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: «МКиИТ»

Курсовая работа по СиАОД

Выполнил студент группы БФИ 2001

Шатковский Д.А.

Проверил Симонов С.Е.

Москва 2022

**1 задание**

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.

Код для решения данной задачи представлен в листинге 1.

#Подсчет стоимости сувенира

def bSearch(left,right,money,a):

if (right == left):

if (canBuy(left,money,a)!=-1):

return (left,canBuy(left,money,a))

if (right - 1 == left):

if (canBuy(right,money,a)!=-1):

return (right, canBuy(right,money,a))

if (canBuy(left,money,a)!=-1):

return (left,canBuy(left,money,a))

mid = (right+left)//2

if (canBuy(mid,money,a)!=-1):

return bSearch(mid,right,money,a)

else:

return bSearch(left,mid-1,money,a)

return -1;

def checkMas(arr,l, r):

for i in range(len(arr)):

if not(l <= arr[i] < r):

print("Введенное значение не принадлежит заданному отрезку")

return False

return True

def checkVal(v,l, r):

if l <= v < r:

return True

else:

print("Введенное значение не принадлежит заданному отрезку")

return False

def canBuy(k,m,x):

arr = x[:]

s = 0

for i in range(len(arr)):

arr[i] += k\*(i+1)

arr.sort()

s = sum(arr[:k])

if (s<=m):

return s

return -1

try:

while True:

n, money = input('').split(' ')

n, money = int(n) , int(money)

if checkVal(n,1,10\*\*5+1):

if checkVal(money,1,10\*\*9+1):

break

while True:

arr = list(map(int, input().split()))

if checkMas(arr,1,10\*\*5+1):

break

a,b = bSearch(0,n,money,arr)

print(a, b)

except Exception:

print("Некорректный ввод")

Листинг 1

**2 задание**

Манао изобрел новый математический термин — красивое множество точек. Он называет множество точек на плоскости красивым, если выполняются следующие условия:

Координаты каждой точки множества — целые числа.  
Для любых двух точек из этого множества, расстояние между ними — нецелое число.

Рассмотрим все точки (x, y), удовлетворяющие неравенствам: 0 ≤ x ≤ n; 0 ≤ y ≤ m; x + y > 0. Выберите из них как можно больше точек, так чтобы все выбранные точки образовывали красивое множество.

Входные данные: в единственной строке записаны два разделенных пробелом целых числа n и m (1 ≤ n, m ≤ 100).

Выходные данные: в первой строке выведите целое число — размер найденного красивого множества k. В следующих k строках выведите по паре разделенных пробелом целых чисел — соответственно x- и y- координаты выбранных в множество точек.

Если существует несколько оптимальных решений, разрешается вывести любое.

Код для решения данной задачи представлен в листинге 2.

def minim(a,b):

if a < b:

mini = a

else:

mini = b

#Находим размер

return mini + 1

def podsh(mini, b):

a = []

for i in range(mini + 1):

a.append([b - i, i])

return a

try:

while True:

a, b = input('').split(' ')

a, b = int(a) , int(b)

if checkVal(a,1,101):

if checkVal(b,1,101):

break

minii = minim(a,b)

print(minii)

a = podsh(minii, a)

for i in range(minii):

print(a[i][0], a[i][1])

except Exception:

print("Некорректный ввод")

Листинг 2

**3 задание**

Tsumugi купила 𝑛 вкусных конфет в Light Music Club. Они пронумерованы целыми числами от 1 до 𝑛, у 𝑖-й конфеты концентрация сахара описана целым числом 𝑎𝑖.

Yui любит конфеты, но она может есть не более 𝑚 конфет каждый день, из соображений здоровья.

Дни нумеруются в 1-индексации (пронумерованы 1,2,3,…). Съесть конфету 𝑖 в 𝑑-й день будет стоить (𝑑⋅𝑎𝑖) сахарного штрафа, так как со временем конфеты становятся более сладкими. Каждая конфета может быть съедена не более одного раза.

Итоговый сахарный штраф равен сумме сахарных штрафов всех съеденных конфет.

Предположим, что Yui выбирает ровно 𝑘 конфет, и ест их в любом порядке, в каком захочет. Какой минимальный сахарный штраф она может получить?

Так как Yui нерешительная девушка, она просит ответить вас на этот вопрос для всех 𝑘 от 1 до 𝑛.

Входные данные:

В первой строке записаны два целых числа 𝑛 и 𝑚 (1≤𝑚≤𝑛≤200 000).

