**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Курсовая работа

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БФИ2001

Заморский Пётр Владимирович

Вариант 7

Научный руководитель: Симонов С.Е.

Москва, 2022

Цель работы: Разработать алгоритмы для решения 10 задач, для каждой задачи реализовать ввод/вывод в консоль с проверкой введенных значений.

Ход работы:

Перед реализацией кода для решения самих задач, определим функции для валидации ввода данных от пользователя:

def input\_string(prompt: str, min\_len: int, max\_len: int) -> str:

"""Read a string from a user with validation"""

s = input(prompt)

if not min\_len <= len(s) <= max\_len:

print(f"Input error: input string length must in range [{min\_len}, {max\_len}]")

return None

return s

def input\_int(prompt: str, min\_value: int, max\_value: int, default\_value: int = 0) -> int:

"""Read a single int from a user with validation"""

s = input(prompt)

try:

value = int(s)

except ValueError:

print(f"Input error: input must be a valid single integer number")

return default\_value

if not min\_value <= value <= max\_value:

print(f"Input error: input integer number length must in range [{min\_value}, {max\_value}]")

return default\_value

return value

def input\_ints(prompt: str, count: int = 0, minmax: tuple[int, int] | list[tuple[int, int]] = None) -> list[int]:

"""Read a whitespace-separated int list from a user with validation"""

if type(minmax) is list: count = len(minmax)

s = input(prompt).split()

if len(s) != count:

print(f"Input error: there are must be {count} integer numbers")

return []

result = []

for i in range(count):

try:

value = int(s[i])

except ValueError:

print(f"Input error: input item must be a valid integer number")

return []

if type(minmax) is list and not minmax[i][0] <= value <= minmax[i][1]:

print(f"Input error: input integer number length must in range [{minmax[i][0]}, {minmax[i][1]}]")

return []

if type(minmax) is tuple and not minmax[0] <= value <= minmax[1]:

print(f"Input error: input integer number length must in range [{minmax[0]}, {minmax[1]}]")

return []

result.append(value)

return result

1. Корова Бесси только что перехватила сообщение, которое Фермер Джон отправил Бургер Куин! Бесси уверена, что в нем скрыто секретное сообщение.

Сообщение представляет из себя строку 𝑠, состоящую из строчных букв латинского алфавита. Бесси считает, что строка 𝑡 скрыта в строке 𝑠, если 𝑡 является подпоследовательностью 𝑠, индексы которой формируют арифметическую прогрессию. Например, строка aab скрыта в строке aaabb, потому что она получена в индексах 1, 3 и 5, которые формируют арифметическую прогрессию с шагом 2. Бесси считает, что любая скрытая строка, которая имеет наибольшее количество вхождений и является скрытым сообщением. Два вхождения подпоследовательности 𝑆 различны, если различаются их множества индексов. Помогите Бесси узнать количество вхождений скрытого сообщения!

Например, в строке aaabb, a скрыта 3 раза, b скрыта 2 раза, ab скрыта 6 раз, aa скрыта 3 раза, bb скрыта 1 раз, aab скрыта 2 раза, aaa скрыта 1 раз, abb скрыта 1 раз, aaab скрыта 1 раз, aabb скрыта 1 раз и aaabb скрыта 1 раз. Количество вхождений скрытого сообщения равно 6.

Входные данные

В первой строке задана строка 𝑠, состоящая из строчных букв латинского алфавита (1≤|𝑠|≤105) — текст, который перехватила Бесси.

Выходные данные

Выведите единственное число — количество вхождений секретного сообщения.

Решение:

Листинг 1

def sequence\_code(s: str) -> int:

"""Finds a count of secret message occurrences in string"""

d = dict()

result = 0

for start in range(len(s)):

c = d.get(s[start], 0) + 1

d[s[start]] = c

result = max(result, c)

for steps in range(2, len(s)):

step = 1

l = (steps - 1) \* step

while start + l < len(s):

part = s[start:start+l+1:step]

c = d.get(part, 0) + 1

d[part] = c

result = max(result, c)

step += 1

l = (steps - 1) \* step

return result

s = input\_string("Enter a string: ", 1, 1e5)

if s is not None:

print(f"Secret message occurrences = {sequence\_code(s)}")

1. Вам заданы две строки 𝑠 и 𝑡, каждая из которых состоит из строчных букв латинского алфавита. Вы собираетесь посимвольно напечатать строку 𝑠, начиная с первого символа и заканчивая последним.

