных данных из текста задачи	0.6375131607055664	6057984

Вывод по задаче: я научилась работать с методами и стэком

Задание №7. Максимум в движущейся последовательности [N баллов]

Задан массив из n целых чисел - a_1, \dots, a_n и число $m < n$, нужно найти максимум среди последовательности ("окна") $\{a_i, \dots, a_{i+m-1}\}$ для каждого значения $1 \leq i \leq n - m + 1$. Простой алгоритм решения этой задачи за $O(nm)$ сканирует каждое "окно" отдельно.

Ваша цель - алгоритм за $O(n)$.

- **Формат входного файла (input.txt).** В первой строке содержится целое число n ($1 \leq n \leq 10^5$) – количество чисел в исходном массиве, вторая строка содержит n целых чисел a_1, \dots, a_n этого массива, разделенных пробелом ($0 \leq a_i \leq 10^5$). В третьей строке - целое число m - ширина "окна" ($1 \leq m \leq n$).
- **Формат выходного файла (output.txt).** Нужно вывести $\max a_i, \dots, a_{i+m-1}$ для каждого $1 \leq i \leq n - m + 1$.

Листинг кода.

```
with open("input.txt", "r") as input_file:
    n = int(input_file.readline())
    numbers = list(map(int,
input_file.readline().split()))
    m = int(input_file.readline())

with open("output.txt", "w") as output_file:
    left = 0
    right = m - 1
    max_window = max(numbers[left:right + 1])
    output_file.write(str(max_window) + " ")

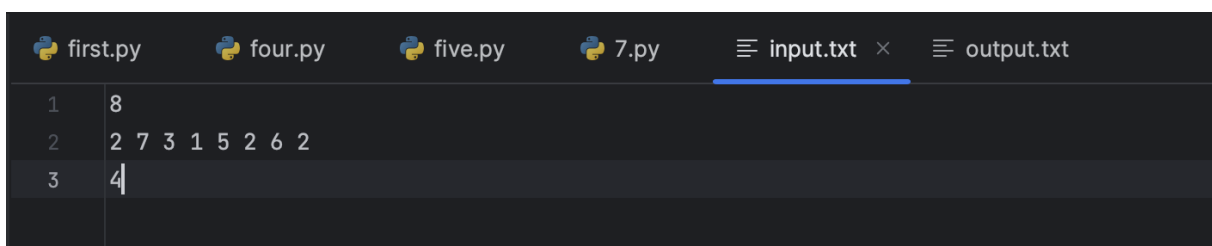
    while right < n - 1:
        left += 1
        right += 1

        if numbers[right] > max_window:
            max_window = numbers[right]
        elif numbers[left - 1] == max_window:
            max_window = max(numbers[left:right + 1])
        output_file.write(str(max_window) + " ")
```

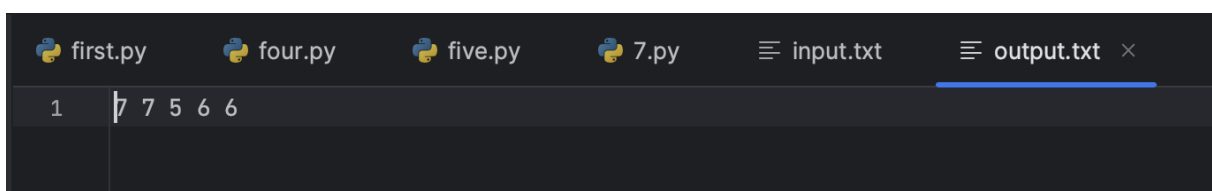
Текстовое объяснение решения.

Код начинается с открытия файла "input.txt" для чтения. Затем он считывает первую строку, которая содержит число "n", обозначающее количество чисел в последовательности. Далее он считывает вторую строку, которая содержит сами числа, разделенные пробелами. Эти числа сохраняются в переменной "numbers" в виде списка целых чисел. Затем код считывает третью строку, содержащую число "m", которое обозначает размер окна. Затем код открывает файл "output.txt" для записи. Далее он инициализирует переменные "left" и "right" со значениями 0 и "m-1" соответственно. Затем переменная "max_window" инициализируется значением максимального числа в текущем окне, полученного путем вызова функции "max()" на списке чисел срезом от "left" до "right + 1". Затем максимальное число записывается в файл "output.txt". Затем запускается цикл while с условием "right < n - 1", что означает, что окно не дошло до конца последовательности чисел. Внутри цикла проверяется, является ли следующее число больше текущего максимального числа. Если это так, то изменяется значение переменной "max_window". Если же число перед текущим окном равно текущему максимальному числу, то опять вызывается функция "max()" на списке чисел срезом от "left" до "right + 1". Текущее максимальное число записывается в файл "output.txt". Затем файлы закрываются. Код записывает результат в файл "output.txt", где на каждой строке записывается максимальное число в текущем окне. В этом решении использована очередь.

Результат работы кода на примере из текста задачи:



```
first.py four.py five.py 7.py input.txt x output.txt
1 8
2 2 7 3 1 5 2 6 2
3 4
```



```
first.py four.py five.py 7.py input.txt output.txt x
1 7 7 5 6 6
```

Вывод по задаче: я научилась работать с методами и стэком

Вывод по лабораторной работе №4:

В ходе данной лабораторной работы я узнала о стеках.