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 БФИ 2001

Чуйкин М.Е.

Москва, 2022

Вариант 6.

Задание 1.

Скоро состоится большой футбольный чемпионат! 𝑛 команд будут участвовать в нем, и каждая пара различных команд сыграет ровно один матч друг с другом.

У матча может быть два различных исхода:

матч может закончиться вничью, тогда обе команды получат по 1 очку; одна из команд может победить, тогда победившая команда получит 3 очка, а проигравшая — 0.  
Счет команды — суммарное количество очков, полученное командой за все матчи, в которых она играла.

Вас интересует гипотетическая ситуация, в которой все команды будут в конце чемпионата иметь одинаковый счет. Простой пример такой ситуации — если все матчи закончатся вничью, но вас интересует подобная ситуация с минимально возможным количеством ничьих.

Ваша задача — найти ситуацию (выбрать результат каждой игры), в которой у всех команд одинаковый итоговый счет, а количество ничьих минимально возможно.

Входные данные  
В первой строке задано одно целое число 𝑡 (1≤𝑡≤100) — количество наборов входных данных.

Затем следуют сами наборы входных данных. Каждый набор описывается одной строкой, содержащей целое число 𝑛 (2≤𝑛≤100) — количество команд.

Выходные данные  
Для каждого набора входных данных выведите 𝑛(𝑛−1)2 целых чисел, описывающих результаты игр в следующем порядке: первое число соответствует результату матча между командой 1 и командой 2, второе — между командой 1 и командой 3, затем 1 и 4, ..., 1 и 𝑛, 2 и 3, 2 и 4, ..., 2 и 𝑛, и так далее, до результата матча между командой 𝑛−1 и командой 𝑛.

Число, соответствующее результату матча между командой 𝑥 и командой 𝑦, должно быть равно 1, если побеждает команда 𝑥, −1, если побеждает команда 𝑦, или 0, если матч заканчивается вничью.

Все команды должны иметь одинаковый итоговый счет, и количество ничьих должно быть минимально возможным. Если оптимальных ответов несколько — выведите любой из них. Можно показать, что всегда есть способ составить результаты игр так, что все команды имеют одинаковый итоговый счет.

Пример  
входные данные  
2  
2  
3  
выходные данные  
0  
1 -1 1\ Примечание  
В первом наборе входных данных из примера обе команды получают по 1 очку, так как их матч заканчивается вничью.

Во втором наборе входных данных из примера команда 1 побеждает команду 2 (команда 1 получает 3 очка), команда 1 проигрывает команде 3 (команда 3 получает 3 очка), и команда 2 выигрывает у команды 3 (команда 2 получает 3 очка).

Ход решения.

Идея решения довольно проста. Все случаи делятся на 2 типа:

1. Нечетное количество команд
2. Четное количество команд

В первом случае у каждой команды будет n – 1 матч, то есть для поддержания равного количества очков у команд и минимизации количества ничьих, каждая команда должна выйграть первые (n-1)/2 матчей и проиграть оставшиеся. Не тружно заметить, что благодаря таким действиям у каждой команды будет равное количество очков и ничьих не будет.

Во втором случае обойтись без ничьих не удастся. В данном случае мы изменим наш алгоритм и будем действовать в таком ключе: пусть каждая команда выйграет первые (n-2)/2, затем сыграет в ничью и проиграет оставшиеся.

Код решения:

def solutionOfnum1(countsOfTeams):

results = []

for count in countsOfTeams:

temp = ''

if count % 2 == 1:

for i in range(count):

for j in range(i + 1, count):

if j - i <= count / 2:

temp += '1 '

else:

temp += '-1 '

else:

for i in range(count):

for j in range(i + 1, count):

if j - i < count / 2:

temp += '1 '

elif j - i == count / 2:

temp += '0 '

else:

temp += '-1 '

results.append(temp)

return results

def main1():

try:

t = int(input('Введите число наборов данных (от 1 до 100)').replace(' ', ''))

if t < 1 or t > 100:

print('Неверный ввод\nЗавершение работы программы')

return 1

countsOfTeams = []

for i in range(t):

temp = int(input('Введите число команд (от 2 до 100)').replace(' ', ''))

if temp < 2 or temp > 100:

print('Неверный ввод\nЗавершение работы программы')

return 1

countsOfTeams.append(temp)

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum1(countsOfTeams=countsOfTeams)

for result in results:

print(result)

return 0

main1()

Задание 2.

У вас есть последовательность 𝑎 из 𝑛 элементов 1,2,3,…,𝑘−1,𝑘,𝑘−1,𝑘−2,…,𝑘−(𝑛−𝑘) (𝑘≤𝑛<2𝑘).

Назовем инверсией в 𝑎 пару индексов 𝑖<𝑗 таких, что 𝑎[𝑖]>𝑎[𝑗].

Предположим, что у вас есть некоторая перестановка 𝑝 размера 𝑘 и вы строите последовательность 𝑏 размера 𝑛 следующим образом: 𝑏[𝑖]=𝑝[𝑎[𝑖]].

Ваша задача — найти такую перестановку 𝑝, что суммарное количество инверсий в 𝑏 не превосходит суммарного количества инверсий в 𝑎, и 𝑏 лексикографически максимальна.

Напоминание: последовательность из 𝑘 целых чисел называется перестановкой, если она содержит все числа от 1 по 𝑘 ровно по одному разу.