Когда вы собираетесь напечатать какой-то символ, вместо того, чтобы нажать на кнопку, печатающую этот символ, вы можете нажать кнопку «Backspace». Нажатие на эту кнопку удаляет последний напечатанный символ, который еще не был удален (или ничего не делает, если все напечатанные символы уже удалены или вы еще не напечатали ни одного символа). Например, если строка 𝑠 — «abcbd», и вы нажимаете на кнопку Backspace вместо печати первого и четвертого символа, в результате получится строка «bd» (первое нажатие Backspace не удалит ни одного символа, а второе нажатие этой кнопки удалит символ «c»). Другой пример: если 𝑠 равна «abcaa», и вы нажимаете Backspace вместо двух последних букв, получается «a».

Вы должны определить, можно ли получить строку 𝑡, если вы попробуете набрать строку 𝑠, нажимая Backspace вместо нажатия кнопок, соответствующих некоторым (возможно, ни одному) буквам строки 𝑠.

Входные данные

В первой строке задано одно целое число 𝑞 (1≤𝑞≤105) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит строку 𝑠 (1≤|𝑠|≤105). Каждый символ строки 𝑠 — строчная буква латинского алфавита.

Вторая строка каждого набора входных данных содержит строку 𝑡 (1≤|𝑡|≤105). Каждый символ строки 𝑡 — строчная буква латинского алфавита.

Гарантируется, что суммарное по всем наборам входных данных количество символов во всех строках не превосходит 2⋅105.

Выходные данные

Для каждого набора выходных данных выведите «YES», если можно получить строку 𝑡, набирая строку 𝑠 и заменяя некоторые символы нажатиями клавиши Backspace, или «NO» в противном случае.

Решение:

Листинг 2

def can\_type(s: str, t: str) -> bool:

"""Returns 'True' if 't' can writed from 's', otherwise 'False'"""

pos\_s = 0

pos\_t = 0

start = 0

dist = 0

while pos\_t < len(t) and pos\_s < len(s):

if s[pos\_s] == t[pos\_t]:

if pos\_t == 0:

pos\_t += 1

start = pos\_s

else:

if dist % 2 == 0:

pos\_t += 1

else:

pos\_t = 0

pos\_s = start

dist = 0

else:

dist += 1

pos\_s += 1

return pos\_t == len(t)

def can\_type\_many(sets: list[tuple[str, str]]) -> list[bool]:

"""Bulk call for can\_type function"""

result = []

for s, t in sets:

if s is None or t is None: return []

result.append(can\_type(s, t))

return result

q = input\_int("Sets count (q): ", 1, 1e5)

input\_set = [(input\_string("s: ", 1, 1e5), input\_string("t: ", 1, 1e5)) for \_ in range(q)]

for r in can\_type\_many(input\_set):

print('YES' if r else 'NO')

1. В поездке в Луксор и Асуан Сахир зашел на нубийский рынок, чтобы купить сувениров друзьям и родственникам. Он узнал, что на рынке действуют странные правила. На рынке имеются n различных сувениров, пронумерованных от 1 до n. Сувенир номер i имеет базовую стоимость ai Египетских фунтов. Если Сахир купит k сувениров с индексами x1, x2, ..., xk, то стоимость сувенира xj будет равна axj + xj·k (для 1 ≤ j ≤ k). Другими словами, стоимость сувенира равна его базовой стоимости плюс его номер, умноженный на k.

Сахир хочет купить как можно больше сувениров, заплатив не более S Египетских долларов. Заметьте, что он не может купить никакой сувенир более, чем один раз. Если есть несколько вариантов купить как можно больше сувениров, он хочет минимизировать суммарную их стоимость. Помогите ему выбрать эти сувениры!

Входные данные

Первая строка содержит два целых числа n и S (1 ≤ n ≤ 105, 1 ≤ S ≤ 109) — количество сувениров и бюджет Сахира.

Вторая строка содержит n целых числе a1, a2, ..., an (1 ≤ ai ≤ 105) — базовые стоимости сувениров.

Выходные данные

В единственной строке выведите два целых числа k и T — максимальное число сувениров, которое может купить Сахир, и минимальную стоимость покупки этих k сувениров.

