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 8

Выполнил:
Макунина А.А.
К32421

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №2. Очередь

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

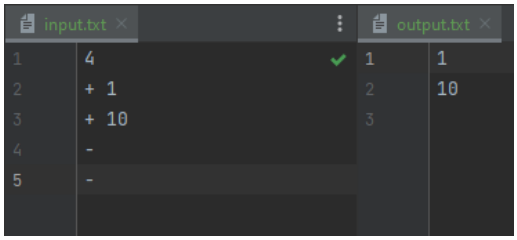
На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «–». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 10^9. Команда «–» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 10^6 элементов.

- **Формат входного файла (input.txt).** В первой строке содержится M ($1 \leq M \leq 10^6$) – число команд. В последующих строках содержатся команды, по одной в каждой строке.
- **Формат выходного файла (output.txt).** Выведите числа, которые удаляются из очереди с помощью команды «–», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

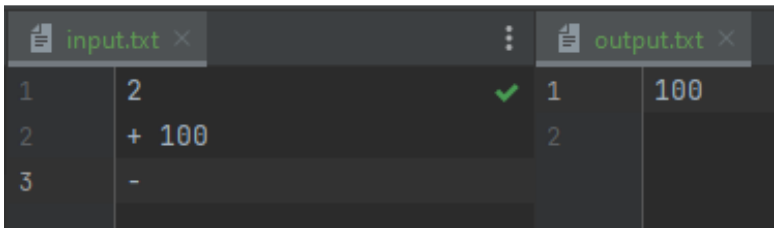
У нас есть пустой массив для очереди. Из первой строки файла считывается количество итераций. Далее в цикле for/in мы начинаем проводить данные итерации, считывая команды: «+» - добавляем элемент в очередь, «-» - изъятие методом pop() и запись в файл.

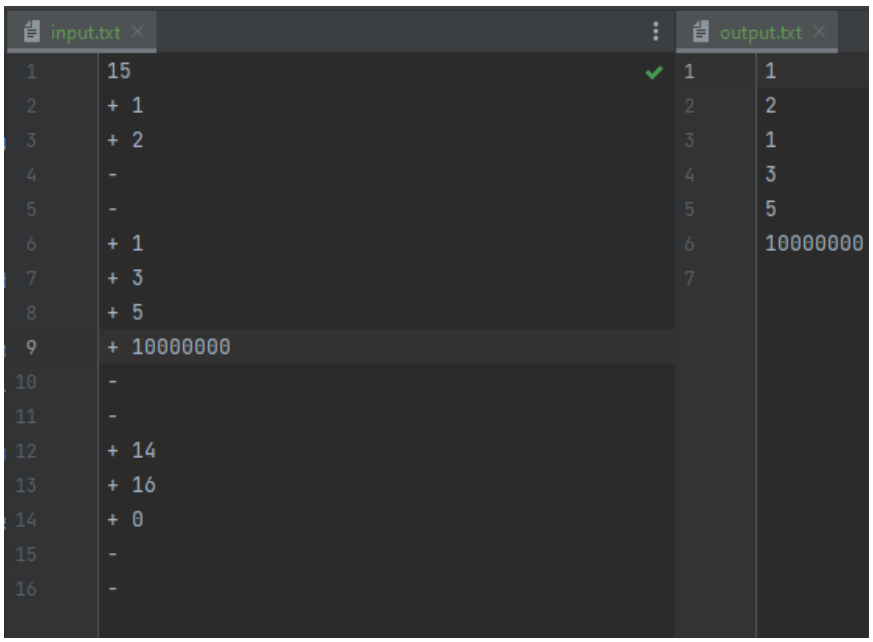
```
queue = []
f = open("input.txt")
iterations = int(f.readline())
w = open("output.txt", "w")
for i in range(iterations):
    line, *n = f.readline().split()  # обрежет строку с обоих концов
    if line == '+':
        queue.append(int(n[0]))
    else:
        w.write(str(queue.pop(0)) + '\n')
f.close()
w.close()
```

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:





	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0012886000000000009	0.026192665100097656
Пример из задачи	0.0023330000000000017	0.026153564453125
Верхняя граница диапазона значений входных данных из текста задачи	0.0034671999999999897	0.026462554931640625

Вывод по задаче: возникли проблемы с генерацией огромного файла из-за нарушения логики генерации, прошу простить, но сам код работает исправно!

Задача №4. Скобочная последовательность. Версия 2

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 $S - 1 \leq S\ leq 10^5$.

- **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.