Еще одно напоминание: последовательность 𝑠 лексикографически меньше последовательности 𝑡, если либо 𝑠 — префикс 𝑡, или для первого 𝑖, для которого 𝑠𝑖≠𝑡𝑖, выполняется 𝑠𝑖<𝑡𝑖 (в первой позиции, в которой эти последовательности различаются, элемент в 𝑠 меньше элемента в 𝑡).

Входные данные  
В первой строке задано одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных.

В первой и единственной строке каждого набора заданы два целых числа 𝑛 и 𝑘 (𝑘≤𝑛<2𝑘; 1≤𝑘≤105) — длина последовательности 𝑎 и ее максимум.

Гарантируется, что сумма 𝑘 по всем наборам входных данных не превосходит 105.

Выходные данные  
Для каждого набора входных данных, выведите 𝑘 целых чисел — перестановку 𝑝, которая максимизирует 𝑏 лексикографически, не увеличивая суммарное количество инверсий.

Можно доказать, что 𝑝 существует и единственная.

Пример  
входные данные  
4  
1 1  
2 2  
3 2  
4 3  
выходные данные  
1  
1 2  
2 1  
1 3 2 \

Примечание  
В первом наборе входных данных, последовательность 𝑎=[1], поэтому существует только одна перестановка 𝑝=[1].

Во втором наборе, последовательность 𝑎=[1,2]. В 𝑎 нет инверсий, а потому только одна перестановка 𝑝=[1,2] не увеличивает количество инверсий.

В третьем наборе, 𝑎=[1,2,1] и имеет 1 инверсию. Если мы используем 𝑝=[2,1], то 𝑏=[𝑝[𝑎[1]],𝑝[𝑎[2]],𝑝[𝑎[3]]]=[2,1,2] и также имеет 1 инверсию.

В четвертом наборе, 𝑎=[1,2,3,2], и так как 𝑝=[1,3,2] то 𝑏=[1,3,2,3]. И 𝑎, и 𝑏 имеют по 1 инверсии и 𝑏 — лексикографически максимальна.

Ход решения.

Так как элементы 1..k-(n-k)-1 не повторяются, а значит не увеличивают количество инверсий, то мы их выводим без изменений.

Так как осташиеся элементы возрастают от k-(n-k) до k и затем убывают до k-(n-k), то для сохранения количества инвесий можно вывести посделовательность k, k-1..k-(n-k).., k-1, k. Но так как нас интересует не вся последовательность, то достаточно будет вывести лишь элементы k..k-(n-k)

Код решения.

def solutionOfnum2(data):

results = []

for nk in data:

n = nk[0]

k = nk[1]

temp = [i for i in range(1, k - (n - k))]

for i in range(k, k - (n - k) - 1, -1):

temp.append(i)

results.append(temp)

return results

def main2():

try:

t = int(input('Введите число наборов данных (от 1 до 1000)').replace(' ', ''))

if t < 1 or t > 100:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(t):

print('Ввод набора ', i + 1)

n = int(input('Введите число n ( k <= n < 2k )').replace(' ', ''))

k = int(input('Введите число k ( 1 <= k <= 105 )').replace(' ', ''))

if k < 1 or k > 105:

print('Неверный ввод(k)\nЗавершение работы программы')

return 1

elif n < k or n >= 2 \* k:

print('Неверный ввод(n)\nЗавершение работы программы')

return 1

data.append([n, k])

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum2(data=data)

for result in results:

for num in result:

print(num, end=' ')

print()

return

main2()

Задание 3.

Дан набор из 𝑛 (𝑛 всегда равно степени 2) элементов, содержащих все целые числа 0,1,2,…,𝑛−1 единожды.

Найдите 𝑛2 пар элементов таких, что:

Каждый элемент набора принадлежит ровно одной паре. Сумма по всем парам побитового И элементов пары должна быть в точности равна 𝑘. Формально, если 𝑎𝑖 и 𝑏𝑖 — элементы 𝑖-й пары, то должно выполняться ∑𝑖=1𝑛/2𝑎𝑖&𝑏𝑖=𝑘, где & обозначает операцию побитового И. Если существует несколько решений, найдите любое из них. Если решений не существует, выведите −1.

Входные данные  
Каждый тест состоит из нескольких наборов входных данных. Первая строка ввода содержит одно целое число 𝑡 (1≤𝑡≤400) — количество наборов входных данных. Далее следует их описание.

Каждый набор входных данных состоит из одной строки, содержащей два целых числа 𝑛 и 𝑘 (4≤𝑛≤216, 𝑛 — степень 2, 0≤𝑘≤𝑛−1).

Сумма 𝑛 по всем наборам входных данных не превосходит 216. Все наборы в каждом тесте будут различными.

Выходные данные  
Для каждого набора входных данных, если решения не существует, выведите одну строку, содержащую −1.

В противном случае выведите 𝑛2 строк, 𝑖-я их них должна содержать 𝑎𝑖 и 𝑏𝑖 — элементы 𝑖-й пары.

Если существует несколько решений, выведите любое. Порядок пар и элементов в парах не имеет значения.

Пример  
входные данные  
4  
4 0  
4 1  
4 2  
4 3  
выходные данныеСкопировать  
1 2  
0 2  
1 3  
0 1  
2 3  
-1  
Примечание  
В первом наборе входных данных (0&3)+(1&2)=0.