Решение:

Листинг 3

def souvenirs(S: int, a: list[int]) -> tuple[int, int]:

"""Returns count and total price of 'a' souvenirs that can be bought with 'S' money"""

def price(base, idx, k): return base + idx \* k

def sum\_price(count):

prices = [price(a[i], i + 1, count) for i in range(len(a))]

prices.sort()

sum = 0

for i in range(count):

sum += prices[i]

return sum

left = 0

right = len(a)

result\_sum = 0

while left < right:

mid = (left + right + 1) // 2

sum = sum\_price(mid)

if sum <= S:

result\_sum = sum

left = mid

else:

right = mid - 1

return left, result\_sum

[n, S] = input\_ints('Count and total money: ', minmax=[(1, 1e5), (1, 1e9)])

a = input\_ints(f'Base prices ({n}): ', count=n, minmax=(1, 1e5))

if len(a) > 0:

[can\_buy\_count, can\_buy\_price] = souvenirs(S, a)

print(f"{can\_buy\_count} {can\_buy\_price}")

1. Вам дана бинарная строка длины 𝑛 (т. е. строка, состоящая из 𝑛 символов '0' и '1').  
   За один ход вы можете поменять местами два соседних символа в строке. Какую лексикографически минимальную строку вы можете получить из заданной, если вы можете сделать не более, чем 𝑘 ходов? Возможно, что вы не сделаете ни одного хода.  
   Заметьте, что вы можете менять местами одну и ту же пару соседних символов с индексами 𝑖 и 𝑖+1 любое (возможно, нулевое) количество раз. Каждый такой обмен является отдельным ходом.  
   Вам необходимо ответить на 𝑞 независимых наборов входных данных.

Входные данные  
Первая строка входных данных содержит одно целое число 𝑞 (1≤𝑞≤104) — количество наборов входных данных.

Первая строка набора входных данных содержит два целых числа 𝑛 и 𝑘 (1≤𝑛≤106,1≤𝑘≤𝑛2) — длину строки и количество ходов, которые вы можете сделать.  
Вторая строка набора входных данных содержит одну строку, состоящую из 𝑛 символов '0' и '1'.  
Гарантируется, что сумма 𝑛 по всем наборам входных данных не превосходит 106 (∑𝑛≤106).

Выходные данные  
Для каждого запроса выведите ответ на него — лексикографически минимальную возможную строку длины 𝑛, которую вы можете получить из заданной, сделав не более, чем 𝑘 ходов.

Решение:  
Листинг 4

def try\_minimaze\_bin(k: int, s: str) -> str:

"""Minimizes 's' string of ones and zeros with 'k' count of swaps"""

ones = 0

added\_ones = 0

result = []

for i in range(len(s)):

if s[i] == '1':

ones += 1

else:

if ones > 0:

m = min(ones, k)

k -= m

cur\_ones = (ones - added\_ones)

for \_ in range(m - cur\_ones):

result.pop()

added\_ones -= 1

for \_ in range(cur\_ones - m):

result.append('1')

result.append('0')

for \_ in range(m):

result.append('1')

added\_ones += 1

if k == 0:

result.append(s[i+1:])

break

else:

result.append('0')

return ''.join(result)

def try\_minimaze\_bin\_many(sets: list[tuple[int, str]]) -> list[str]:

"""Bulk version of 'try\_minimaze\_bin'"""

results = []

for k, s in sets:

results.append(try\_minimaze\_bin(k, s))

return results

sets = []

for i in range(input\_int("Sets count (q): ", 1, 1e4)):

n = input\_int("String length (n): ", 1, 1e6)

k = input\_int("Swaps count (k): ", 1, n \*\* 2)

if n == 0 or k == 0: break

s = input\_string("Binary string: ", n, n)

if s is None: break

for c in s:

if c not in '01':

print(f"Input error: binary string can only contains 1's and 0's")

s = None

break

if s is None: break

sets.append((k, s))

for r in try\_minimaze\_bin\_many(sets):

print(r)

1. Перестановка — это последовательность из 𝑛 целых чисел от 1 до 𝑛, в которой все числа встречаются ровно по одному разу. Например, [1], [3,5,2,1,4], [1,3,2] — перестановки, а [2,3,2], [4,3,1], [0] — нет.

