**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

Горохов К.И.

Проверил: Симонов Е.С.

Вариант №5

Москва 2022

Задание №1

**Условие задания:**

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.

Примеры

входные данные

3 11

2 3 5

выходные данные

2 11

входные данные

4 100

1 2 5 6

выходные данные

4 54

входные данные

1 7

7

выходные данные

0 0

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 1 будет продемонстрирован результат работы программы.

def BinarySearch(Number\_of\_Souvenirs:int,Budget:int,Price\_of\_souv:list ): # Бинарный поиск для нахождения лучшего варианта покупки

first = 0 #Первое значение =0

last = Number\_of\_Souvenirs #Последнее значене = числу сувениров

while (first <= last):

mid = (first+last)//2 #Ищем середину

if price(mid, Price\_of\_souv) == Budget: #Если цена равна бюджета, то возвращаем значение в середине

return mid

elif price(mid, Price\_of\_souv) < Budget: #Если ниже бюджета, берём следующий элемент, иначе предыдущий

first = mid + 1

else:

last = mid - 1

return first - 1

def price(Budget: int, Price\_of\_souv: int): # Расчёт новой стоимости

ar = []

for i in range(len(Price\_of\_souv)):

p = Price\_of\_souv[i] + (i+1)\*Budget #Расчёт по цены по формуле

ar.append(p) #Добавляем в массив

ar.sort() #Сортируем

return sum(ar[:Budget]) #возвращаем сумму цен относительно бюджета

def input\_values():

print("Введите количество сувениров и бюджет Сахира: ")

Number\_of\_Souvenirs, Budget=map(int, input().split()) #Ввод количества сувениров и бюджета Сахира

check\_type\_int(Number\_of\_Souvenirs) #Проверка количества сувениров на тип int

check\_type\_int(Budget)

check\_value(Number\_of\_Souvenirs, 1, 10, 5) #Проверка выхода за границы ограничений

check\_value(Budget, 1, 10, 9)

print("Введите цены на сувениры: ")

Price\_of\_souv = list(map(int, input().split())) # Ввод цен на сувениры

check\_type\_int\_for\_elems\_in\_list(Price\_of\_souv) # Проверка на ввод int значений элементов list

check\_value\_in\_list(Price\_of\_souv, 1, 10, 9) # Проверка выхода

return Number\_of\_Souvenirs, Budget, Price\_of\_souv

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество сувениров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int, power\_for\_right\_edge: int):

if not (left\_edge <= n <= np.power(right\_edge, power\_for\_right\_edge)): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество сувениров должно принимать значение от 1 до 10^5") # Поднимает исключение ValueError

def check\_type\_int\_for\_elems\_in\_list(n: list):

for elems in range(len(n)):

if not isinstance(n[elems], int):

raise TypeError("Каждая цена сувенира должна быть целым числом")

def check\_value\_in\_list(n: list,left\_edge: int, right\_edge: int, power\_for\_right\_edge: int):

for elems in range(len(n)):

if not (left\_edge <= n[elems] <= np.power(right\_edge,power\_for\_right\_edge)):

raise ValueError("Вся стоимость сувениров должна принимать значение от 1 до 10^9")

if \_\_name\_\_ == '\_\_main\_\_':

start\_time = timeit.default\_timer() # Создание таймера

Number\_of\_Souvenirs, Budget, Price\_of\_souv =input\_values()

temp\_solve = BinarySearch(Number\_of\_Souvenirs, Budget, Price\_of\_souv) # Лучший вариант покупки

print("Выходные данные: ")

print(f"{temp\_solve} {price(temp\_solve,Price\_of\_souv)}") # Расчёт стоимости лучшего варианта покупки

print("Время выполнения программы(sec) :", (timeit.default\_timer() - start\_time)) #Вычисление и вывод общего затраченного времени

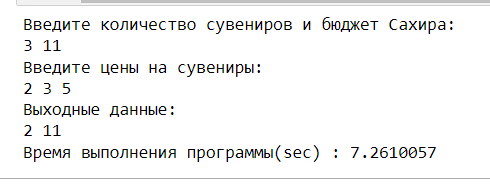


Рисунок 1 – Результат работы программы

Задача 2.

Условие задачи:

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.