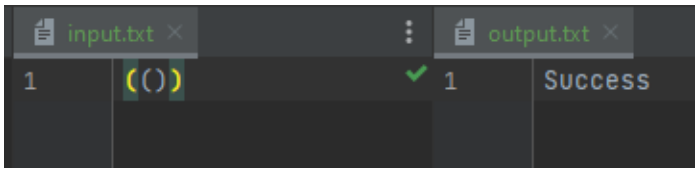
Получаем на вход строку. Создаём массив стэка, а также создаём словарь: с «ключами», то есть) : (,] : [, а } : {. Далее мы переходим к методу, где проверяем, что находится на входе, и если символ найден в словаре, то он отправляется в стек. Если скобка закрывается, то мы проверяем ключ в словаре (последний элемент в стеке). Если он совпадает, то мы удаляем последний элемент из стека, а если нет, то добавляем в стек скобку, потому что найденная ошибка должна быть правильно выведена. Чтобы сделать это, мы создаем цикл, чтобы проверить, есть ли у нас закрывающие скобки в стеке. В то же время мы обходим стек с конца, чтобы поймать первую закрывающую скобку. Если в стеке такового нет, мы выводим индекс первой скобки (уже очевидно, что она является открывающей, потому что закрывающих больше нет).

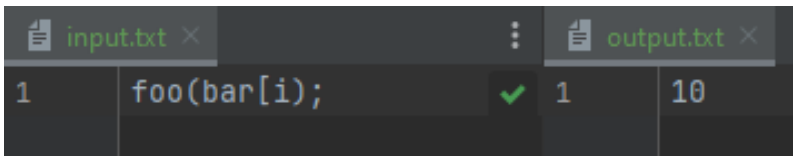
Если наш стэк оказывается пустым, а нам нужна закрывающаяся скобка, мы выводим индекс. Если мы доходим до конца и в стеке все еще есть элементы, мы выводим индекс самой первой скобки, потому что в стеке больше нет закрывающих скобок. Мы выводим «Success» только в одном случае — если проверка завершается с пустым стеком.

```
def Bracket(line):
    for i in range(len(line)):
        if line[i] not in ['(', ')', '[', ']', '{',
'}']:
            continue #следующий проход цикла
        if line[i] in dict.values():
            S.append([line[i], i])
        else:
            if S:
                if S[-1][0] == dict[line[i]]:
                    S.pop()
                else:
                    S.append([line[i], i])
                    for p in range(len(S) - 1, -1, -
1):
                        if S[p][0] in dict.keys():
                            return str(S[p][-1] + 1)
                    return str(S[0][-1] + 1)
            else:
                return str(i + 1)
    if not S:
```

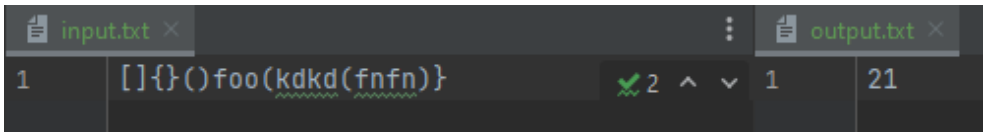
```
        return 'Success'
    else:
        return str(S[0][-1] + 1)

S = []
dict = {')': '(', ']': '[', '}': '{'} #hashmap
f = open('input.txt')
line = f.readline().split()
idx = Bracket(line[0])
f.close()
w = open('output.txt', 'w')
w.write(idx)
w.close()
```

Результат работы кода на примере из текста задачи:





Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0018114000000000047	0.017444610595703125
Пример из задачи	0.0012062000000000045	0.017498016357421875
Пример из задачи	0.0014859999999999873	0.017511367797851562
Верхняя граница	0.003851000000000049	0.01954174041748047

диапазона значений входных данных из текста задачи		

Вывод по задаче: ну, это жесть…это первая задача, которую я делала очень долго и с интеллектуальной помощью программиста. Не одобряю, задача не для меня.

Задача №5. 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()`.

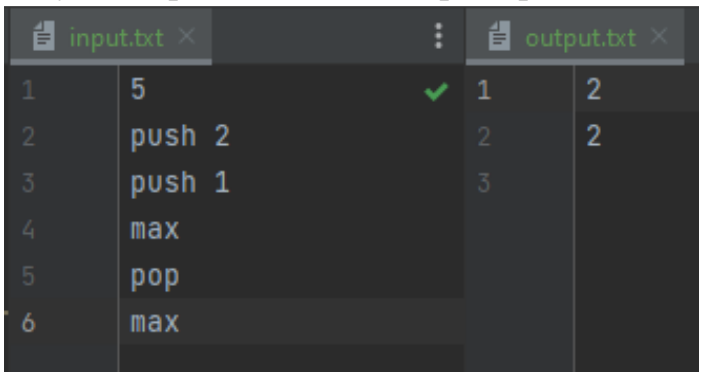
- **Формат входного файла (input.txt).** В первой строке входного файла содержится n ($1 \le n \le 400000$) – число команд. Последующие n строк исходного файла содержит ровно одну команду: `push V`, `pop` или `max`. $0 \le V \le 10^5$.
- **Формат выходного файла (output.txt).** Для каждого запроса `max` выведите (в отдельной строке) максимальное значение стека.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.