Во втором наборе (0&2)+(1&3)=1.

В третьем наборе (0&1)+(2&3)=2.

В четвертом наборе решения не существует.

Ход решения.

В данной задаче у нас есть 3 интересующих случая:

1. K=0
2. 1<K<N
3. K=N-1

В первом случае нам достаточно вывести все пары: mas[i] и mas[n-i],

Так как сумма побитового И для данной пары = 0, то и сумма всех пар будет равна нулю.

Во втором случае мы первой же парой наберём необходимое нам значение, а оставшиеся элменты выведем по представленной в первом случае схеме.

В третьем случае мы не сможем набрать необходимое количество очков одной парой. Тогда мы соберём первой парой k-1 и второй доберём 1, остальные пары обратим в ноль по описаной в первом случае схеме.

Код решения.

def solutionOfnum3(data):

results = []

for nk in data:

temp = ''

n = nk[0]

k = nk[1]

mas = [i for i in range(0, n)]

if k == 0:

while len(mas) > 0:

temp += str(mas[0]) + ' ' + str(mas[-1]) + '\n'

mas.pop(0)

mas.pop(-1)

elif k < n - 1:

temp += str(mas[0]) + ' ' + str(mas[n - 1 - k]) + '\n'

temp += str(mas[k]) + ' ' + str(mas[n - 1]) + '\n'

mas.pop(0)

mas.pop(k - 1)

mas.pop(n - k - 3)

mas.pop(n - 4)

while len(mas) > 0:

temp += str(mas[0]) + ' ' + str(mas[-1]) + '\n'

mas.pop(0)

mas.pop(-1)

else:

if n > 4:

temp += str(mas[n - 1]) + ' ' + str(mas[n - 2]) + '\n'

temp += str(mas[1]) + ' ' + str(mas[n - 3]) + '\n'

temp += str(mas[0]) + ' ' + str(mas[2]) + '\n'

mas.pop(0)

mas.pop(0)

mas.pop(0)

mas.pop(-1)

mas.pop(-1)

mas.pop(-1)

while len(mas) > 0:

temp += str(mas[0]) + ' ' + str(mas[-1]) + '\n'

mas.pop(0)

mas.pop(-1)

else:

temp = '-1'

results.append(temp)

return results

def main3():

try:

t = int(input('Введите число t наборов данных (1 <= t <= 400)').replace(' ', ''))

if t < 1 or t > 400:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(t):

print('Ввод набора ', i + 1)

n = int(input('Введите число n (n - степень двойки)').replace(' ', ''))

k = int(input('Введите число k (0 <= k <= n-1)').replace(' ', ''))

if n & (n - 1) != 0 or k < 0 or k > n - 1:

print('Неверный ввод\nЗавершение работы программы')

return 1

data.append([n, k])

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum3(data=data)

for result in results:

print(result)

return

main3()

Задание 4.

Вася и Петя играют в одну простую игру. Вася загадал число x от 1 до n, а Петя пытается угадать это число.

Петя может задавать вопросы вида: «Делится ли загаданное число на число y?».

Игра происходит по следующим правилам: вначале Петя спрашивает все вопросы, которые его интересуют (в том числе, он может не задать ни одного вопроса), затем Вася отвечает на каждый из вопросов «да» или «нет». После получения всех ответов Петя должен назвать число, которое загадал Вася.

К сожалению, Петя не слишком хорошо разбирается в теории чисел. Помогите ему найти минимальное количество вопросов, которые он должен задать, чтобы гарантированно угадать число Васи, а также сами числа yi, про которые он должен задать вопросы.

Входные данные  
В единственной строке записано число n (1 ≤ n ≤ 103).

Выходные данные  
Выведите длину искомой последовательности вопросов k (0 ≤ k ≤ n), а затем k чисел — саму последовательность вопросов yi (1 ≤ yi ≤ n).

Если существует несколько корректных последовательностей вопросов минимальной длины, то разрешается вывести любую.

Примеры  
входные данные  
4  
выходные данные  
3  
2 4 3\ входные данные  
6  
выходные данные  
4  
2 4 3 5  
Примечание  
Последовательность из ответа на первый тест из условия действительно корректна.

Если загаданное число не делится ни на одно из чисел последовательности, то оно равно 1.

Если же загаданное число делится на 4, то оно равно 4.

Если загаданное число делится на 3, то загаданное число равно 3.

Иначе, оно равно 2. Стало быть, эта последовательность вопросов действительно угадывает загаданное число. Можно показать, что не существует последовательности вопросов состоящей из менее, чем трёх вопросов, удовлетворяющей условию.

Ход решения.

Чтобы узнать какое число загадал Вася за минимальное количество вопросов мы будем спрашивать делится ли загаданное число на каждое простое число начиная с 2-х до n и на каждое число деляещееся на такое.

Код решения.

def solutionOfnum4(n):

results = []

primes = [1 for i in range(n + 1)]

primes[0] = primes[1] = 0

for i in range(2, n):

if primes[i]:

for j in range(i + i, n + 1, i):

primes[j] = 0

for i in range(2, n):

if primes[i]:

p = 1

while p <= n / i:

p \*= i

results.append(p)

return results

def main4():

try:

n = int(input('Введите число n (1 <= n <= 103)').replace(' ', ''))

if n < 1 or n > 103:

print('Неверный ввод\nЗавершение работы программы')

return 1

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum4(n=n)

print(len(results))

for num in results:

print(num, end=' ')

return

main4()

Задание 5.

