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 11

Выполнил:
Тимаков Е.П. (фамилия имя)
К3141 (номер группы)

Проверила:
Артамонова В.Е.

Санкт-Петербург 2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Высота дерева

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 109 . Команда “–”означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 106 элементов.

• Формат входного файла (input.txt). В первой строке входного файла содержится M ($1 \le M \le 106$) – 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.

Листинг кода:

```
from collections import deque
file1=open("input.txt")
nums=[]
stack=deque()
removed=[]
while True:
    line = file1.readline()
    if not line:
        break
    nums.append((line.split()))
for i in range(len(nums)):
    if nums[i][0]=='+':
        stack.append(nums[i][1])
    elif nums[i][0] == '-':
        removed.append(str(stack[-1]))
        stack.pop()
```

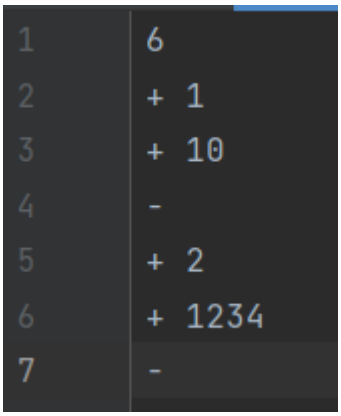
```
g = open('output.txt', 'w')
g.write(" ".join(str(i) for i in removed))
g.close()
```

Текстовое объяснение решения:

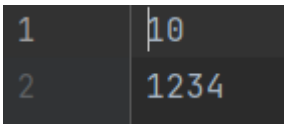
В данном коде мы реализуем основные функции такой структуры данных, как стек. В зависимости от типа операции мы проводим необходимые действия, в случае если это удаление, то мы добавляем этот элемент в removed.

Результат работы кода на примерах из текста задачи:

input.txt:



output.txt:

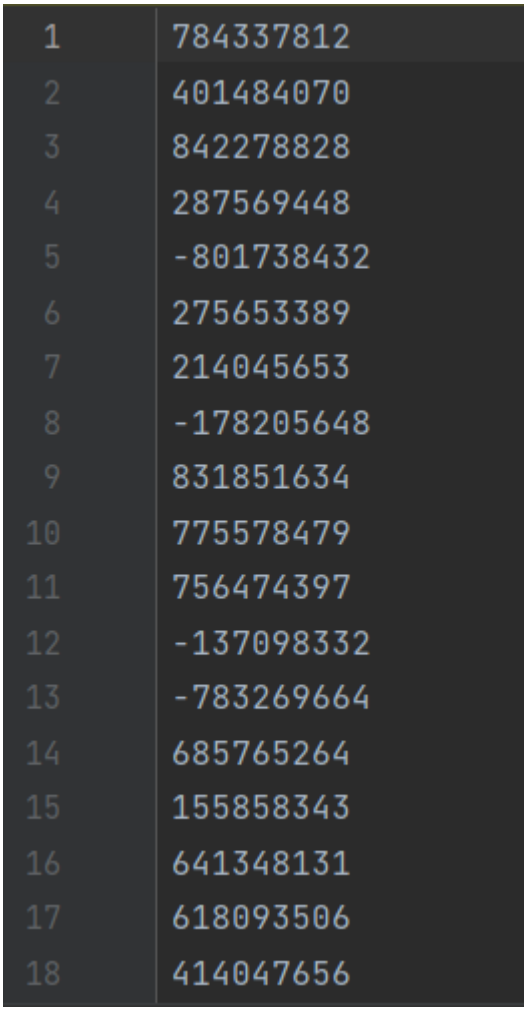


Результат работы кода на максимальных:

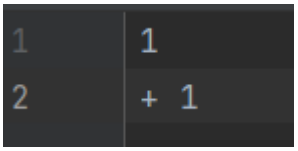
input.txt:

```
1000000
-
+ 784337812
-
-
+ 842278828
+ 401484070
-
-
-
-
+ -801738432
+ 287569448
-
-
+ 214045653
+ 275653389
-
```

output.txt:



Результат работы кода на минимальных значениях:
input.txt:



output.txt:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из	0.001	25

текста задачи		
Пример из задачи	0.001	25
Верхняя граница диапазона значений входных данных из текста задачи	0.923	220

Вывод по задаче:

В данной задаче мы изучили и реализовали такую структуру данных, как стек. Также изучили его основные функции.

Задача №4. Наибольшая возрастающая подпоследовательность

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, (в ([]. Если не найдено ни одной из указанный 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) .

• Формат входного файла (input.txt). Входные данные содержат одну строку S, состоящую из больших и маленьких латинских букв, цифр, знаков препинания и скобок из набора [] (). Длина строки S – 1 ≤ S leq105 .