Примеры

входные данные

2 2

выходные данные

3

0 1

1 2

2 0

входные данные

4 3

выходные данные

4

0 3

2 1

3 0

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 2 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Граница должна быть целым числом") # Вызывает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Граница должна принимать значение от 1 до 100") # Вызывает исключение ValueError

def input\_type\_and\_values(n: int, m: int):

check\_type\_int(n) #Проверка на целочисленное число

check\_type\_int(m)

check\_value(n, 1, 100) # Проверка на границу ограничений

check\_value(m, 1, 100)

def solution\_and\_output(n: int, m: int):

answer = min(n, m) # Находим минимум среди 2х чисел

size\_of\_set\_of\_numbers = answer+1 #Высчитываем размер множества

print("Выходные данные:\n Размер красивого множества:")

print(size\_of\_set\_of\_numbers)

print("Координаты x и y:")

for elems in range(size\_of\_set\_of\_numbers): #Вывод координат и их расчёт

print(elems, answer - 1)

return elems, answer - 1

if \_\_name\_\_ == '\_\_main\_\_':

start\_time = timeit.default\_timer() # Запуск таймера

print("Входные данные:")

n, m = map(int, input().split()) # Ввод n и m

input\_type\_and\_values(n, m) # Проверка на тип и значения

solution\_and\_output(n,m) # Вывод

print("Время выполнения программы(sec) :", (timeit.default\_timer() - start\_time)) # Окончание и вывод общего времени работы

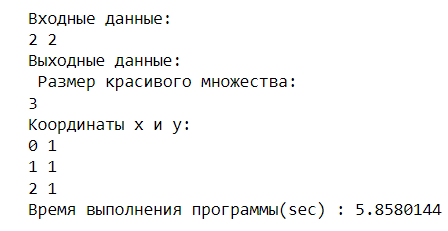


Рисунок 2 – Результат работы второй программы

Задача 3.

**Условие задачи:**

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.

Примеры

входные данные

9 2

6 19 3 4 4 2 6 7 8

выходные данные

2 5 11 18 30 43 62 83 121

входные данные

1 1

7

выходные данные

7

Примечание

Проанилизируем ответ для 𝑘=5 первого примера. Вот один из возможных способов съесть 5 конфет, чтобы минимизировать суммарный сахарный штраф:

День 1: конфеты 1 и 4

День 2: конфеты 5 и 3

День 3 : конфета 6

Итоговый штраф равен 1⋅𝑎1+1⋅𝑎4+2⋅𝑎5+2⋅𝑎3+3⋅𝑎6=6+4+8+6+6=30. Мы можем доказать, что это минимальный возможный сахарный штраф, который Yui может получить если она съест 5 конфет, таким образом 𝑥5=30.

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 3 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Значение n и m должны быть целыми числами") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество конфет должно принимать значение от 1 до 200000") # Поднимает исключение ValueError

def check\_type\_int\_for\_elems\_in\_list(n: list):

for elems in range(len(n)):

if not isinstance(n[elems], int): #Проверка элементов списка на тип данных Int

raise TypeError("Элементы сахарного штрафа должны быть целыми числом") # Вызывает исключение TypeError

def check\_value\_in\_list(n: list, left\_edge: int, right\_edge: int):

for elems in range(len(n)):

if not (left\_edge <= n[elems] <= right\_edge): #Проверка элементов списка на выход из границ ограничений

raise ValueError("Сахарный штраф должен принимать значение от 1 до 200000") #Вызов исключения ValueError

def check\_types\_and\_values\_for\_int\_list(n: int, m: int, x: list):

check\_type\_int(n) # Проверка числа n на тип int

check\_type\_int(m)

check\_value(n, 1, 200000) # Проверка на выход из границы ограничений

check\_value(m, 1, 200000)

check\_type\_int\_for\_elems\_in\_list(x) #Проверка на тип элементов в списке

check\_value\_in\_list(x, 1, 200000) #Проверка значений элементов списка на выход из границ ограничений

def solution\_and\_output(a: list, n: int, m: int):

sugar\_list = a

sorted(sugar\_list)

for elems in range(1, n):

sugar\_list[elems] += sugar\_list[elems - 1] #Вычисление значения сахара от 1 до n

for elems in range(m,n):

sugar\_list[elems] += sugar\_list[elems - m] #Вычисление значения сахара от m до n