Поликарпу подарили перестановку 𝑝 чисел от 1 до 𝑛. Однако, когда Поликарп пришёл домой, он заметил, что в кармане перестановка 𝑝 превратилась в массив 𝑞 согласно следующему правилу:

𝑞𝑖=max(𝑝1,𝑝2,…,𝑝𝑖).

Теперь Поликарпу стало интересно: какую лексикографически минимальную и лексикографически максимальную перестановки ему могли подарить.

Массив 𝑎 длины 𝑛 лексикографически меньше массива 𝑏 длины 𝑛, если существует индекс 𝑖 (1≤𝑖≤𝑛), такой что первые 𝑖−1 элементов у массивов 𝑎 и 𝑏 совпадают, а 𝑖-й элемент у массива 𝑎 меньше, чем 𝑖-й элемент у массива 𝑏. Например массив 𝑎=[1,3,2,3] лексикографически меньше массива 𝑏=[1,3,4,2].

Например, если 𝑛=7 и 𝑝=[3,2,4,1,7,5,6], тогда 𝑞=[3,3,4,4,7,7,7] и следующие перестановки могли быть в качестве 𝑝 изначально:

[3,1,4,2,7,5,6] (лексикографически минимальная перестановка);

[3,1,4,2,7,6,5];

[3,2,4,1,7,5,6];

[3,2,4,1,7,6,5] (лексикографически максимальная перестановка).

Для заданного массива 𝑞 найдите лексикографически минимальную и лексикографически максимальную перестановки, которые могли быть подарены Поликарпу изначально.

Входные данные

В первой строке содержится одно целое число 𝑡 (1≤𝑡≤104). Далее следуют 𝑡 наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число 𝑛 (1≤𝑛≤2⋅105).

Вторая строка каждого набора входных данных содержит 𝑛 целых чисел 𝑞1,𝑞2,…,𝑞𝑛 (1≤𝑞𝑖≤𝑛).

Гарантируется, что массив 𝑞 был получен применением правила из условия к какой-то перестановке 𝑝.

Гарантируется, что сумма 𝑛 по всем наборам входных данных не превосходит 2⋅105.

Выходные данные

Для каждого набора входных данных выведите две строки:

в первой строке выведите 𝑛 целых чисел — лексикографически минимальную перестановку, которая могла быть подарена Поликарпу изначально; во второй строке выведите 𝑛 целых чисел — лексикографически максимальную перестановку, которая могла быть подарена Поликарпу изначально.

Решение:

Листинг 5

def permutation\_min\_max(q: list[int]) -> tuple[list[int], list[int]]:

free = []

m = 0

for i in range(len(q)):

if (i + 1) not in q:

free.append((i + 1))

if m < q[i]:

m = q[i]

else:

q[i] = 0

result\_min = []

result\_max = []

free\_idx = 0

for i in range(len(q)):

if q[i] > 0:

result\_min.append(q[i])

else:

result\_min.append(free[free\_idx])

free\_idx += 1

for i in range(len(q)):

if q[i] > 0:

result\_max.append(q[i])

m = q[i]

else:

free\_idx = 0

for j in range(len(free)):

if free[j] > m:

break

elif free[j] > free[free\_idx]:

free\_idx = j

result\_max.append(free[free\_idx])

free.pop(free\_idx)

return (result\_min, result\_max)

def permutation\_min\_max\_many(sets: list[list[int]]) -> list[tuple[list[int], list[int]]]:

"""Bulk version of 'permutation\_min\_max' function"""

results = []

for q in sets:

results.append(permutation\_min\_max(q))

return results

sets = []

for i in range(input\_int("Sets count (t): ", 1, 1e4)):

n = input\_int("Permutation length (n): ", 1, 2e5)

if n == 0: break

q = input\_ints(f"Permutation ({n} ints): ", count=n, minmax=(1,n))

for j in range(1, len(q)):

if q[j-1] > q[j]:

print("Input error: input ints sequence is not valid for this action")

q = []

break

if len(q) == 0: break

sets.append(q)

for p\_min, p\_max in permutation\_min\_max\_many(sets):

print(' '.join(map(str, p\_min)))

print(' '.join(map(str, p\_max)))

1. Стивен Квин хочет написать новый рассказ. Вы знаете, он очень необычный писатель — Стивен использует только буквы 'a', 'b', 'c', 'd' и 'e'!