Где-то в параллельном Средиземье, когда Саруман поймал и обыскал Фродо, он обнаружил лишь 𝑛 колец. Причем 𝑖-е кольцо было либо золотым, либо серебряным. Для удобства Саруман записал бинарную строку 𝑠 из 𝑛 символов, где 𝑖-м символом был 0, если 𝑖-е кольцо золотое, и 1, если серебряное.

У Сарумана есть волшебная функция 𝑓, которая принимает бинарную строку, а в ответ выдает число, полученное путем перевода строки в двоичное число, а затем перевода двоичного числа в десятичное. Например, 𝑓(001010)=10,𝑓(111)=7,𝑓(11011101)=221.

Саруман, однако, считает, что порядок колец играет какую-то важную роль. Он хочет найти 2 пары целых чисел (𝑙1,𝑟1),(𝑙2,𝑟2), для которых:

1≤𝑙1≤𝑛, 1≤𝑟1≤𝑛, 𝑟1−𝑙1+1≥⌊𝑛2⌋  
1≤𝑙2≤𝑛, 1≤𝑟2≤𝑛, 𝑟2−𝑙2+1≥⌊𝑛2⌋  
Пары (𝑙1,𝑟1) и (𝑙2,𝑟2) различны. То есть, должно выполняться хотя бы одно из условий 𝑙1≠𝑙2 и 𝑟1≠𝑟2.  
Пусть 𝑡 — подстрока 𝑠[𝑙1:𝑟1] строки 𝑠, а 𝑤 — подстрока 𝑠[𝑙2:𝑟2] строки 𝑠. Тогда существует неотрицательное целое число 𝑘, такое, что 𝑓(𝑡)=𝑓(𝑤)⋅𝑘.  
Здесь подстрока 𝑠[𝑙:𝑟] обозначает 𝑠𝑙𝑠𝑙+1…𝑠𝑟−1𝑠𝑟, а ⌊𝑥⌋ обозначает округление числа вниз до ближайшего целого.

Помогите Саруману решить эту задачу! Гарантируется, что при ограничениях задачи существует хотя бы одно решение.

Входные данные  
Каждый тест содержит несколько наборов входных данных.

В первой строке находится одно целое положительное число 𝑡 (1≤𝑡≤103) — количество наборов входных данных. Описание наборов входных данных приведено ниже.

В первой строке каждого набора входных данных находится одно целое положительное число 𝑛 (2≤𝑛≤2⋅104) — количество колец.

Во второй строке каждого набора входных данных находится непустая бинарная строка длины 𝑛.

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

Выходные данные  
Для каждого набора входных данных выведите в отдельной строке четыре числа 𝑙1, 𝑟1, 𝑙2, 𝑟2, которые обозначают начало первой подстроки, конец первой подстроки, начало второй подстроки и конец второй подстроки соответственно.

Если существует несколько правильных вариантов ответа, вы можете вывести любой.

Пример  
входные данные  
7  
6  
101111  
9  
111000111  
8  
10000000  
5  
11011  
6  
001111  
3  
101  
30  
100000000000000100000000000000  
выходные данные  
3 6 1 3  
1 9 4 9  
5 8 1 4  
1 5 3 5  
1 6 2 4  
1 2 2 3  
1 15 16 30  
Примечание В первом наборе входных данных 𝑓(𝑡)=𝑓(1111)=15, 𝑓(𝑤)=𝑓(101)=5.

Во втором наборе входных данных 𝑓(𝑡)=𝑓(111000111)=455, 𝑓(𝑤)=𝑓(000111)=7.

В третьем наборе входных данных 𝑓(𝑡)=𝑓(0000)=0, 𝑓(𝑤)=𝑓(1000)=8.

В четвертом наборе входных данных 𝑓(𝑡)=𝑓(11011)=27, 𝑓(𝑤)=𝑓(011)=3.

В пятом наборе входных данных 𝑓(𝑡)=𝑓(001111)=15, 𝑓(𝑤)=𝑓(011)=3.

Ход решения.

В данной задаче существует 4 интересующих нас ключа:

1. В послеовательности нет едениц
2. Первый ноль в левой части строки
3. Первый ноль по центру
4. Первый ноль в правой части

В первом случае нам достаточно вывести n-1 элементов, из любых участков отрезка.

Во втором случае мы выводим элементы начиная с нуля до конца отрезка и элементы начиная с элемента после нуля и до конца строки. Не тружно заметить, что данные числа будут равны.

Третий случай является редким и обработать его внутри других у меня не вышло. Он возникает в случае ели n % 2 != 0 и ноль является единственным нулём в последовательности. В данном случае в качестве ответа мы приподим всю последовательность в качестве первого отрезка, а в качестве второго отрезка выведем элементы с индекса нуля до конца отрезка.

В последнем случае мы выводим первые элменты до нулевого и первые элементы до нулевого, но без него.