return sugar\_list

if \_\_name\_\_ == '\_\_main\_\_':

start\_time = timeit.default\_timer() #Старт таймера

print("Входные данные:\n Введите количество купленных конфет и максимальное количество конфет в день")

n, m = map(int, input().split()) #Ввод 2х чисел n и m

print("Введите значения сахарного штрафа:")

a = list(map(int, input().split())) # Ввод элементов списка

check\_types\_and\_values\_for\_int\_list(n,m,a)

solution\_and\_output(a,n,m)

print("Выходные данные:")

print(\*a) #Вывод элементов листа

print("Время выполнения программы(sec) :", (timeit.default\_timer() - start\_time)) #Окончание таймера и вывод потраченного на решение времени

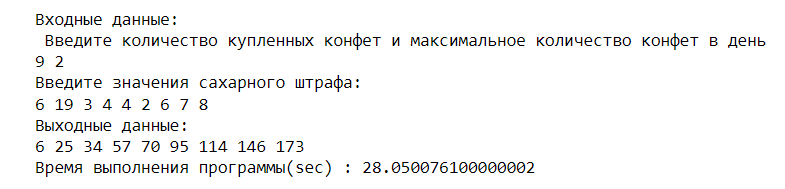


Рисунок 3 – Результат работы 3 программы

Задача 4.

Условие задачи:

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.

Примеры

входные данные

4

3 2 4 1

выходные данные

4

входные данные 6

1 1 1 4 4 4

выходные данные

5

Примечание

В первом примере боксёры не должны менять свои веса — можно просто из всех них составить команду.

Во втором примере можно одного боксёра веса 1 увеличить на единицу (получить вес 2), одного боксёра веса 4 уменьшить на единицу, а другого — увеличить на единицу (получив боксёров с весами 3 и 5 соответственно). Таким образом, можно получить команду, состоящую из боксёров с весами 5,4,3,2,1.

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 4 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество боксёров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество боксёров должно принимать значение от 1 до 150000") # Поднимает исключение ValueError

def check\_type\_int\_for\_elems\_in\_list(n: list):

for elems in range(len(n)):

if not isinstance(n[elems], int): #Проверка элементов списка на тип данных Int

raise TypeError("Вес боксёра должен быть целым числом") # Вызывает исключение TypeError

def check\_value\_in\_list(n: list, left\_edge: int, right\_edge: int):

for elems in range(len(n)):

if not (left\_edge <= n[elems] <= right\_edge): #Проверка элементов списка на выход из границ ограничений

raise ValueError("Вес боксёра должен принимать значение от 1 до 150000") #Вызов исключения ValueError

def check\_types\_and\_values\_for\_int\_list(n: int, x: list):

check\_type\_int(n) # Проверка числа n на тип int

check\_value(n, 1, 150000) # Проверка на выход из границы ограничений

check\_type\_int\_for\_elems\_in\_list(x) #Проверка на тип элементов в списке

check\_value\_in\_list(x, 1, 150000) #Проверка значений элементов списка на выход из границ ограничений

def solution(z: int, list\_weight: list):

last = 0 # счётчик количество человек

for elems in list\_weight:

if last > elems: # если количество человек в команде больше, чем в списке вычитаем 1

z-=1

else:

last = max(elems - 1, last + 1) # выбираем варианта с максимальным значением человек в команде

return z

start\_time = timeit.default\_timer()

print("Входные данные: \n Введите количество боксёров")

z = int(input()) # количество боксёров

list\_weight = list(sorted(map(int, input().split()))) #вес

check\_types\_and\_values\_for\_int\_list(z,list\_weight)

solution(z,list\_weight)

print("Выходные данные: \n Количество человек в команде равно:")

print(z)

print("Время выполнения программы(sec) :", (timeit.default\_timer() - start\_time)) #Окончание таймера и вывод потраченного на решение времени

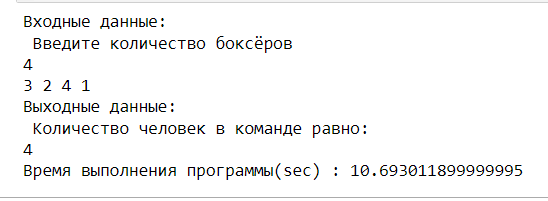


Рисунок 4 – Результат работы 4 программы

Задача 5.