Чтобы написать рассказ, Стивен выписал 𝑛 слов, состоящих из первых 5 строчных букв латинского алфавита. Из них он хочет выбрать максимальное количество слов таким образом, чтобы получился интересный рассказ.

Рассказ представляет собой список слов, необязательно различных. Рассказ называется интересным, если существует буква, которая встречается во всех входящих в него словах большее количество раз, чем все остальные буквы латинского алфавита вместе взятые.

Например, рассказ, состоящий из слов «bac», «aaada», «e», является интересным (буква «a» встречается в словах рассказа всего 5 раз, остальные — 4), а рассказ, состоящий из слов «aba», «abcde» — нет (не существует буквы, встречающейся во всех словах рассказа большее число раз, чем все остальные вместе взятые).

Вам дан список из 𝑛 слов. Выберите из этого списка максимальное количество слов так, чтобы они образовывали интересный рассказ. Если невозможно составить непустой интересный рассказ, выведите 0.

Входные данные

В первой строке записано одно целое число 𝑡 (1≤𝑡≤5000) — количество наборов входных данных. Далее следуют 𝑡 наборов входных данных.

Первая строка каждого набора данных содержит целое число 𝑛 (1≤𝑛≤2⋅105) — количество слов. Далее следует 𝑛 непустых строк — слов, состоящих из строчных букв латинского алфавита. Слова в списке не обязательно различны (иными словами, могут быть одинаковые слова). Слова состоят исключительно из букв 'a', 'b', 'c', 'd' и 'e'.

Гарантируется, что сумма 𝑛 по всем наборам входных данных не превышает 2⋅105 и что суммарное количество букв во всех словах, содержащихся в наборах входных данных, не превышает 4⋅105.

Выходные данные

Для каждого набора входных данных выведите максимальное количество слов из заданного списка, которые могут образовать интересный рассказ. Если невозможно составить непустой интересный рассказ, выведите 0.

Решение:

Листинг 6

def tale(words: list[str]) -> int:

"""Returns max length of interesting tale out of 'words' strings list. Only 'abcde' letters supported"""

result = 0

for c in 'abcde':

letters = []

for w in words:

letters\_target = 0

letters\_other = 0

for wc in w:

if wc == c:

letters\_target += 1

else:

letters\_other += 1

letters.append((letters\_target, letters\_other))

letters.sort(key=lambda l: l[0] - l[1], reverse=True)

cur\_result = 0

cur\_target = letters[cur\_result][0]

cur\_other = letters[cur\_result][1]

while cur\_result < len(words) - 1 and cur\_target > cur\_other:

cur\_result += 1

cur\_target += letters[cur\_result][0]

cur\_other += letters[cur\_result][1]

if cur\_target > cur\_other: cur\_result += 1

result = max(result, cur\_result)

return result

def tale\_many(sets: list[list[str]]) -> list[int]:

"""Bulk version of 'tale' function"""

results = []

for w in sets:

results.append(tale(w))

return results

sets = []

for i in range(input\_int("Sets count (t): ", 1, 5000)):

words\_count = input\_int("Words count (n): ", 1, 2e5)

words = []

for \_ in range(words\_count):

w = input()

for c in w:

if c not in 'abcde':

print(f"Input error: word can only contains abcde letters")

w = ''

break

if len(w) < 1:

print(f"Input error: word should be not empty string")

words = []

break

words.append(w)

if len(words) == 0:

sets.clear()

break

sets.append(words)

for r in tale\_many(sets):

print(r)

1. Маленький Дима много баловался на уроке математики, и злой учитель Леонид Иванович в наказание ему придумал следующую задачу.

Найти все целочисленные решения уравнения x (0 < x < 109):

x = b·s(x)a + c,

где a, b, c — некоторые заранее определенные константы, а функция s(x) обозначает сумму всех цифр в десятичном представлении числа x.

Учитель дает эту задачу Диме на каждом уроке, меняя только параметры уравнения: a, b, c. Диме надоело получать двойки, и он просит вас помочь ему решить эту нелегкую задачу.

Входные данные

В первой строке записаны три целых числа через пробел: a, b, c (1 ≤ a ≤ 5; 1 ≤ b ≤ 10000;  - 10000 ≤ c ≤ 10000).

Выходные данные

Выведите целое число n — количество найденных решений. Затем выведите n целых чисел в порядке возрастания — найденные решения заданного уравнения. Выводить нужно только целочисленные решения, большие нуля и строго меньшие, чем 109.