Код решения.

def solutionOfnum5(data):

results = []

for ls in data:

temp = ''

len = ls[0]

string = ls[1]

ind = string.find('0')

if ind == -1:

temp += '1 ' + str(len - 1) + ' 2 ' + str(len)

elif ind < int(len / 2):

temp += str(ind + 1) + ' ' + str(len) + ' ' + str(ind + 2) + ' ' + str(len)

elif ind == int(len / 2) and len % 2 == 1:

temp += '1 ' + str(len) + ' ' + str(ind + 1) + ' ' + str(len)

else:

temp += '1 ' + str(ind) + ' 2 ' + str(ind)

results.append(temp)

return results

def main5():

try:

n = int(input('Введите число t, количество наборов данных (1 <= t <= 103)').replace(' ', ''))

if n < 1 or n > 103:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(n):

length = int(input('Введите количество колец(от 2 до 2\*104)').replace(' ', ''))

if length < 2 or length > 208:

print('Неверный ввод\nЗавершение работы программы')

return 1

rings = input('Введите последовательность колец(из 0 и 1)').replace(' ', '')

if rings.count('0') + rings.count('1') != length:

print('Неверный ввод\nЗавершение работы программы')

return 1

data.append([length, rings])

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum5(data=data)

for result in results:

print(result)

main5()

Задание 6.

Вам задана последовательность 𝑎 длины 𝑛, состоящая из целых чисел от 1 до 𝑛. Среди чисел могут быть одинаковые.

Найдите количество наборов из 𝑚=3 элементов, таких что максимальное число в наборе отличается от минимального не больше, чем на 𝑘=2. Формально, в этой задаче вам нужно найти количество троек индексов 𝑖<𝑗<𝑧, таких что

max(𝑎𝑖,𝑎𝑗,𝑎𝑧)−min(𝑎𝑖,𝑎𝑗,𝑎𝑧)≤2.  
Например, если 𝑛=4 и 𝑎=[1,2,4,3], то существуют две такие тройки (𝑖=1,𝑗=2,𝑧=4 и 𝑖=2,𝑗=3,𝑧=4). Если же 𝑛=4 и 𝑎=[1,1,1,1], то все четыре возможные тройки подходят.

Входные данные  
В первой строке находится одно целое число 𝑡 (1≤𝑡≤2⋅105) — количество наборов входных данных. Далее следуют 𝑡 наборов входных данных.

В первой строке каждого набора входных данных находится целое число 𝑛 (1≤𝑛≤2⋅105) — длина последовательности 𝑎.

В следующей строке находятся 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤𝑛) — последовательность 𝑎.

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

Выходные данные  
Выведите 𝑡 ответов на наборы входных данных. Каждый ответ — это количество упорядоченных троек элементов, таких что максимальное число в тройке отличается от минимального не больше, чем на 2. Обратите внимание, что в отличии от сложной версии задачи, здесь не нужно выводить ответ по какому-либо модулю. Необходимо вывести точное значение ответа.

Пример  
входные данные  
4  
4  
1 2 4 3  
4  
1 1 1 1  
1  
1  
10  
5 6 1 3 2 9 8 1 2 4  
выходные данные  
2  
4  
0  
15

Ход решения.

Так как в данной задаче нам не важен порядок элементов, то мы ег сортируем по возрастанию. Затем мы проходимся по последовательности с помошью отрезка от i-го до mas[i]+2 на каждом отрезке с помощью формулы сочетаний мы подсчитываем их общее количество.

Код решения.

import math

def solutionOfnum6(data):

def find(mas, start, value):

ind = start

for i in range(start, len(mas)):

if mas[i] == value:

return ind

elif mas[i] > value:

return ind

ind += 1

return ind

results = []

for la in data:

ln = la[0]

a = la[1]

a.sort()

indl = 0

count = 0

while indl + 2 < ln:

indr = find(a, indl, a[indl] + 2)

if indr == ln:

indr -= 1

if a[indr] - a[indl] <= 2 and indr - indl >= 2:

count += int(math.factorial(indr - indl + 1) / (math.factorial(3) \* math.factorial(indr - indl - 2)))

if indl != 0 and indr - indl - 1 >= 2:

count -= int(math.factorial(indr - indl) / (math.factorial(3) \* math.factorial(indr - indl - 3)))

indl = find(a, indl, a[indl] + 1)

results.append(count)

return results

def main6():

try:

t = int(input('Введите число t, количество наборов данных (1 <= t <= 103)').replace(' ', ''))

if t < 1 or t > 103:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(t):

n = int(input('Введите длинну последовательности( 1 <= n <= 2\*105)').replace(' ', ''))

if n < 1 or n > 210:

print('Неверный ввод\nЗавершение работы программы')

return 1

a = [int(x) for x in input('Введите последовательность а').split()]

if len(a) != n:

print('Неверный ввод\nЗавершение работы программы')

return 1

data.append([n, a])

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum6(data=data)

for result in results:

print(result)

main6()

Задание 7.

Ану придумала свою собственную функцию 𝑓: 𝑓(𝑥,𝑦)=(𝑥|𝑦)−𝑦, где | обозначает операцию побитового ИЛИ. К примеру, 𝑓(11,6)=(11|6)−6=15−6=9. Можно показать, что для любых неотрицательных чисел 𝑥 и 𝑦 значение 𝑓(𝑥,𝑦) также неотрицательное.

Она хотела бы исследовать данную функцию, и придумала несколько задач для себя. К сожалению, она не может решить все из них, и ей нужна ваша помощь. Вот одна из этих задач.

