

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

Содержание отчета	2
Задачи по варианту	3
Задача №1. Высота дерева	3
Задача №4. Наибольшая возрастающая подпоследовательность	7
Вывод	10
Задача №6. Очередь с минимумом.	10
Задача №11. Бюрократия	14

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 ($1 \leq M \leq 106$) – 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:

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

output.txt:

1	10
2	1234

Результат работы кода на максимальных:

input.txt:

1	1000000
2	-
3	+ 784337812
4	-
5	-
6	+ 842278828
7	+ 401484070
8	-
9	-
10	-
11	-
12	+ -801738432
13	+ 287569448
14	-
15	-
16	+ 214045653
17	+ 275653389
18	-

output.txt:

1	784337812
2	401484070
3	842278828
4	287569448
5	-801738432
6	275653389
7	214045653
8	-178205648
9	831851634
10	775578479
11	756474397
12	-137098332
13	-783269664
14	685765264
15	155858343
16	641348131
17	618093506
18	414047656

Результат работы кода на минимальных значениях:

input.txt:

1	1
2	+ 1

output.txt:

1	
---	--

	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 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 скобочкой, 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 `(` 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` – $1 \leq S \leq 105$.

- 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 закончилась, а элементы в стеке остались, то выводим длину строки.

Результат работы кода на примерах из текста задачи:

input.txt:

1	<code>[()]</code>
---	-------------------

output.txt:

1	Success
---	---------

input.txt:

1	foo(bar);
---	-----------

output.txt:

1	Success
---	---------

Результат работы кода на максимальных:

input.txt:

[illegible]

output.txt:

1	4
---	---

Результат работы кода на минимальных значениях:

input.txt:

1	{ }
---	-----

output.txt:

1	Success
---	---------

	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:

1	7
2	+ 1
3	? 10
4	+ 10
5	?
6	-
7	?
8	-

output.txt:

1	1
2	10

Результат работы кода на максимальных:

input.txt:

1	1000000
2	+ 190591141
3	+ 510228006
4	-
5	?
6	+ -973906078
7	+ -272847346
8	+ 834707150
9	+ 340067956
10	+ -54103461
11	?
12	-
13	-
14	+ 379872766
15	?
16	+ 967669417
17	+ -333534336
18	-

output.txt:

1	510228006
2	-973906078
3	-272847346
4	-333534336
5	-333534336
6	-333534336
7	-333534336
8	-333534336
9	-333534336
10	-333534336
11	-805310779
12	-805310779
13	-393315548
14	-393315548
15	381285862
16	381285862
17	-708983240
18	-708983240
19	-158987857

Результат работы кода на минимальных значениях:
input.txt:

1	1
2	+ 1

output.txt:

1	

	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 a_i 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 ($1 \leq n \leq 105$) и количество справок m ($0 \leq m \leq 109$). Во второй строке для каждого посетителя в порядке очереди указано количество справок a_i ($1 \leq a_i \leq 106$), 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 заново в очередь. В случае если очередь пуста выводим -1.

1	3 2
2	1 2 3

output.txt:

1	2
2	3 1

input.txt:

1	4 5
2	2 5 2 3

output.txt:

1	3
2	4 1 2

Результат работы кода на максимальных:

input.txt:

1	100000 1000000000
2	49018 266699 138759 314976 883067 788750 742145 310048 361103 236476 271484 452907 793824 515077 469638 74764 274391 742048 851867 179093 974592 504453 126037 475005 349218 907021 166339 899922 928314 676420 121651 169785 721836 236094 538019 748623 92791 23381 594812 173581 422435 770106 49411 181637 697138 672230 970510 755245 269929 799389 815333 205807 608416 247210 237995 848105 341569 524187 635514 765397 737431 555467 503531 116864 138558 332749 411446 774416 868389 264384 173540 189628 79060 855047 782277 504563 65499 477818 314839 281688 114245 681809 801799 955571 241301 693821 934964 220474 219393 602774 400944 123349 487974 310945 179178 497210 211083 693102 775617 275222 376281 273140 805382 985332 373139 124097 882559 141406 597596 79998 177805 750347 297277 153098 159328 535482 533698 505019 550357 253309 462043 167311 180446 966885 856430 290110 445286 82936 598948 485750 188294 316013 627975 9468 691265 970964 316324 79898 302561 108813 341540 625004 293392 728271 815406 403138 985850 200391 428157 720968 559938 889106 233868 946983 921760 452725 215556 904589 747141 778567 750896 490925 149624 257828 630790 286359 464009 866299 567066 263465 89523 949225 602236 403077 464435 415823 337799 30817 603428 161326 697907 220621 823683 341986 319522 278391 673484 883201 647782 336719 81143 464896 44157 190862 264070 775963 576756 867319 128889 137880 476771 870449 534901 845218 349019 299494 721463 105460 7093 585483 467926 648638 418407 814694 415078 126029 868756 198977 414866 726280 820037 913562 24311 62646 844921 332853 485117 950504 897558 781208 709073 967235 693168 400347 65411 473091 586604 558350 481754 870541 885153 167784 829164 520637 469994 898481 450719 378411 709108 894771 617881 225416 178948 581299 943615 723455 484482 139638 103178 684402 772947 348799 434011 726761 244660 205885 416680 858397 415977 300725 666347 924468 506667 488644 568344 831482 661202 517354 175605 600570 622138 498036 478369 886415 877018 297103 298065 221549 554578 660161 26198 865293 577192 954014 152113 520133 660939 896411 94009 804042 611641 382899 546657 655542 220246 746959

output.txt:


```

1 88959
2 1 517 560822 684746 579411 928510 67321 401518 180108 811639 546819 704275 339526 242983 85293 911007 566641 333703 791469
777168 943547 857882 760125 742234 584760 148274 324459 491654 772450 973727 851081 216641 252578 529079 441503 351009 387629
333285 172964 790923 200811 330393 190115 560989 961721 35097 939228 857489 197589 283478 709269 922507 953454 431744 932323
88540 299625 614423 817391 901488 104259 677089 491274 160326 971251 94363 137595 609166 362068 703497 53129 417037 138109
166524 156958 908780 959353 331144 387138 251797 728714 674668 352791 44962 971906 415039 504720 678780 296361 962244 447078
972306 804930 13491 862472 252103 378592 647868 752168 664984 812603 989921 113839 432255 745227 849262 920374 746888 649386
702457 855481 831713 382128 382733 858099 194704 555546 440201 195032 61390 88116 295740 652483 870765 556226 530130 330381
944948 339636 146586 784001 195361 200802 745314 885344 56366 326447 597247 525884 393888 896718 458265 678240 421874 48732
448733 804766 247321 837514 430077 240398 301550 761515 339494 229761 408107 684362 464912 280750 820550 892081 397712 676028
717794 814080 917104 550233 320643 92108 852659 330784 520669 913396 704802 291345 978997 601530 452926 337361 5877 839362
658548 740738 329824 602824 870937 530800 741905 404794 98662 442925 619696 840906 134525 570210 470137 16279 508848 336008
14376 373683 335014 676573 333617 420277 61778 553106 280774 208334 241425 777655 158793 790123 506161 234021 682472 198732
338078 151627 974443 504967 349446 891662 305768 697268 91196 785234 379601 529603 651554 327076 856455 672323 379524 453802
214352 643825 889674 287809 446132 801523 553857 408039 406910 324287 480413 93831 585788 903310 297955 43325 732407 538017
656685 131494 40853 104117 429695 255055 894843 893769 508990 55319 651442 33389 436623 372036 920569 868184 823652 205806
287707 66246 821285 807549 878386 599668 212054 886259 563289 839990 138640 489678 576686 273152 882760 432691 655016 59617

```

Результат работы кода на минимальных значениях:

input.txt:

```

1 1 0
2 1|

```

output.txt:

```

1 1|
2 1

```

	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.