Во второй строке записаны 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤200 000).

Выходные данные:

Вы должны вывести 𝑛 целых чисел 𝑥1,𝑥2,…,𝑥𝑛 в отдельной строке, разделяя пробелами, где 𝑥𝑘 это минимальный сахарный штраф который Yui может получить, съев ровно 𝑘 конфет.

Код для решения данной задачи представлен в листинге 3.

def Candy(a):

a = sorted(a)

c = [a[0]]

#добавляем в массив значения

#первый день она съедает конфету с самым маленьким штрафом

#Добавляем в массив суммы значения отсортированного массива и предыдущего элемента

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

c.append(c[-1]+a[i])

#Изменяем элементы массива, начиная с кол-ва конфет, которые может съесть девочка, прибавляя значения с индексом меньше на это кол-во конфет

for i in range(m,len(c)):

c[i] = c[i]+c[i-m]

return c

try:

while True:

n, m = input('').split(' ')

n, m = int(n) , int(m)

if checkVal(n,1,200000):

if checkVal(m,1,200000):

break

while True:

arr = list(map(int, input().split()))

if checkMas(arr,1,200000) & (len(arr) == n):

break

c = Candy(arr)

print(\*c)

except Exception:

print("Некорректный ввод")

Листинг 3

**4 задание**

Есть 𝑛 боксёров, вес 𝑖-го равен 𝑎𝑖. Каждый из них перед соревнованием может изменить свой вес не более чем на 1 (вес не может стать равным нулю, то есть должен остаться положительным). Вес — это всегда целое число.

Необходимо выбрать наибольшую по количеству человек такую команду боксёров, что все веса боксёров в ней — различны.

Напишите программу, которая для заданных текущих значений 𝑎𝑖 найдет максимальное возможное количество боксёров в команде.

Возможно, что после какого-то изменения вес какого-то боксера станет 150001 (но не больше).

Входные данные:

В первой строке задано единственное целое число 𝑛 (1≤𝑛≤150000) — количество боксёров. В следующей строке через единичный пробел заданы 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛, где 𝑎𝑖 (1≤𝑎𝑖≤150000) — вес 𝑖-го боксёра.

Выходные данные:

Выведите единственное целое число — максимальное возможное количество человек в команде.

Код для решения данной задачи представлен в листинге 4.

def boxed(a):

a.sort()

box=set()

for x in a:

if x-1>0 and x-1 not in box:

box.add(x-1) #Добавляем в словарь значение, если подходит по условиям с отниманием одного

elif x not in box:

box.add(x) #Добавляем в словарь значение, если подходит по условиям без отнимания

elif x+1 not in box:

box.add(x+1) #Добавляем в словарь значение, если подходит по условиям с добавлением

return len(box)

try:

while True:

n = int(input())

if checkVal(n,1,150000):

break

while True:

arr = list(map(int, input().split()))

if checkMas(arr,1,200000) & (len(arr) == n):

break

print(boxed(arr))

except Exception:

print("Некорректный ввод")

Листинг 4

**5 задание**

Недавно Пари и Арий узнали про NP-трудные задачи, особенно им понравилась задача о минимальном вершинном покрытии.

Пусть нам дан некоторый граф G. Подмножество A его вершин называется вершинным покрытием, если для любого ребра uv хотя бы один его конец лежит в множестве, то есть выполнено или (или оба условия).

Пари и Арий выиграли на командной олимпиаде чудесный неориентированный граф, и теперь каждый хочет забрать себе множество его вершин, являющееся вершинным покрытием.

Они отдали свой граф вам и попросили выбрать два непересекающихся множества вершин A и B, таких что и A и B являются вершинным покрытием, или определить, что сделать это невозможно. Каждая вершина может быть отдана только одному из друзей (а некоторые и вовсе можно не отдавать никому).

Входные данные:

В первой строке входных данных записаны два числа n и m (2 ≤ n ≤ 100 000, 1 ≤ n, m ≤ 100 000) — количество вершин и количество рёбер в выигранном друзьями графе.

В каждой из следующих m строк записана пара чисел ui и vi (1  ≤  ui,  vi  ≤  n), означающая ненаправленное ребро между вершинами ui и vi. Гарантируется, что в графе отсутствуют петли и кратные рёбра.

Выходные данные:

Если невозможно поделить граф между Пари и Арием, как они этого хотят, то выведите «-1» (без кавычек).

Если же существуют два непересекающихся вершинных покрытия, то выведите их описания.