Значение массива [𝑎1,𝑎2,…,𝑎𝑛] определяется как 𝑓(𝑓(…𝑓(𝑓(𝑎1,𝑎2),𝑎3),…𝑎𝑛−1),𝑎𝑛) (для примеров обратитесь к примечаниям). Вам дан массив из не обязательно различных чисел. Как нужно переставить местами его элементы, чтобы его значение стало максимальным возможным?

Входные данные  
Первая строка содержит одно целое число 𝑛 (1≤𝑛≤105) — длина массива.

Вторая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (0≤𝑎𝑖≤109) — элементы массива. Не гарантируется, что элементы массива различны.

Выходные данные  
Выведите 𝑛 чисел — перестановку массива с максимальным значением. Если существует несколько решений, выведите любое из них.

Примеры  
входные данные  
4  
4 0 11 6  
выходные данные  
11 6 4 0  
входные данные  
1  
13  
выходные данные  
13  
Примечание  
В первом тестовом случае, значение массива [11,6,4,0] равно 𝑓(𝑓(𝑓(11,6),4),0)=𝑓(𝑓(9,4),0)=𝑓(9,0)=9.

[11,4,0,6] также является верным ответом.

Ход решения.

Так как для получения максимального результата нам необходимо из большего вычитать меньиее значение, то в качестве ответа дотаточно вывести отсортированную в обратном порядке последоватеьность.

Код решения.

def solutionOfnum7(a):

results = a[::-1]

return results

def main7():

try:

n = int(input('Введите число n, длинну массива (1 <= n <= 105)').replace(' ', ''))

if n < 1 or n > 105:

print('Неверный ввод\nЗавершение работы программы')

return 1

a = [int(x) for x in input('Введите массив a( 0 <= a[i] <= 109)').split()]

a.sort()

if a[0] < 0 or a[-1] > 109:

print('Неверный ввод\nЗавершение работы программы')

return 1

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum7(a=a)

for result in results:

print(result, end=' ')

main7()

Задание 8.

Магазин целых чисел продаёт 𝑛 отрезков, 𝑖-й из которых состоит из чисел от 𝑙𝑖 до 𝑟𝑖 и стоит 𝑐𝑖 монет.

Завтра Вася отравится в этот магазин и купит несколько отрезков. Он получит все числа, лежащие хотя бы в одном из выбранных отрезков. Количество монет, которое он потратит на покупку равно сумме стоимостей отрезков, которые он купит.

В качестве подарка за покупку, Вася дополнительно получит ещё несколько целых чисел. Число 𝑥, не купленное Васей ранее, он получит тогда и только тогда, когда для 𝑥 выполняются два следующих условия:

Найдётся купленное Васей число 𝑙, меньшее 𝑥.  
Найдётся купленное Васей число 𝑟, большее 𝑥.  
Число 𝑥 Вася может получить в подарок не более одного раза, таким образом все числа, которые получит Вася будут различными.

Например, если Вася купит отрезок [2,4] за 20 монет и отрезок [7,8] за 22 монеты, он потратит 42 монеты и получит числа 2,3,4,7,8 как содержимое отрезков. В качестве подарка он получит числа 5 и 6.

По техническим причинам, завтра в магазине будут доступны к покупке только 𝑠 первых отрезков (то есть отрезки [𝑙1,𝑟1],[𝑙2,𝑟2],…,[𝑙𝑠,𝑟𝑠]). Вася не сможет купить отрезки, не доступные к покупке.

Вася хочет получить (купить или получить в подарок) как можно больше целых чисел. Если это можно сделать разными способами, то он выберет самый дешёвый из них.

Для всех значений 𝑠 от 1 до 𝑛 определите, сколько монет потратит Вася, если к покупке будут доступны только 𝑠 первых отрезков.

Входные данные  
В первой строке задано одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных. Далее следуют описания этих наборов.

В первой строке дано одно число 𝑛 (1≤𝑛≤105) — количество отрезков в магазине.

В следующих 𝑛 строках дано по три числа 𝑙𝑖, 𝑟𝑖, 𝑐𝑖 (1≤𝑙𝑖≤𝑟𝑖≤109,1≤𝑐𝑖≤109) — концы 𝑖-го отрезка и его стоимость.

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

Выходные данные  
Для каждого набора входных данных выведите 𝑛 чисел: 𝑠-е (1≤𝑠≤𝑛) из них должно быть равно количеству монет, которое потратит Вася, если к покупке будут доступны только первые 𝑠 отрезков.

Пример  
входные данные  
3  
2  
2 4 20  
7 8 22  
2  
5 11 42  
5 11 42  
6  
1 4 4  
5 8 9  
7 8 7  
2 10 252  
1 11 271  
1 10 1  
выходные данные  
20  
42  
42  
42  
4  
13  
11  
256  
271  
271  
Примечание  
В первом наборе входных данных при 𝑠=1 Вася может купить только один отрезок [2,4] за 20 монет и получить 3 числа.

Способ получить 7 чисел за 42 монеты при 𝑠=2 описан в условии.

Во втором наборе входных данных обратите внимание на то, что в магазине могут быть одинаковые отрезки.

Ход решения.

Так как нас в первую очередь интересует максимаьная длинна отрезка, то мы учитываем самые крайние значения отрезков и так же запоминаем их отрезков содержащих их. В случае если встречается отрезок который включает одну из крайних координат, но его цена меньше, то запоминаем её как цену за данную величину. Так же стоит учитывать длинну итогового отрезка для обработки случая когда мы встретим отрезок содержащий обе крайние вершины и его покупка будет выгоднее нежели покупка двух отрезков.

