# 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.

Выполнил: Криличевский М. Е.

Номер группы: К3139

Проверил: Афанасьев А. В.

Санкт-Петербург

2024 г.

# Содержание

Задание 1	3
Задание 3	6
Задание 5	9
Задание 7	12
Задание 9	15
Вывод	

#### 1 задача. Стек

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.

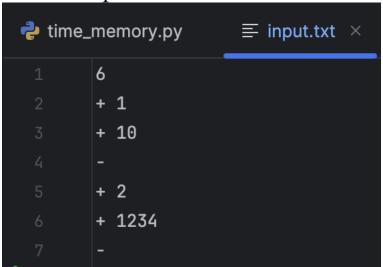
- Формат входного файла (input.txt). В первой строке входного файла содержится M ( $1 \le M \le 10^6$ ) число команд. Каждая последующая строка исходного файла содержит ровно одну команду.
- Формат выходного файла (output.txt). Выведите числа, которые удаляются из стека с помощью команды "—", по одному 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	
-	

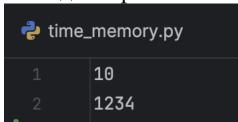
```
1. from lab_4.utils import read_input, write_output,
  check, string_column
2.
3.
4.def steck(a):
5.
       m = []
       for i in range(len(a)-1):
    if a[i+1]=="-" and len(a[i+1])==1:
6.
7.
                 s = "".join(x for x in a[i] if
  x.isdecimal())
9.
                 m.append(s)
10.
            return m
11.
       if __name__=="__main__":
    FILE_INPUT = "../txtf/input.txt"
    FILE_OUTPUT = "../txtf/output.txt"
12.
13.
14.
15.
            list = read_input(FILE_INPUT)
16.
            check = check(FILE_INPUT)
            if check:
17.
                 result = string_column(steck(list))
18.
                 write_output(FILE_OUTPUT, result)
19.
20.
                 print("Входные данные корректны")
21.
            if not(check):
                 write_output(FILE_OUTPUT, "ОШИБКА
22.
  ВХОДНЫХ ДАННЫХ")
                 raise ValueError("ОШИБКА ВХОДНЫХ
23.
  ДАННЫХ")
```

#### Объяснение решения:

Этот код анализирует список строк, извлечённый из файла, и возвращает последовательность чисел, находящихся перед строками, содержащими один символ "-". Если входные данные корректны, программа обрабатывает их по заданным правилам и записывает результат в выходной файл. В случае ошибки данных выводится сообщение об ошибке.



#### Выходной файл:



```
/usr/local/bin/python3.12 /Users/max/Pycharm
ВРЕМЯ: 0.00036733399610966444
Память: 9.6 МБ
Размер списка: 88 байт
Process finished with exit code 0
```

#### 3 задача. Скобочная последовательность. Версия 1

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем *правильной скобочной последовательностью*, если выполняется одно из следующих утверждений:

- А пустая последовательность;
- первый символ последовательности A это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как A=(B)C, где B и C правильные скобочные последовательности;
- первый символ последовательности A это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как A=(B)C, где B и C правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не приметел

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

• Формат входного файла (input.txt). Первая строка входного файла содержит число N ( $1 \le N \le 500$ ) — число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до  $10^4$  включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

3

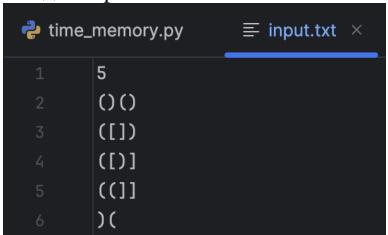
- Формат выходного файла (output.txt). Для каждой строки входного файла (кроме первой, в которой записано число таких строк) выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
5	YES
00	YES
([])	NO
(D)	NO
((]]	NO
)(	

