Алгоритмы для начинающих

Тест 2: Линейные структуры

10 ноября 2018

ФИО

_____

1. Общие знания+

Задача 1

Напишите формулу деления с округлением вверх числа `a` на число `b`. Нужно использовать целочисленное деление (обозначение `'//'`). Нельзя использовать функции округления (такие как `abs, int, floor`). Например $\lceil 5/3 \rceil = 2, \lceil 9/3 \rceil = 3$

_____

_____

Задача 2

Сколько чисел необходимо проверить на простоту, чтобы найти все простые числа до `10000`? Каким образом нужно выбирать числа? Какие алгоритмы проверки чисел на простоту бывают?

_____

_____

Задача 3

Алгоритм Евклида. На каком равенстве основан алгоритм Евклида?

_____

_____

Задача 4

Какая реализация будет работать быстрее? Рекурсивная или нерекурсиваная? Почему?

_____

_____

Задача 5

Чем может быть вызвана ошибка “Превышено максимальное время работы”?

_____

_____

Задача 6

Чем может быть вызвана ошибка “Ошибка во время работы программы”?

_____

_____

Задача 7

Чем, кроме неверного решения, может быть вызвана ошибка “Неправильный ответ”?

_____

_____

2. Стеки и очереди

F - First, L - Last, I - in, O - out

Задача 1

Что из этого стек?

1. FIFO
2. FILO
3. LILO
4. LIFO

Задача 2

Что из этого очередь?

1. FIFO
2. FILO
3. LILO
4. LIFO

Задача 3

Какие операции легко поддерживать на стеке?

1. Максимум на стеке
2. Минимум на стеке
3. Сумму на стеке
4. Призведение на стеке
5. НОД

Задача 4

Какие операции легко поддерживать на очереди?

1. Максимум на очереди
2. Минимум на очереди
3. Сумму на очереди
4. Призведение на очереди
5. НОД на очереди

Задача 5

Чему соответствует идея двух указателей

1. Стеку
2. Очереди

3. Префикс функции

Немного Теории: Срезы в Питоне.

s[start:stop:step], start включен, stop не включен

```
>>> s = [1, 2, 3, 4, 6] #простой список
>>> s[-1] #последний элемент списка
6
>>> s[1:] # все элементы кроме первого
[2, 3, 4, 6]
>>> s[-3:] # последние 3 элемента
[3, 4, 6]
>>> s[2:-2] #откидываем первые и последние 2
[3]
>>> s[::2] #парные элементы
[1, 3, 6]
>>> s[1:4:2] #элементы с первого по четвертый с шагом 2
[2, 4]
>>> s[4:-1:-2] #элементы с последнего по нулевой с шагом -2
[6, 3, 1]
```

Задача 1

Какие из следующих срезов являются префиксами массива a?

1) a[1:] 2) a[:1] 3) a[:5] 4) a[5:] 5)a[4:0:-1]
6) a[4:-1:-1][::-1] 7) a[:-1] 8) a 9)a[::-1]

Задача 2

Что должно быть вместо пропуска при подсчёте максимума на суффиксах?

```
x - массив

n = len(x)
suf_max = [0] * n
suf_max[-1] = x[-1]
for i in range(      , -1, -1):
    suf_max[i] = max(x[i], suf_max[i + 1])

```

_____

_____

Задача 3

Что должно быть вместо пропусков(A, B) при подсчёте суммы на префиксах?

```
x - массив

n = len(x)
pref_sum = [0] * n
pref_sum[0] = x[0]
for i in range(1, n):
    pref_sum[i] = x[  A  ] + pref_sum[  B  ]

```

_____

_____

Задача 4

Выберите варианты кода, который подсчитывает некоторую функцию на префиксе.

```
prx = [a[0]]
for i in a[1:]:
    prx.append(prex[-1] + i)
```

```
prx = [0] * len(a)
prx[0] = a[0]
for i in range(1, len(a)):
    a[i] = a[i-1] + prx[i]
```

```
prx = [0] * len(a)
prx[0] = a[0]
for i in range(1, len(a)):
    prx[i] = min(prx[i-1], a[i])
```

```
prx = [0] * len(a)
prx[0] = a[0]
for i in range(1, len(a)):
    x, y = prx[i-1], a[i]
    while y != 0:
        x, y = y, x%y
    prx[i] = x
```

Задача 5

Что подсчитывает последний (правый-нижний) код? В каких случаях это осмысленно?

_____

_____

Задача 6

Часто (но есть исключения, например, минимум на префиксе) использовать префикс-функции имеет смысл, когда операция обратима. т.е. Найдётся такая операция `-`, что, если `a+b+c=x`, то `x-a = b+c`. Какие обратимые операции вы знаете? Является ли НОД обратимой операцией?

_____

_____