Каждое описание состоит из двух строк. Первая строка должна содержать единственное число k — количество вершин в данном вершинном покрытии, а вторая строка должна содержать k чисел — индексы вершин в покрытии.

Обратите внимание, что поскольку m ≥ 1, никакое вершинное покрытие не может быть пустым.

Код для решения данной задачи представлен в листинге 5.

from collections import deque

n, m = [int(x) for x in input().split()]

color = [0]\*n

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

for i in range(0, m):

u, v = [int(x) - 1 for x in input().split()]

graph[u].append(v)

graph[v].append(u)

# Двудольный граф

res = True

for i in range(0, n):

if color[i] != 0:

continue

q = deque()

color[i] = 1

q.append(i)

while q:

v = q.popleft()

for a in graph[v]:

if color[a] == 0:

color[a] = -color[v]

q.append(a)

elif color[a] != -color[v]:

res = False

break

if not res:

break

if not res:

print(-1)

else:

s1 = []

s2 = []

for i in range(0, n):

if color[i] == 1:

s1.append(i)

else:

s2.append(i)

print(len(s1))

print(" ".join(str(x + 1) for x in s1))

print(len(s2))

print(" ".join(str(x + 1) for x in s2))

Листинг 5

**6 задание**

Приближается лунный новый год, а Боб решил отправиться на прогулку в ближайший парк.

Парк может быть представлен как связный граф из 𝑛 вершин и 𝑚 неориентированных ребер. Изначально Боб находился в вершине 1 и записал 1 в свою записную книжку. Он может переходить из одной вершины в другую по данным ребрам. Каждый раз, когда он посещает вершину, еще не записанную в его книжку, он записывает ее. После того, как он посетит все вершины как минимум по разу, он закончит прогулку, а в его книжке будет записана перестановка вершин 𝑎1,𝑎2,…,𝑎𝑛.

Гулять скучно, а решать задачи — интересно. Боб хочет узнать лексикографически минимальную перестановку, которую он может получить по итогам прогулки. Бобу эта задача кажется простой, поэтому он отдал ее вам.

Последовательность 𝑥 лексикографически меньше последовательности 𝑦, если и только если выполняется один из следующих пунктов:

𝑥 — префикс 𝑦, но 𝑥≠𝑦 (обратите внимание, в этой задаче такое невозможно, так как все рассматриваемые последовательности имеют одинаковую длину); в первой позиции, где 𝑥 и 𝑦 различны, в последовательности 𝑥 элемент меньше, чем соответствующий элемент в 𝑦. Входные данные Первая строка содержит два целых числа 𝑛 и 𝑚 (1≤𝑛,𝑚≤105) — числе вершин и ребер в графе, соответственно.

Следующие 𝑚 строк описывают неориентированные ребра графа. 𝑖-я из этих строк содержит два целых числа 𝑢𝑖 и 𝑣𝑖 (1≤𝑢𝑖,𝑣𝑖≤𝑛) — вершины, соединенные 𝑖-м ребром.

Обратите внимание, что в графе могут быть кратные ребра и петли. Гарантируется, что граф связный.

Код для решения данной задачи представлен в листинге 6.

n, m = map(int, input().split())

a = [[] for i in range(n)]

pos = [False] \* n

for i in range(m):

u, v = map(int, input().split())

a[u - 1].append(v - 1)

a[v - 1].append(u - 1)

otv = []

q = []

q.append(0)

while len(q) > 0:

k = min(q)

q.remove(k)

pos[k] = True

otv.append(k + 1)

for i in a[k]:

if not pos[i]:

q.append(i)

pos[i] = True

print(\*otv)

Листинг 6

**7 задание**

У AquaMoon есть 𝑛 друзей. Они встали в ряд. 𝑖-й слева друг надел футболку с числом 𝑎𝑖, написанным на ней. Каждый друг смотрит в одну из двух сторон налево или направо. В начале все друзья смотрят направо.

AquaMoon может делать операции с друзьями. На каждой операции AquaMoon может выбрать два соседних в ряду друга и поменять их местами. После каждой операции оба друга изменяют направление, в котором смотрели, на противоположное: если кто-то смотрел налево, он будет смотреть направо, и наоборот.

AquaMoon надеется, что после нескольких операций числа, написанные на футболках у 𝑛 друзей станут неубывающими, если смотреть на них слева направо. Также она хочет, чтобы в этот момент все друзья смотрели направо. Установите, возможно ли это.

Входные данные:

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится единственное целое число 𝑡 (1≤𝑡≤50) — количество наборов входных данных.