Решение:

Листинг 7

def dima\_equation(a: int, b: int, c: int) -> int:

def sum\_digits(n):

sum = 0

while n > 0:

sum += n % 10

n //= 10

return sum

result = []

for i in range(1, 9 \* 9):

x = b \* (i \*\* a) + c

if sum\_digits(x) == i and 0 < x < 10 \*\* 9:

result.append(x)

return result

[a, b, c] = input\_ints("a, b, c (whitespace separated): ", minmax=[(1, 5), (1, 10000), (-10000, 10000)])

r = dima\_equation(a, b, c)

print(len(r))

print(' '.join(map(str, r)))

1. Массив целых чисел 𝑝1,𝑝2,…,𝑝𝑛 называется перестановкой, если он содержит каждое число от 1 до 𝑛 ровно один раз. Например, следующие массивы являются перестановками: [3,1,2], [1], [1,2,3,4,5] и [4,3,1,2]. Следующие массивы перестановками не являются: [2], [1,1], [2,3,4].

Поликарп изобрел невероятно крутую перестановку 𝑝1,𝑝2,…,𝑝𝑛 длины 𝑛. К сожалению, он забыл эту перестановку. Единственное, что он помнит, это массив 𝑞1,𝑞2,…,𝑞𝑛−1 длины 𝑛−1, где 𝑞𝑖=𝑝𝑖+1−𝑝𝑖.

Заданы 𝑛 и 𝑞=𝑞1,𝑞2,…,𝑞𝑛−1, помогите Поликарпу по этой информации восстановить придуманную им перестановку.

Входные данные

Первая строка содержит целое число 𝑛 (2≤𝑛≤2⋅105) — длину перестановки, которую надо восстановить. Следующая строка содержит 𝑛−1 целое число 𝑞1,𝑞2,…,𝑞𝑛−1 (−𝑛<𝑞𝑖<𝑛).

Выходные данные

Выведите -1, если перестановки длины 𝑛, которая соответствует массиву 𝑞, не существует. В противном случае выведите 𝑝1,𝑝2,…,𝑝𝑛. Если существует несколько подходящих перестановок, то выведите любую из них.

Решение:

Листинг 8

def restore\_permutation(q: list[int]) -> list[int] | None:

"""Returns a valid permutation based on diffs list 'q' or None"""

p = [0] \* (len(q) + 1)

p[0] = 1

for i in range(1, len(p)):

if q[i - 1] < 0:

to\_add = -(p[i - 1] + q[i - 1] - 1)

to\_add = max(0, to\_add)

for j in range(i):

p[i-j-1] += to\_add

if p[i-j-1] > len(p): return -1

if p[i-j-1] == p[i - 1] + q[i - 1]: return -1

p[i] = p[i - 1] + q[i - 1]

else:

p[i] = p[i - 1] + q[i - 1]

if p[i] < i + 1:

for j in range(i):

if p[i-j-1] == p[i]: return -1

if p[i] > len(p): return -1

return p

n = input\_int("Permutation length (n): ", 2, 2e5)

if n > 0:

q = input\_ints("Array to restore (q): ", count=n-1, minmax=(-n, n))

if len(q) > 0:

r = restore\_permutation(q)

if r is not None:

print(' '.join(map(str, r)))

else:

print(-1)

1. Вам дан массив 𝑎 из 𝑛 целых чисел, пронумерованных от 1 до 𝑛.

Назовем 𝑘-удивительным числом массива минимальное число, которое встречается во всех подмассивах длины 𝑘 (напомним, что подмассивом массива 𝑎 длины 𝑘 называются 𝑘 подряд идущих элементов массива 𝑎). Если для некоторого 𝑘 не существует ни одного числа, встречающегося во всех подмассивах длины 𝑘, то 𝑘-удивительным числом считается −1.

Для каждого 𝑘 от 1 до 𝑛 найдите 𝑘-удивительное число массива 𝑎.

Входные данные

Первая строка теста содержит одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Первая строка набора содержит одно целое число 𝑛 (1≤𝑛≤3⋅105) — длина массива. Вторая строка набора содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤𝑛) — элементы массива.

Гарантируется, что сумма 𝑛 по всем наборам тестовых данных не превосходит 3⋅105.