Код решения.

import math

def solutionOfnum8(data):

results = []

for pack in data:

res = []

l = math.inf

r = -math.inf

lc = math.inf

rc = math.inf

tc = math.inf

ln = 0

for values in pack:

a = values[0]

b = values[1]

c = values[2]

if a < l:

l = a

lc = c

elif a == l and c < lc:

lc = c

if b > r:

r = b

rc = c

elif b == r and c < rc:

rc = c

if b - a == r - l and c < lc + rc:

ln = b - a

tc = c

if ln == r - l and tc < lc + rc:

res.append(tc)

else:

res.append(lc + rc)

results.append(res)

return results

def main8():

tenpownine = 10\*\*9

try:

t = int(input('Введите число t наборов данных (1 <= t <= 1000)').replace(' ', ''))

if t < 1 or t > 1000:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(t):

print('Ввод набора ', i + 1)

temp = []

n = int(input('Введите число n количество отрезков (1 <= t <= 105)').replace(' ', ''))

for j in range(n):

st = input('Введите числа li ri ci ( 1 <= li <= ri <= 10^9; 1 <= ci <= 10^9 )').split()

if len(st) != 3:

print('Неверное количество введённых данных\nЗавершение работы программы')

return 1

if int(st[0]) < 1 or int(st[0]) > tenpownine or int(st[1]) < int(st[0]) or\

int(st[1]) > tenpownine or int(st[2]) < 1 or int(st[2]) > tenpownine:

print('Неверный ввод\nЗавершение работы программы')

return 1

temp.append([int(st[0]), int(st[1]), int(st[2])])

data.append(temp)

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum8(data=data)

for result in results:

for res in result:

print(res)

main8()

Задание 9.

Предположим, вы стоите на плоскости 𝑋𝑌 в точке (0,0) и хотите попасть в точку (𝑛,𝑛).

Вы можете двигать только в двух направлениях:

вправо, т. е. горизонтально и в направлении увеличения 𝑥 коодинаты, или вверх, т. е. вертикально и в направлении увеличения 𝑦 координаты. Другими словами, ваш путь имеет следующую структуру:

первоначально, вы выбираете: пойти вправо или вверх; далее вы проходите некоторое положительное целое расстояние в выбранном направлении (расстояния можно выбирать независимо); далее вы меняете направление движения (от «вправо» к «вверх», либо от «вверх» к «вправо») и повторяете процесс. Вам не нравится менять свое направление слишком много раз, а потому вы сделаете не более 𝑛−1 изменений направления.

В результате ваш путь будет представлять ломаную от (0,0) к (𝑛,𝑛), состоящую из не более чем 𝑛 отрезков, каждый из которых имеет положительную целую длину, а также вертикальные и горизонтальные отрезки идут по очереди.

Не все пути равны. У вас есть 𝑛 целых чисел 𝑐1,𝑐2,…,𝑐𝑛, где 𝑐𝑖 — это цена 𝑖-го отрезка.

Используя эти цены, можно определить цену пути как сумму длин отрезков этого пути умноженных на их стоимости, т. е. если путь состоит из 𝑘 отрезков (𝑘≤𝑛), то цена пути равна ∑𝑖=1𝑘𝑐𝑖⋅𝑙𝑒𝑛𝑔𝑡ℎ𝑖 (отрезки нумеруются от 1 по 𝑘 в порядке их обхода).

Определите путь минимальной стоимости и выведите его цену.

Входные данные  
В первой строке задано одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных.

В первой строке каждого набора задано одно целое число 𝑛 (2≤𝑛≤105).

Во второй строке каждого набора заданы 𝑛 целых чисел 𝑐1,𝑐2,…,𝑐𝑛 (1≤𝑐𝑖≤109) — цены каждого отрезка.

Гарантируется, что сумма 𝑛 не превосходит 105.

Выходные данные  
Для каждого набора входных данных вы должны вывести минимальную цену пути из (0,0) в (𝑛,𝑛), состоящего из не более 𝑛 чередующихся отрезков.

Пример  
входные данные  
3  
2  
13 88  
3  
2 3 1  
5  
4 3 2 1 4  
выходные данные  
202  
13  
19  
Примечание  
В первом примере из условия, чтобы достигнуть (2,2), нужно сделать хотя бы один поворот, и путь может состоять из 2 отрезков: горизонтального длины 2 и вертикального длины 2. Стоимость равна 2⋅𝑐1+2⋅𝑐2=26+176=202.

Во втором примере можно построить путь из 3 отрезков: длины 1, длины 3 и длины 2. Стоимость равна 1⋅2+3⋅3+2⋅1=13.

В третьем примере можно построить путь из 4 отрезков: длины 1, длины 1, длины 4 и длины 4. Стоимость равна 1⋅4+1⋅3+4⋅2+4⋅1=19.

Ход решения.

В данной задаче мы будем действовать жадно каждый раз перемещаясь вдоль оси на одну еденицу, кроме случая когда стоимость перемещения не будет минимаьна. Данный алгоритм будем проводить для количества отрезков от n до 2. В качестве ответа будет выведена минимальная стоимость перехода.