В первой строке описания каждого набора входных данных находится единственное целое число 𝑛 (1≤𝑛≤105) — количество друзей Aquamoon.

Во второй строке находится 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤105) — числа, записанные на футболках.

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

Выходные данные:

Для каждого набора входных данных выведите «YES» (без кавычек), если существует возможная последовательность операций; иначе, выведите «NO» (без кавычек).

Вы можете выводить каждую букву в любом регистре (строчную или заглавную).

Код для решения данной задачи представлен в листинге 7.

def Perest():

n = int(input())

a = list(map(int, input().split()))

b = [a[i] for i in range(n)]

f = True

ind1 = {} #Так как работает правило право лево, то при перемещении объектов с четным индексом их возможно перенести только на четные индексы

ind2 = {} #Так как работает правило право лево, то при перемещении объектов с нечетным индексом их возможно перенести только на нечетные индексы

b.sort()

#Заносим четные и нечетные элементы

for i in range(n):

if i % 2 == 0:

if b[i] in ind2:

ind2[b[i]].append(i)

else:

ind2[b[i]] = [i]

else:

if b[i] in ind1:

ind1[b[i]].append(i)

else:

ind1[b[i]] = [i]

#Проверка на соответствие четных и нечетных элементов сортированного массива и не сортированного

for i in range(n):

if i % 2 == 0 and (a[i] in ind2) and len(ind2[a[i]]) > 0:

ind2[a[i]].pop(0)

elif i % 2 == 1 and (a[i] in ind1) and len(ind1[a[i]]) > 0:

ind1[a[i]].pop(0)

else:

f = False

break

if f:

print('YES')

else:

print('NO')

t = int(input())

for i in range(t):

Perest()

Листинг 7

**8 задание**

Паша любит обмениваться со своим другом положительными целыми числами. Паша заботится о своей безопасности, поэтому он шифрует каждое загаданное натуральное число 𝑛 следующим образом. Он заранее выбрал какие-то три целых числа 𝑎, 𝑏 и 𝑐, что 𝑙≤𝑎,𝑏,𝑐≤𝑟. После этого он вычисляет число 𝑚=𝑛⋅𝑎+𝑏−𝑐 и пересылает его своему другу.

В руки злоумышленника попали три значения: 𝑙, 𝑟 и 𝑚. Может ли злоумышленник «восстановить» параметры 𝑎, 𝑏 и 𝑐? Какие это могут быть значения?

Иными словами, найдите любую такую тройку чисел 𝑎, 𝑏 и 𝑐, что:

𝑎, 𝑏 и 𝑐 — целые числа, 𝑙≤𝑎,𝑏,𝑐≤𝑟, существует такое целое положительное 𝑛, что 𝑛⋅𝑎+𝑏−𝑐=𝑚. Входные данные Первая строка теста содержит одно целое число 𝑡 (1≤𝑡≤20) — количество наборов входных данных. Затем следуют 𝑡 наборов входных данных.

Единственная строка набора входных данных содержит три целых числа 𝑙, 𝑟 и 𝑚 (1≤𝑙≤𝑟≤500000, 1≤𝑚≤1010). Эти числа таковы, что ответ на задачу существуют.

Выходные данные:

Для каждого набора входных данных в единственной строке выведите три целых числа 𝑎, 𝑏 и 𝑐 такие, что 𝑙≤𝑎,𝑏,𝑐≤𝑟 и существует такое натуральное число 𝑛, что 𝑛⋅𝑎+𝑏−𝑐=𝑚. Гарантируется, что такие числа существуют. Если подходящих решений несколько, выведите любое из них.

Код для решения данной задачи представлен в листинге 8.

for \_ in range(int(input())):

l,r,m=map(int,input().split())

d=r-l

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

v=m%i

#Проверяем разницу остатка и разницу макс. и мин. значения,

if v<=d and m>=i:

print(i,l+v,l)

break

elif v>=i-d:

print(i,l,l+i-v)

break

Листинг 8

**Задание 9**

𝑛 друзей хотят подарить друг другу подарки на Новый год. Каждый друг должен подарить ровно один подарок и получить ровно один подарок. Друг не может подарить подарок самому себе.

Для каждого друга известно значение 𝑓𝑖: оно или равно 𝑓𝑖=0, если 𝑖-й друг не знает, кому он хочет подарить подарок, или равно 1≤𝑓𝑖≤𝑛, если 𝑖-й друг хочет подарить подарок другу 𝑓𝑖.