• Формат выходного файла (output.txt). Если в строке S скобки используются правильно, выведите «Success» (без кавычек). В противном случае выведите отсчитываемый от 1 индекс первой несовпадающей закрывающей скобки, а если нет несовпадающих закрывающих скобок, выведите отсчитываемый от 1 индекс первой открывающей скобки, не имеющей закрывающей.

• Ограничение по времени. 5 сек.

• Ограничение по памяти. 256 мб.

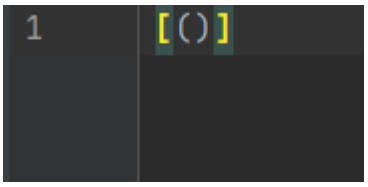
Листинг кода:

```
def check_brackets(sequence):
    stack = []
    brackets = {'(': ')', '[': ']', '{': '}'}

    for i, char in enumerate(sequence):
        if char in brackets:
            stack.append((char, i + 1))
        elif char in brackets.values():
            if not stack:
                return i + 1
            top, top_index = stack.pop()
            if brackets[top] != char:
                return i + 1

    if stack:
        return stack[0][1]

    return "Success"

with open("input.txt", "r") as file:
    sequence = file.read().strip()

result = check_brackets(sequence)

with open("output.txt", "w") as file:
    file.write(str(result))
```

Текстовое объяснение решения:

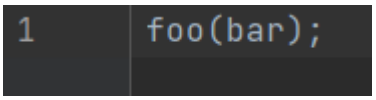
Для решения данной задачи мы используем структуру данных такую, как стек. Мы проходимся по строке, считая корректно открывающиеся и закрывающиеся скобки, в случае если строка закончалась, а элементы в стеке остались, то выводим длину строки.

Результат работы кода на примерах из текста задачи:
input.txt:

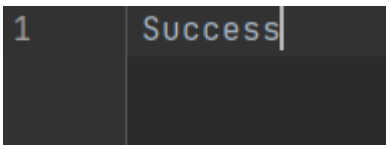


output.txt:

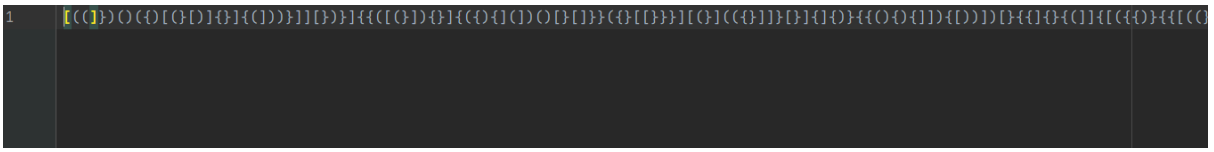
input.txt:



output.txt:



Результат работы кода на максимальных:
input.txt:



output.txt:

Результат работы кода на минимальных значениях:
input.txt:

output.txt:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001	25
Пример из задачи	0.001	25
Пример из задачи	0.001	25
Верхняя граница диапазона значений входных данных из текста задачи	0.005	25

Вывод по задаче:

В данной лабораторной работе мы реализовали на практической задаче возможности стека.

Задача №6. Очередь с минимумом.

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат. На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда – это либо «+ N», либо «–», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 109 . Команда «–» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди. • Формат входного файла (input.txt). В первой строке содержится M ($1 \leq M \leq 106$) – число команд. В последующих строках содержатся команды, по одной в каждой строке.

• Формат выходного файла (output.txt). Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть

выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

Листинг кода:

```
q = []

f = open('input.txt')
n = open('output.txt', 'w')
cmds = [el.replace('\n', '') for el in f.readlines()]
for cmd in cmds:
    if cmd[0] == '+':
        q.append(int(cmd[2:]))
    elif cmd == '-':
        del q[0]
    elif cmd == '?':
        m = 10**9+1
        for el in q:
            if el < m:
                m = el
        print(m, file=n)
```

Текстовое объяснение решения:

В данном коде мы реализуем основные функции такой структуры данных, как очередь. В данной задаче помимо минимальных команд очереди, добавляем поиск минимального элемента.

Результат работы кода на примерах из текста задачи:

input.txt:

```
7
+ 1
? 10
+ 10
?
-
?
-
```

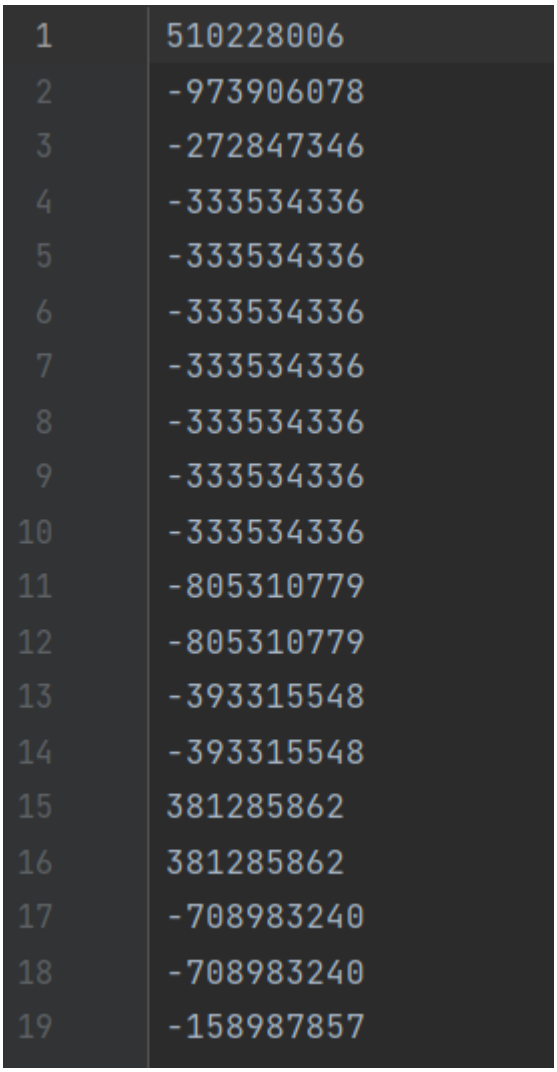
output.txt:

```
1
10
```

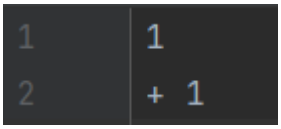
Результат работы кода на максимальных:
input.txt:

```
1000000
+ 190591141
+ 510228006
-
?
+ -973906078
+ -272847346
+ 834707150
+ 340067956
+ -54103461
?
-
-
+ 379872766
?
+ 967669417
+ -333534336
-
```

output.txt:



Результат работы кода на минимальных значениях:
input.txt:



output.txt:

	Время выполнения	Затраты памяти
Нижняя граница	0.001	25

диапазона значений входных данных из текста задачи		
Пример из задачи	0.001	25
Верхняя граница диапазона значений входных данных из текста задачи	1.5	95

Вывод по задаче:

В данной задаче мы изучили и реализовали такую структуру данных, как очередь. Также изучили его основные функции и реализовали дополнительную, такую как поиск минимального элемента.

Задача №11. Бюрократия

В министерстве бюрократии одно окно для приема граждан. Утром в очередь встают n человек, i-й посетитель хочет получить ai справок. За один прием можно получить только одну справку, поэтому если после приема посетителю нужны еще справки, он встает в конец очереди. За время приема министерство успевает выдать m справок. Остальным придется ждать следующего приемного дня. Ваша задача - сказать, сколько еще справок хочет получить каждый из оставшихся в очереди посетитель в тот момент, когда прием закончится. Если все к этому моменту разойдутся, выведите -1.

• Формат входного файла (input.txt). В первой строке - количество посетителей n ($1 \le n \le 105$) и количество справок m ($0 \le m \le 109$). Во второй строке для каждого посетителя в порядке очереди указано количество справок ai ($1 \le ai \le 106$), которое он рассчитывает получить. Номером посетителя называется его место в исходной очереди.

• Формат выходного файла (output.txt). В первой строке выведите, сколько посетителей останется в очереди, когда прием закончится. Во второй строке выведите состояние очереди на тот момент, когда прием закончится: для всех посетителей по порядку выведите по одному числу через пробел - количество справок, которое он хочет еще получить. В случае, если в очереди никого не останется выведите одно число: -1

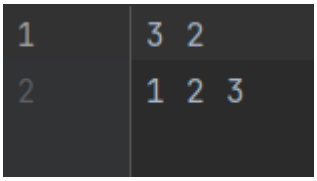
• Ограничение по времени. Оцените время работы и используемую память при заданных максимальных значениях.

Листинг кода:

```
from collections import deque
file1 = open("input.txt", "r")
nums=[]
stack=deque()
legit=0
while True:
    line = file1.readline()
    if not line:
        break
    nums.append((line.split()))
file1.close()
for i in range(len(nums[1])):
    stack.append(nums[1][i])
kolvo_spravok=nums[0][1]
for i in range(int(kolvo_spravok)):
    new_kolvo_spravok=int((stack.popleft()))-1
    if new_kolvo_spravok!=0:
        stack.append(new_kolvo_spravok)
    if len(stack)==0:
        legit=-1
if legit==-1:
    print(legit)
else:
    fileVar = open("output.txt", "w")
    fileVar.write(str(len(stack))+'\n')
    fileVar.write(str(stack))
    fileVar.close()
```

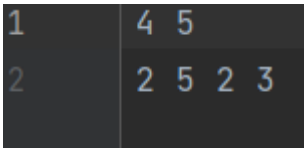
Текстовое объяснение решения:

В данном коде мы реализуем основные функции такой структуры данных, как очередь. Заполняем очередь. В случае если посетителю нужна еще одна справка, то добавляем его в заново в очередь. В случае если очередь пуста выводим -1.



output.txt:

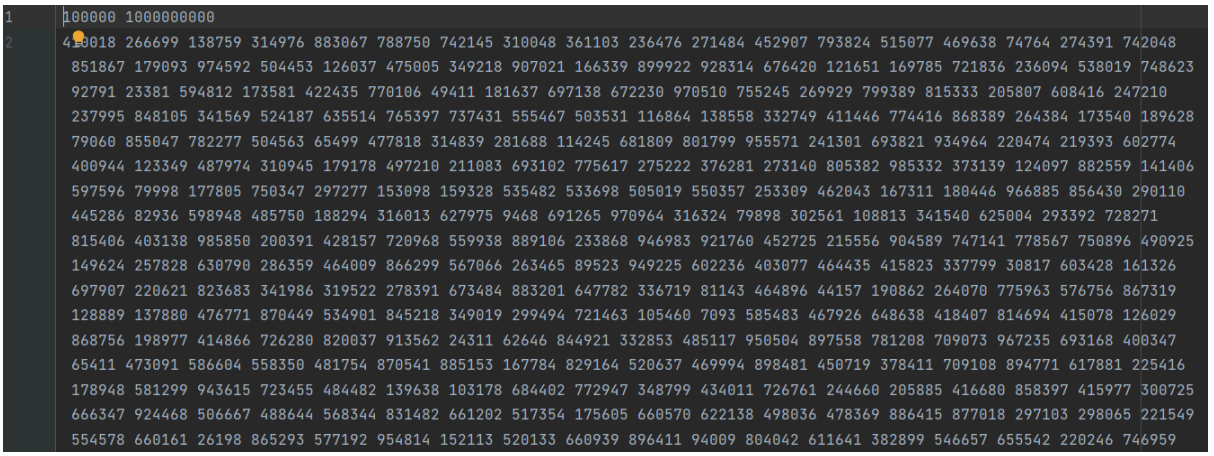
input.txt:



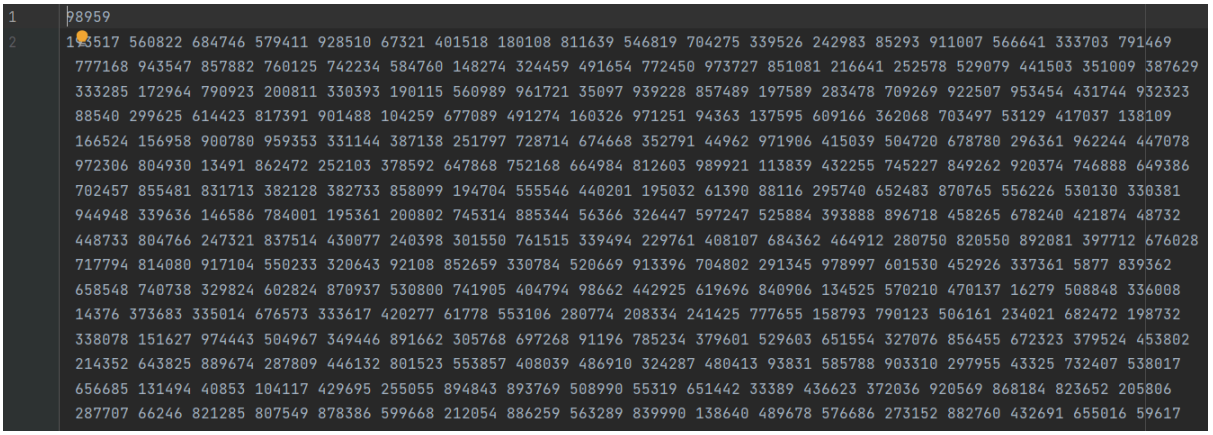
output.txt:

Результат работы кода на максимальных:

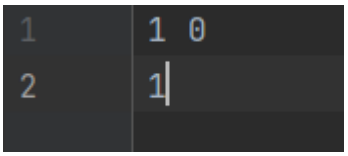
input.txt:



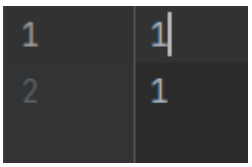
output.txt:



Результат работы кода на минимальных значениях:
input.txt:



output.txt:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001	25
Пример из задачи	0.001	25
Пример из задачи	0.001	25
Верхняя граница диапазона значений входных данных из текста задачи	4.20.524	36

Вывод по задаче:

В данной задаче мы изучили и реализовали такую структуру данных, как очередь. И применили ее в практической задаче.

Вывод:

В данной лабораторной работе мы изучили и реализовали такие структуры данных, как стек и очередь. Также применили их в практических задачах.