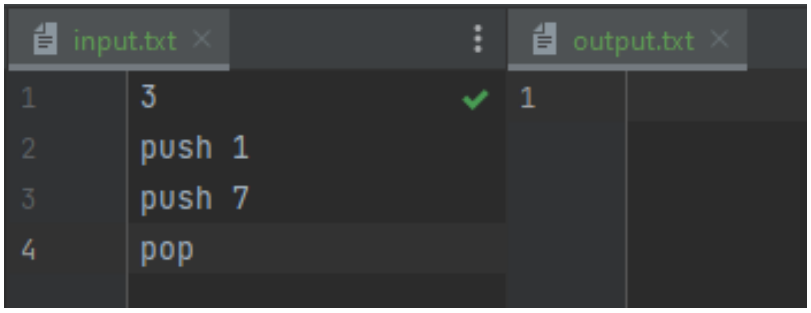
Максимум буду хранить в первом элементе стэка (S[0][0]), чтобы сравнивать с другими элементами. Для начала там будет -1, т.к. все остальные значения неотрицательны. Каждый поступающий элемент сопровождается текущим максимальным значением (первое предложение абзаца). Максимум динамически обновляется. Если вдруг удаляем элемент с головы стека, то и текущий максимум в S[0][0] изменяем на максимум рядом с последним элементом. Когда вызывается максимум, то выводим крайнее значение массива.

```
def push(number, S):
    current_max = max(number, S[0][0])
    S[0][0] = current_max
    S.append([number, current_max])


def pop(S):
```

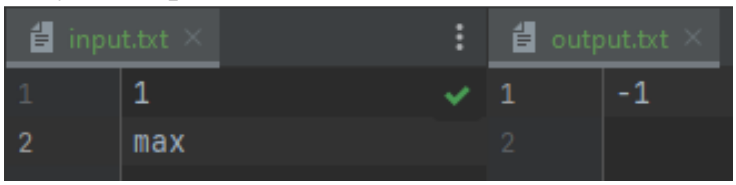
```
    S.pop()
    S[0][0] = S[-1][-1]


def get_max(S):
    current.append(S[-1][-1])
    return current


S = [[-1]]  # двумерный массив
current = []

f = open("input.txt")
iterations = int(f.readline())
for i in range(iterations):
    line = list(f.readline().splitlines())
    line = list(line[0].split())
    if len(line) > 1:
        number = int(line[-1])
        push(number, S)
    else:
        action = line[0]
        if action == "pop":
            pop(S)
        else:
            get_max(S)

w = open("output.txt", "w")
for elem in current:
    w.write(str(elem) + '\n')
w.close()
```

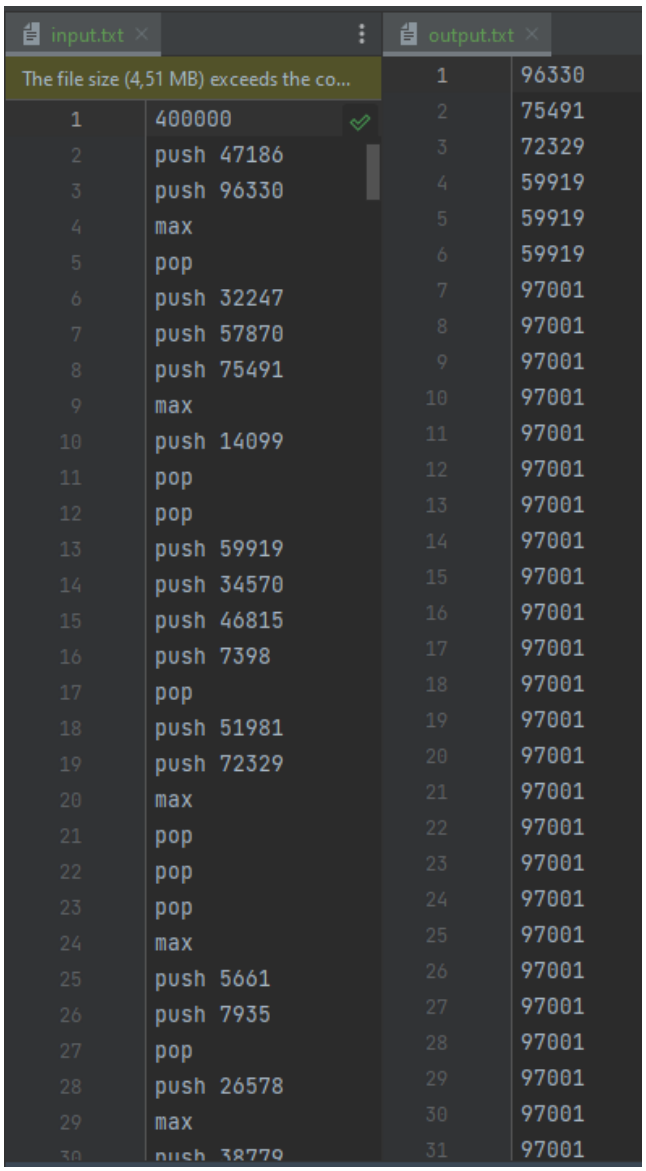
Результат работы кода на примерах из текста задачи:





Результат работы кода на максимальных и минимальных значениях:





	Время выполнения, с	Затраты памяти, мб

Нижняя граница диапазона значений входных данных из текста задачи	0.002960299999999999	0.01976776123046875
Пример из задачи	0.0011937000000000059	0.02069091796875
Пример из задачи	0.0011427999999999994	0.01976776123046875
Верхняя граница диапазона значений входных данных из текста задачи	5.045118799999999	11.08068561553955

Вывод по задаче: и снова затрачено куча сил, чтобы понять, как составить алгоритм этой задачи. Её отношу к разряду непростых.

Задача №8. Постфиксная запись

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как A B +. Запись B C + D * обозначает привычное нам (B + C) * D, а запись A B C + D * + означает A + (B + C) * D. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

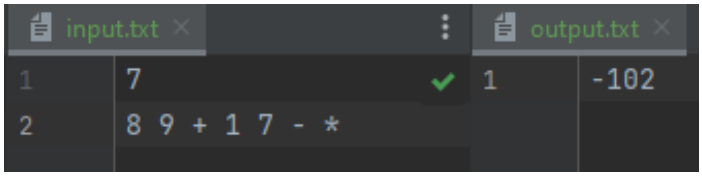
- **Формат входного файла (input.txt).** В первой строке входного файла дано число N ($1 \le n \le 10^6$) – число элементов выражения. Во второй строке содержится выражение в постфиксной записи, состоящее из N элементов. В выражении могут содержаться неотрицательные однозначные числа и операции +, -, *. Каждые два соседних элемента выражения разделены ровно одним пробелом.
- **Формат выходного файла (output.txt).** Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений, по модулю будут меньше, чем 2^{31}.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

Считываем строку, создаём стэк, в цикле for/in строка считывается посимвольно. Если символ число, он отправляется в стэк, а если оператор – проводим арифметику с помощью лямбда выражений (привет джава),

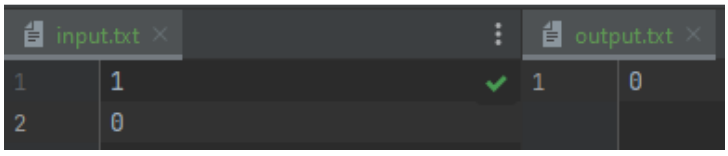
извлекая числа из массива функией pop(). Возвращаем полученное число в стэк из переменной z. Ответ будет храниться с голове стэка.

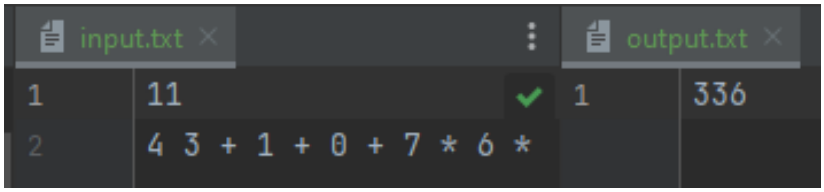
```
def calculator(line):
    S = []
    for i in line:
        if i.isdigit():
            S.append(int(i))
            continue
        y = S.pop()
        x = S.pop()
        z = {
            '+': lambda x, y: x + y,
            '-': lambda x, y: x - y,
            '*': lambda x, y: x * y,
            '/': lambda x, y: x // y
        }[i](x, y)
        S.append(z)
    return S.pop()

f = open('input.txt')
n = int(f.readline())
line = f.readline().split()
f.close()
w = open('output.txt', 'w')
w.write(str(calculator(line)))
w.close()
```

Результат работы кода на примере из текста задачи:



Результат работы кода на максимальных и минимальных значениях:





	Время выполнения,с	Затраты памяти, мб
Нижняя граница диапазона значений входных данных из текста задачи	0.0010000000000000009	0.017287254333496094
Пример из задачи	0.001162300000000005	0.017357826232910156
Верхняя граница диапазона значений входных данных из текста задачи	0.0010349000000000053	0.017374038696289062

Вывод по задаче: задача кажется крутой на фоне двух предыдущих. Не стала генерировать огромный файл, потому что нужно сохранять логику алгебраического выражения.

Вывод

Стэки и очереди интересные структуры данных, они довольно простые и функциональные. Но какие задачи придумали на эти темы….да, треш.