Выходные данные

Для каждого набора тестовых данных выведите 𝑛 целых чисел, где 𝑖-е число является 𝑖-удивительным числом массива.

Решение:

Листинг 9

def k\_amazing(a: list[int]) -> list[int]:

"""Returns k-amazing numbers for 'a' list"""

result = [-1] \* len(a)

offset = [0] \* len(a)

last = [0] \* len(a)

for k in range(1, len(a) + 1):

x = a[k - 1]

offset[x - 1] = max(offset[x - 1], k - last[x - 1])

last[x - 1] = k

for x in range(1, len(a) + 1):

offset[x - 1] = max(offset[x - 1], len(a) - last[x - 1] + 1)

i = offset[x - 1]

while i <= len(a) and result[i - 1] == -1:

result[i - 1] = x

i += 1

return result

def k\_amazing\_many(sets: list[list[int]]) -> list[list[int]]:

"""Bulk version of 'k\_amazing' function"""

results = []

for a in sets:

results.append(k\_amazing(a))

return results

sets = []

for i in range(input\_int("Sets count (t): ", 1, 1000)):

n = input\_int("Array length (n): ", 1, 3e5)

if n == 0: break

a = input\_ints("Array (a): ", count=n, minmax=(1, n))

if len(a) == 0: break

sets.append(a)

for r in k\_amazing\_many(sets):

print(' '.join(map(str, r)))

1. Ура! Король Берлядии Берл II устраивает рыцарский турнир. Король уже разослал послание всем рыцарям королевства, а они, в свою очередь, дали согласие на участие в этом грандиозном событии.

Что же касается вас, то вы — простой крестьянин. Не удивительно, что рыцарский турнир вы проспали (ведь он проводился в выходной), и теперь вам очень хочется узнать его результаты. В этот раз рыцарский турнир в Берляндии проходил следующим образом:

В турнире принимали участие n рыцарей. Каждому рыцарю был присвоен уникальный номер — целое число от 1 до n.

Турнир проводился в m сражений, в i-ом сражении все еще не выбывшие рыцари с номерами не меньше li и не больше ri боролись за право продолжить участие в турнире.

После i-го сражения среди всех рыцарей, которые боролись, победил только один — рыцарь с номером xi, он продолжил участие в турнире. Остальные рыцари выбыли из турнира.

Победитель самого последнего (m-го) сражения (рыцарь с номером xm) был объявлен победителем всего турнира.

Вы узнали у своих друзей информацию про все сражения, и теперь для каждого рыцаря вам интересно знать, каким рыцарем он был побежден. Считается, что рыцарь с номером a победил рыцаря с номером b, если было такое сражение, в котором боролись оба этих рыцаря, а победителем из этого сражения вышел рыцарь с номером a.

Напишите программу, вычисляющую для каждого рыцаря, каким рыцарем он был побежден.

Входные данные

В первой строке записано два целых числа n, m (2 ≤ n ≤ 3·105; 1 ≤ m ≤ 3·105) — количество рыцарей и количество сражений. В каждой из следующих m строк записано три целых числа li, ri, xi (1 ≤ li < ri ≤ n; li ≤ xi ≤ ri) — описание i-го сражения.

Гарантируется, что входные данные корректны и соответствуют условию задачи.\ Гарантируется, что в каждом сражении участвовали как минимум два рыцаря.

Выходные данные

Выведите n целых чисел. Если i-ый рыцарь был побежден, то i-ое число должно быть равно номеру рыцаря, который победил рыцаря с номером i. Если i-ый рыцарь является победителем турнира, то i-ое число должно быть равно 0.

Решение:

Листинг 10

def tournament(n: int, table: list[tuple[int, int, int]]) -> list[int]:

"""Returns array with length 'n' of knight's slayers, using turnament's battles list 'table'"""

result = [0] \* n

for i in range(len(table)):

l = table[i][0]

r = table[i][1]

x = table[i][2]

for j in range(l, r + 1):

if j == x or result[j - 1] > 0: continue

result[j - 1] = x

return result

[n, m] = input\_ints("Knights count (n) and battles count (m): ", minmax=[(2, 3e5), (1, 3e5)])

table = [input\_ints('', 3, minmax=(1, n)) for \_ in range(m)]

print(' '.join(map(str, tournament(n, table))))

Вывод: реализованы алгоритмы для решения всех 10 задач.