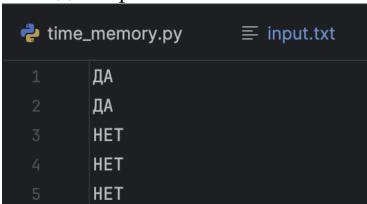
```
1. from lab_4.utils import read_input, write_output, check_three,
   string_column
  def bracket_sequence(a):
       results = []
5.
6.
       for s in a:
7.
            stack = []
            is_valid = True
8.
9.
                  for char in s:
10.
                       if char in "([":
11.
                       stack.append(char)
elif char in ")]":
   if not stack:
13.
                                is_valid = False
15.
                           top = stack.pop()
  if (top == '(' and char != ')') or (top
== '[' and char != ']'):
18.
19.
                                is_valid = False
20.
21.
                                break
22.
                  if stack:
23.
                       is_valid = False
25.
                  results.append("ДА" if is_valid else "HET")
26.
27.
             return string_column(results)
28.
29.
         if ___name___=="__main___":
30.
             FILE_INPUT = "../txtf/input.txt"
FILE_OUTPUT = "../txtf/output.txt"
31.
32.
33.
             list = read_input(FILE_INPUT)
34.
             check = check_three(FILE_INPUT)
35.
             if check:
                  write_output(FILE_OUTPUT, bracket_sequence(list))
36.
37.
                  print("Входные данные корректны")
38.
                  raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
39.
```

#### Объяснение решения:

Этот код проверяет корректность последовательностей скобок в списке строк, считанном из входного файла. Если последовательность скобок правильная (каждая открывающая скобка закрывается соответствующей закрывающей), результат для данной строки — "ДА". Если последовательность некорректная, результат — "НЕТ". Все результаты выводятся в виде столбца. Программа сохраняет результаты в выходной файл, если входные данные корректны.



## Выходной файл:



```
/usr/local/bin/python3.12 /Users/max/Pycharml
BPEMЯ: 0.00032295800338033587
Память: 9.7 МБ
Размер списка: 94 байт
Process finished with exit code 0
```

#### 5 задача. Стек с максимумом

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 выходного файла (output.txt). Для каждого запроса тах выведите (в отдельной строке) максимальное значение стека.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input.txt	output.txt	input.txt	output.txt	input.txt	output.txt
5	2	5	2	3	
push 2	2	push 1	1	push 1	
push 1		push 2		push 7	
max		max		pop	
pop		pop			
max		max			

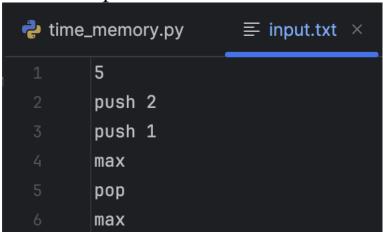
5

input.txt	output.txt	input.txt	output.txt
10	9	6	7
push 2	9	push 7	7
push 3	9	push 1	
push 9	9	push 7	
push 7		max	
push 2		pop	
max		max	
max			
max			
pop			
max			

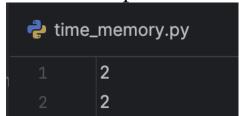
```
1. from lab_4.utils import read_input, check_five, write_output,
   string_column
3. def process_stack(commands):
       stack = []
5.
       max_stack = []
       results = []
6.
7.
8.
       for command in commands:
            if command.startswith("push"):
9.
                       _, value = command.split()
10.
                      value = int(value)
11.
                      stack.append(value)
12.
13.
                      if max_stack:
14.
                          max_stack.append(max(value, max_stack[-
15.
   1]))
16.
                      else:
17.
                          max_stack.append(value)
18.
                  elif command.startswith("pop"):
19.
                      if stack:
20.
                           stack.pop()
                 max_stack.pop()
elif command.startswith("max"):
23.
                      if max_stack:
25.
                           results.append(str(max_stack[-1]))
26.
27.
             return results
28.
29.
         if ___name__=="__main___":
             FILE_INPUT = "../txtf/input.txt"
FILE_OUTPUT = "../txtf/output.txt"
30.
31.
32.
             list = read_input(FILE_INPUT)
33.
             check = check_five(FILE_INPUT)
34.
             if check:
35.
                  result = process_stack(list)
                 write_output(FILE_OUTPUT, string_column(result))
36.
                 print("Входные данные корректны")
37.
             else:
38.
                 write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
39.
  ДАННЫХ")
40.
                  raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

#### Объяснение решения:

Этот код реализует обработку команд работы со стеком, включая команды добавления числа, удаления верхнего элемента и получения максимального элемента в текущем состоянии стека. Помимо основного стека, используется дополнительный стек для отслеживания максимального значения, что позволяет выполнять команду 'max'. Если входные данные корректны, результат записывается в выходной файл. При некорректных входных данных выводится сообщение об ошибке.



## Выходной файл:



```
/usr/local/bin/python3.12 /Users/max/Pycharml
ВРЕМЯ: 0.0003386669995961711
Память: 9.5 МБ
Размер списка: 88 байт
Process finished with exit code 0
```

#### 7 задача. Максимум в движущейся последовательности

Задан массив из n целых чисел -  $a_1,...,a_n$  и число m < n, нужно найти максимум среди последовательности ("окна")  $\{a_i,...,a_{i+m-1}\}$  для каждого значения  $1 \le i \le n-m+1$ . 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 ( $1 \le n \le 10^5$ ) количество чисел в исходном массиве, вторая строка содержит n целых чисел  $a_1, ..., a_n$  этого массива, разделенных пробелом ( $0 \le a_i \le 10^5$ ). В третьей строке целое число m ширина "окна" ( $1 \le m \le n$ ).
- Формат выходного файла (output.txt). Нужно вывести  $\max a_i,...,a_{i+m-1}$  для каждого  $1 \le i \le n-m+1$ .
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример:

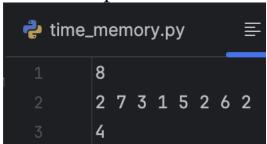
input.txt	output.txt
8	77566
27315262	
4	

- Есть несколько решений этой задачи. Например:
  - использование очереди на основе двух стеков;
  - использование Dequeue.

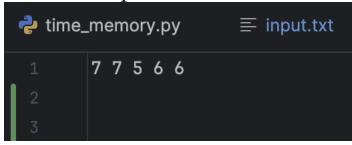
```
1. from collections import deque
2. from lab_4.utils import string, read_input_seven, check_seven,
   write_output
4. def max_dynamic_sequence(n,a,m):
5.
        decart_queue = deque()
6.
        result = []
7.
8.
        for i in range(n):
9.
             if decart_queue and decart_queue[0] < i - m + 1:</pre>
10.
                         decart_queue.popleft()
11.
12.
                   while decart_queue and a[decart_queue[-1]] <=</pre>
   a[i]:
13.
                         decart_queue.pop()
14.
15.
                    decart_queue.append(i)
16.
17.
                    if i >= m - 1:
                         result.append(a[decart_queue[0]])
18.
19.
20.
               return result
21.
         if __name__=="__main__":
    FILE_INPUT = "../txtf/input.txt"
    FILE_OUTPUT = "../txtf/output.txt"
22.
23.
24.
25.
               check = check_seven(FILE_INPUT)
26.
               if check:
                    argues = read_input_seven(FILE_INPUT)
27.
                   n, a, m = argues[0], argues[1], argues[2]
result = max_dynamic_sequence(n, a, m)
write_output(FILE_OUTPUT, string(result))
28.
29.
30.
31.
                    print("Входные данные корректны")
32.
33.
               else:
                   write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
   данных")
                    raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

#### Объяснение решения:

Этот код решает задачу поиска максимума в подмассивах фиксированной длины m для массива а длины n. Алгоритм эффективно использует двухстороннюю очередь ('deque') для поддержки скользящего окна, обеспечивая временную сложность. Результаты записываются в выходной файл, если входные данные корректны.