Код решения.

import math

def solutionOfnum9(data):

results = []

for pack in data:

minc = math.inf

fmas = []

smas = []

sc = len(pack) - 1

for i in range(len(pack)):

if i % 2 == 0:

fmas.append(pack[i])

else:

smas.append(pack[i])

fmin = min(fmas)

index\_fmin = fmas.index(fmin)

smin = min(smas)

index\_smin = smas.index(smin)

while sc > 0:

tempc = fmin \* (len(pack) - (len(fmas) - 1)) + smin \* (len(pack) - (len(smas) - 1))

for i in range(len(fmas)):

if i != index\_fmin:

tempc += fmas[i]

for i in range(len(smas)):

if i != index\_smin:

tempc += smas[i]

if tempc < minc:

minc = tempc

if sc % 2 == 0:

fmas.pop(-1)

if len(fmas) > 0:

fmin = min(fmas)

index\_fmin = fmas.index(fmin)

else:

smas.pop(-1)

if len(smas) > 0:

smin = min(smas)

index\_smin = smas.index(smin)

sc -= 1

results.append(minc)

return results

def main9():

def findMax(mas):

max = -math.inf

for num in mas:

if num > max:

max = num

return max

def findMin(mas):

min = math.inf

for num in mas:

if num < min:

min = num

return min

try:

t = int(input('Введите число t наборов данных (1 <= t <= 1000)').replace(' ', ''))

if t < 1 or t > 1000:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(t):

print('Ввод набора ', i + 1)

temp = []

n = int(input('Введите число n (2 <= n <= 105)').replace(' ', ''))

st = [int(x) for x in input('Введите числа ci ( 1 <= ci <= 109 )').split()]

if findMin(st) < 1 or findMax(st) > 109:

print('Неверный ввод\nЗавершение работы программы')

return 1

data.append(st)

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

results = solutionOfnum9(data=data)

for result in results:

print(result)

main9()

Задание 10.

Необъяснимый народ эти битландцы. Какие же у них странные обычаи!

Как обычно, дядя J. хочет раскрасить n яиц для известного Битландского фестиваля, который называется Битруз. Дядя J. попросил выполнить эту работу G. и A. Дети очень обрадовались. Ведь они знают, что им заплатят за работу.

Всего у дяди J. есть n яиц. Для каждого яйца G. назвал количество денег, которое он требует за покраску этого яйца. Аналогично, для каждого яйца A. назвал количество денег, которое он требует за покраску этого яйца. Оказалось, что для каждого яйца сумма количеств денег, которые требует за покраску A. и G., равна 1000.

Дядя J. хочет поделить яйца между детьми так, чтобы каждое яйцо было отдано на покраску ровно одному ребенку. Также дядя хочет, чтобы сумма денег которая достанется A. отличалась от суммы денег, которая достанется G. не более чем на 500.

Помогите дяде J. Найдите требуемое распределение яиц или скажите, что разделить яйца требуемым образом не получится.

Входные данные  
В первой строке записано целое число n (1 ≤ n ≤ 106) — количество яиц. В каждой из следующих n строк записано два целых числа: ai, gi (0 ≤ ai, gi ≤ 1000; ai + gi = 1000) — количество денег, которое требует A. за покраску i-того яйца, и количество денег, которое требует G. за покраску i-того яйца.

Выходные данные  
Если разделить яйца между детьми требуемым образом нельзя, выведите «-1» (без кавычек).

Иначе выведите строку, состоящую из n символов «G» и «A»: i-тый символ должен обозначать того, кому достанется i-тое яйцо в требуемом распределении (символ «A» обозначает A., символ «G» обозначает G.). Если обозначить сумму денег, которую дядя должен будет отдать A. за покраску, через Sa, а сумму денег, которую дядя должен будет отдать G. за покраску, через Sg, должно выполняться неравенство |Sa - Sg| ≤ 500.

Если существует несколько решений, разрешается вывести любое.

Примеры  
входные данные  
2  
1 999  
999 1  
выходные данные  
AG  
входные данные  
3  
400 600  
400 600  
400 600  
выходные данные  
AGA

Ход решения.

В данной задаче тоже будем действовать жадно. Будем отдавать все яица А, до тех пор пока передача следующего элемента не увеличит размер до более чем 500. В таком случа отдадим элемент G и продолжим действовать по описаному выше алгоритму.

Код решения.

def solutionOfnum10(data):

a = 0

g = 0

res = ''

for pack in data:

f = pack[0]

s = pack[1]

if a + f - g < 500:

res += 'A'

a += f

else:

res += 'G'

g += s

return res

def main10():

try:

n = int(input('Введите число n (1 <= n <= 106)').replace(' ', ''))

if n < 1 or n > 106:

print('Неверный ввод\nЗавершение работы программы')

return 1

data = []

for i in range(n):

pack = [int(x) for x in input('Введите числа ai и gi ( 0 <= ai, gi <= 1000; ai + gi = 1000 )').split()]

if pack[0] < 0 or pack[1] < 0 or pack[0] > 1000 or pack[1] > 1000 or pack[0] + pack[1] != 1000:

print('Неверный ввод\nЗавершение работы программы')

return 1

data.append(pack)

except TypeError:

print('Неверный ввод\nЗавершение работы программы')

return 2

except ValueError:

print('Неверный ввод\nЗавершение работы программы')

return 3

result = solutionOfnum10(data=data)

print(result)

main10()