Вы хотите заполнить неизвестные значения (𝑓𝑖=0) таким образом, чтобы каждый друг подарил ровно один подарок и получил ровно один подарок, а также не было друга, который дарит подарок сам себе. Гарантируется, что изначальная информация не противоречива.

Если существует несколько возможных ответов, выведите любой.

Входные данные  
Первая строка входных данных содержит одно целое число 𝑛 (2≤𝑛≤2⋅105) — количество друзей.

Вторая строка входных данных содержит 𝑛 целых чисел 𝑓1,𝑓2,…,𝑓𝑛 (0≤𝑓𝑖≤𝑛, 𝑓𝑖≠𝑖, все 𝑓𝑖≠0 различны), где 𝑓𝑖 или равно 𝑓𝑖=0, если 𝑖-й друг не знает, кому он хочет подарить подарок, или равно 1≤𝑓𝑖≤𝑛, если 𝑖-й друг хочет подарить подарок другу 𝑓𝑖. Также гарантируется, что есть хотя бы два значения 𝑓𝑖=0.

Выходные данные  
Выведите 𝑛 целых чисел 𝑛𝑓1,𝑛𝑓2,…,𝑛𝑓𝑛, где 𝑛𝑓𝑖 должно быть равно 𝑓𝑖, если 𝑓𝑖≠0, или номеру друга, которому 𝑖-й друг хочет подарить подарок. Все значения 𝑛𝑓𝑖 должны быть различны, 𝑛𝑓𝑖 не может быть равно 𝑖. Каждый друг должен подарить ровно один подарок и получить ровно один подарок, а также не должно быть друга, который дарит подарок сам себе.

Если существует несколько возможных ответов, выведите любой.

Код для решения данной задачи представлен в листинге 9.

n=int(input())

arr=list(map(int,input().strip().split()))

s=set(arr)

ans=[]

#Заполняем массив друзьями, которых не выбрали

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

if not i in s:

ans.append(i)

p1=0

for i in range(n):

#проверяем по каждому другу, который не выбрал есть ли у него напарник на метке p1, если есть меняем значение на соседние, нет, увеличиваем метку

if arr[i]==0 and ans[p1]==i+1:

ans[p1-1],ans[p1]=ans[p1],ans[p1-1]

p1+=1

continue

if arr[i]==0:

p1+=1

p1=0

for i in range(n):

if arr[i]==0:

arr[i]=ans[p1]

p1+=1

print(\*arr)

Листинг 9

**10 задание**

У вас есть два числа 𝑎 и 𝑏. Вы можете выполнять следующие операции с ними: в качестве первой операции увеличить одно из этих двух чисел на 1; в качестве второй — увеличить на 2, и так далее. Количество таких операций вы выбираете сами.

Например, если 𝑎=1 и 𝑏=3, вы можете сделать следующую последовательность из трех операций: добавить 1 к 𝑎, тогда 𝑎=2 и 𝑏=3;  
добавить 2 к 𝑏, тогда 𝑎=2 и 𝑏=5; добавить 3 к 𝑎, тогда 𝑎=5 и 𝑏=5.  
Вычислите минимальное количество операций, необходимое для того, чтобы сделать числа 𝑎 и 𝑏 равными.

Входные данные:

Первая строка содержит число 𝑡 (1≤𝑡≤100) — количество наборов входных данных.

Каждый набор входных данных содержит два числа 𝑎 и 𝑏 (1≤𝑎,𝑏≤109).

Выходные данные:

На каждый набор входных данных выведите число — минимальное количество операций необходимое для того, чтобы сделать числа 𝑎 и 𝑏 равными.

Код для решения данной задачи представлен в листинге 10.

def ok(d):

temp=0

i = 0

f = True

while f:

i = i + 1

temp += i

#Проверяем счетчик больше ли разности и на четность иъ разности

if(temp >= d and (temp - d) % 2 == 0):

f = False

return i

def shet(a, b):

if a == b:

return('0')

else:

d = abs(a-b)

return ok(d)

try:

while True:

t = int(input())

if checkVal(t, 1, 100):

break

for \_ in range(t):

while True:

a, b = input('').split(' ')

a, b = int(a), int(b)

if checkVal(a,1,10\*\*9):

if checkVal(b,1,10\*\*9):

break

print(shet(a,b))

except Exception:

print("Некорректный ввод”)

Листинг 10

**Вывод**

В ходе выполнения курсовой работы был успешно отработан процесс формализации задачи и ее дальнейшее алгоритмизирование.