## Выходной файл:



#### Тестирование времени и памяти:

/usr/local/bin/python3.12 /Users/max/PycharmI ВРЕМЯ: 0.0003032500098925084 Память: 9.9 МБ Размер списка: 120 байт Process finished with exit code 0

#### 9 задача. Поликлиника

Очередь в поликлинике работает по сложным правилам. Обычные пациенты при посещении должны вставать в конец очереди. Пациенты, которым "только справку забрать встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром. Напишите программу, которая отслеживает порядок пациентов в очереди.

- Формат входного файла (input.txt). В первой строке записано одно целое число n (  $1 \le n \le 10^5$  ) число запросов к вашей программе. В следующих n строках заданы описания запросов в следующем формате:
  - «+ і» к очереди присоединяется пациент i ( $1 \le i \le N$ ) и встает в ее конец;
  - «\* і» пациент i встает в середину очереди (  $1 \leq i \leq N$ );
  - «-» первый пациент в очереди заходит к врачу. Гарантируется, что на момент каждого такого запроса очередь будет не пуста.

8

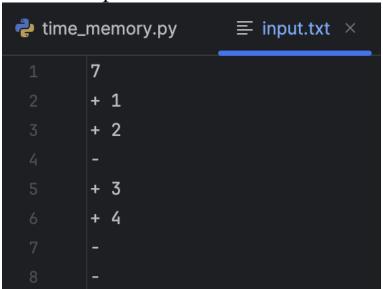
- Формат выходного файла (output.txt). Для каждого запроса третьего типа в отдельной строке выведите номер пациента, который должен зайти к праманам
- Ограничение по времени. Оцените время работы и используемую память при заданных максимальных значениях.
- Пример:

input.txt	output.txt	input.txt	output.txt
7	1	10	1
+ 1	2	+1	3
+ 1 + 2	3	+ 2	2
-		* 3	5
+ 3 + 4		-	4
+ 4		+ 4 * 5	
-		* 5	
-		-	
		-	
		-	
		-	

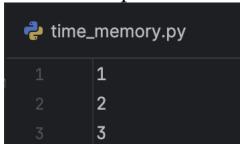
```
1. from collections import deque
2. from lab_4.utils import read_input, check_nine, write_output,
   string_column
4. def process_requests(n, requests):
        queue = deque()
6.
        results = []
7.
8.
        for request in requests:
    if request[0] == '+':
9.
10.
                        _, i = request.split()
11.
                   queue.append(i)
elif request[0] == '*':
12.
13.
14.
                        _, i = request.split()
15.
                        mid = (len(queue) + 1) // 2
16.
                   queue.insert(mid, i)
elif request[0] == '-':
17.
18.
19.
20.
                        results.append(queue.popleft())
21.
22.
              return results
23.
         if __name__=="__main__":
    FILE_INPUT = "../txtf/input.txt"
    FILE_OUTPUT = "../txtf/output.txt"
25.
              check = check_nine(FILE_INPUT)
27.
28.
              if check:
29.
                   list = read_input(FILE_INPUT)
30.
   write_output(FILE_OUTPUT, string_column(process_requests(list[1
   :],list)))
31.
                   print("Входные данные корректны")
32.
33.
              else:
                   write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
34.
   ДАННЫХ")
35.
                   raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

#### Объяснение решения:

Этот код реализует обработку последовательности запросов на двустороннюю очередь deque. Входные запросы могут добавлять элемент в конец, в середину или удалять первый элемент из очереди. После выполнения запросов выводится список результатов операций удаления. Если входные данные корректны, результат записывается в выходной файл, иначе программа выводит сообщение об ошибке.



## Выходной файл:



```
/usr/local/bin/python3.12 /Users/max/Pycharm
ВРЕМЯ: 0.00025029199605342
Память: 10.5 МБ
Размер списка: 88 байт
Process finished with exit code 0
```

## Вывод

В данной лабораторной работе познакомился и поработал с элементарными структурами данных: стеком, очередью и связанным списком.