Условие задачи:

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.

Примеры

входные данные

4 2

1 2

2 3

выходные данные

1

2

2

1 3\ входные данные

3 3

1 2

2 3

1 3

выходные данные

-1

Примечание

В первом примере можно отдать Арию вершину номер 2, а Пари вершины с номерами 1 и 3. Вершину 4 можно оставить себе (а можно тоже кому-нибудь отдать).

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 5 будет продемонстрирован результат работы программы.

class Graph:

def \_\_init\_\_(self): # Инициализация переменных и функции load

self.numVertices = None # Инициализация вершин

self.numEdges = None # Инициализация краёв

self.neighbors = None # Инициализация соседних вершин

self.loadGraph()

def loadGraph(self): # Функция создания графа

print("Входные данные: \n Введите количество вершин и количество рёбер")

self.numVertices, self.numEdges = map(int, input().split()) # Ввод количества вершин и рёбер

if not isinstance(self.numVertices, int): # Проверка вершин на соответствие типу данных int

raise TypeError("Количество вершин должно быть целым числом") # Поднимает исключение TypeError

if not isinstance(self.numEdges, int):

raise TypeError("Количество рёбер должно быть целым числом")

if not (2 <= self.numVertices <= 100000): # Проверка вершин на соответствие вхождения в границы допустимых значений

raise ValueError("Значение вершины должно принимать значение от 1 до 100000") # Поднимает исключение ValueError

if not (2 <= self.numEdges <= 100000):

raise ValueError("Значение рёбер должно принимать значение от 1 до 100000")

self.numVertices += 1 # Счётчик

self.neighbors = tuple(list() for vertex in range(self.numVertices)) # Заполнение элементов в черепашку

self.groups = (set(), set())

for edge in range(self.numEdges):

vertex1, vertex2 = map(int, input().split()) #Ввод ненаправленных рёбер вершин

self.neighbors[vertex1].append(vertex2) #Создание вершин

self.neighbors[vertex2].append(vertex1)

def isBipartite(self): # Проверка на двудольность графа

return all(self.isBipartiteComponent(vertex) for vertex in range(1, self.numVertices)

if self.groupOf(vertex) == None)

def isBipartiteComponent(self, vertex): # Проверка на часть двудольного графа

if len(self.neighbors[vertex]) == 0:

return True

self.groups[0].add(vertex) # Объединение вершин в группу

queue = [vertex]

while queue:

vertex = queue.pop() # возвращаем значение очереди

groupOfVertex = self.groupOf(vertex)

for neighbor in self.neighbors[vertex]:

groupOfNeighbor = self.groupOf(neighbor)

if groupOfNeighbor == groupOfVertex: #Если группа вершин равна группе ненаправленных рёбер, вернём false

return False

elif groupOfNeighbor == None: #очищаем

self.groups[not groupOfVertex].add(neighbor)

queue.append(neighbor)

return True

def groupOf(self, vertex): #функция группировки ненаправленных рёбер

return 0 if vertex in self.groups[0] else 1 if vertex in self.groups[1] else None

def solve():

ans = 0

graph = Graph() #инициализация графа

if graph.isBipartite(): #проверка на двудольность графа

for group in graph.groups: #вывод результатов на экран

count\_vertex = len(group)

#print(count\_vertex)

index\_of\_vertex = " ".join(map(str, group))

#print(index\_of\_vertex)

else:

ans = -1

return count\_vertex,index\_of\_vertex,ans

ans = 0

count\_vertex, index\_of\_vertex, ans = solve()

if (ans == -1):

print("Невозможно поделить графы")

else:

print("Выходные данные:")

print(count\_vertex)

print(index\_of\_vertex)

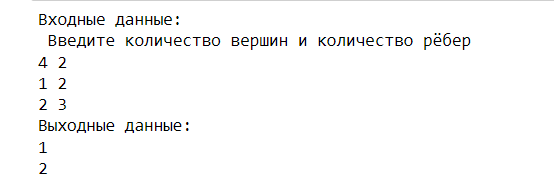


Рисунок 5 – Результат работы 5 программы

Задача 6.

Условие задачи:

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.

Выходные данные

Выведите лексикографически наименьшую последовательность 𝑎1,𝑎2,…,𝑎𝑛 из тех, которые может записать Боб.

Примеры

входные данные

3 2

1 2

1 3

выходные данные

1 2 3

входные данные

5 5

1 4

3 4

5 4

3 2

1 5

выходные данные

1 4 3 2 5

входные данные

10 10

1 4

6 8

2 5

3 7

9 4

5 6

3 4

8 10

8 9

1 10

выходные данные

1 4 3 7 9 8 6 5 2 10

Примечание

В первом примере одним из возможных путей Боба является путь 1→2→1→3. Боб в этом случае запишет последовательность {1,2,3}, которая является лексикографически наименьшей.

Во втором примере Боб может пойти по пути 1→4→3→2→3→4→1→5. Тогда он запишет последовательность {1,4,3,2,5}, которая является лексикографически наименьшей.

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 6 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество боксёров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество боксёров должно принимать значение от 1 до 150000") # Поднимает исключение ValueError

def check\_types\_and\_values\_for\_int(a: int,b: int ):

check\_type\_int(a) # Проверка числа n на тип int

check\_value(a, 1, 105) # Проверка на выход из границы ограничений

check\_type\_int(b)

check\_value(b, 1, 105)

def solution(a: int, b: int):

print("Введите описание неорентированного графа")

for \_ in range(b):

d, e = list(map(int, input().split())) # Ввод значений для ui vi

path.setdefault(d, []).append(e) #Добавление этих значений в path

path.setdefault(e, []).append(d)

ans = []

visited = set()

ins = [1]

while ins:

u = heapq.heappop(ins) #возвращает и удаляет наименьший элемент из кучи heap

if u not in visited: #Если не зашли на u

ans.append(u) #Добавляем к ответу

visited.add(u) # Пометка, что мы посетили данный путь

for j in path[u]:

heapq.heappush(ins, j) # Добавляем элемент j в кучу ins

return ans

if \_\_name\_\_ == '\_\_main\_\_':

print("Входные данные: \nВведите число вершин и рёбер через пробел")

a, b = list(map(int, input().split())) # Ввод чисел вершин и рёбер

path = {} # создание пути

check\_types\_and\_values\_for\_int(a,b) # Проверка на соответствие типу int

ans = solution(a,b) # вычисление ответа

print("Выходные данные: \n Наименьшая последовательность:")

print(\*ans)

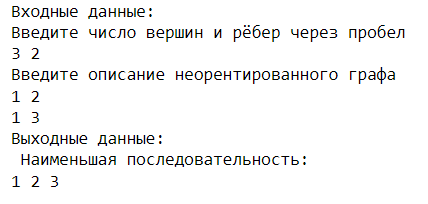


Рисунок 6 – Результат работы 6 программы

Задача 7.

Условие задачи:

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).

Пример

входные данные

3

4

4 3 2 5

4

3 3 2 2

5

1 2 3 5 4

выходные данные

YES

YES

NO

Примечание

Возможная последовательность операций для первого набора входных данных:

Поменять 𝑎1 и 𝑎2. В результате последовательность чисел 3,4,2,5. Друзья смотрят: налево, налево, направо, направо.

Поменять 𝑎2 и 𝑎3. В результате последовательность чисел 3,2,4,5. Друзья смотрят: налево, налево, направо, направо.

Поменять 𝑎1 и 𝑎2. В результате последовательность чисел 2,3,4,5. Друзья смотрят: направо, направо, направо, направо.\

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 7 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество друзей должно быть целым числом") # Поднимает исключение TypeError

def check\_type\_int\_for\_elems\_in\_list(n: list):

for elems in range(len(n)):

if not isinstance(n[elems], int): #Проверка элементов списка на тип данных Int

raise TypeError("Числа на футболке должны быть целыми числами") # Вызывает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество друзей должно принимать значение от 1 до 50") # Поднимает исключение ValueError

def check\_value\_in\_list(n: list, left\_edge: int, right\_edge: int):

for elems in range(len(n)):

if not (left\_edge <= n[elems] <= right\_edge): #Проверка элементов списка на выход из границ ограничений

raise ValueError("Числа на футболке должны принимать значение от 1 до 105") #Вызов исключения ValueError

def solution(a: list,t: int):

for \_ in range(t):

if sorted(a[::2])==sorted(a)[::2]: #Если в сортируемом списке, каждый второй элемент равен каждому второму

return True #элементу в отсортированном списке, вернуть true

else:

return False

if \_\_name\_\_ == '\_\_main\_\_':

print("Входные данные:\n Введите количества набора данных")

t = int(input()) # Ввод количества набора данных

check\_type\_int(t) # Проверка на тип int

check\_value(t,1,50) #Проверка выхода за границу ограничений

for \_ in range(t):

print("Введите количество друзей:")

n = int(input()) #Ввод количества друзей

check\_type\_int(n)

check\_value(n,1,105)

print("Введите числа на футболке:")

a = list(map(int,input().split())) # Ввод количества чисел на футболке

check\_type\_int\_for\_elems\_in\_list(a) #Проверка элементов списка на тип данных int

check\_value\_in\_list(a,1,105) #Проверка элементов списка на выход из границ ограничений

if (solution(a,t)): # Если возвращаемое значение true, то выводим да, иначе нет

print('YES')

else:

print('NO')

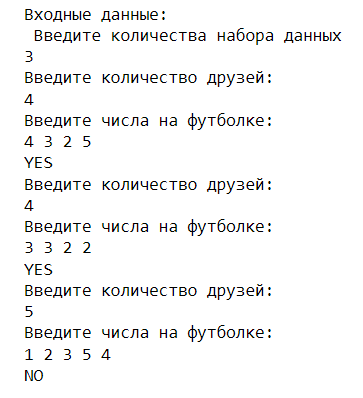


Рисунок 7 – Результат работы 7 программы

Задача 8.

Условие задачи:

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.

Пример

входные данные

2

4 6 13

2 3 1

выходные данные

4 6 5

2 2 3

Примечание

В первом примере было загадано число 𝑛=3, тогда 𝑛⋅4+6−5=13=𝑚. Так же возможны такие ответы: 𝑎=4, 𝑏=5, 𝑐=4 (при этом 𝑛=3); 𝑎=5, 𝑏=4, 𝑐=6 (при этом 𝑛=3); 𝑎=6, 𝑏=6, 𝑐=5 (при этом 𝑛=2); 𝑎=6, 𝑏=5, 𝑐=4 (при этом 𝑛=2).

Во втором примере было загадано число 𝑛=1, тогда 𝑛⋅2+2−3=1=𝑚. Число 𝑛=0 не могло быть загадано, так как число 𝑛 обязательно должно быть натуральным.

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 8 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество боксёров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество боксёров должно принимать значение от 1 до 150000") # Поднимает исключение ValueError

def solution(l: int, r: int, m: int, res: int, res2: int):

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

n = res2 // i #Находим середину

if (n\* i < res or n\*1>res2): #Если не выходит за рамки, продолжаем

continue

flag = n \* i - m #Высчитываем флаг

if flag >= 0:

return (i,l,l+flag) #Возвращаем значения тройки чисел

else:

return (i,r,r+flag) #Если условие не выполняется возвращаем значения и выходим из цикла

break

if \_\_name\_\_ == '\_\_main\_\_':

print("Входные данные: \nВведите количество набора данных")

t = int(input()) # Ввод набора данных

check\_type\_int(t) #Проверка на тип int

check\_value(t,1,20) # Проверка на выход из границ ограничений

for \_ in range(t):

print("Введите тройку чисел:")

l, r, m = map(int, input().strip().split(" ")) #Ввод 3ки чисел

res1 = m - r + l #Рассчёт по данной в условии формуле

res2 = m + r - l

ans1,ans2,ans3 = solution(l,r,m,res1,res2) #Получаем значения для ответа

print("Выходные данные:")

print(ans1,ans2,ans3)

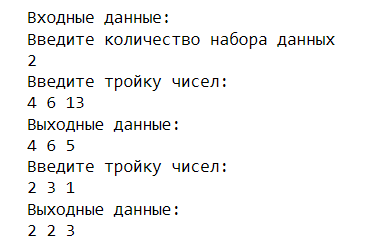


Рисунок 8 – Результат работы 8 программы

Задача 9.

Условие задачи:

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.

Примеры

входные данные

5

5 0 0 2 4

выходные данные

5 3 1 2 4

входные данные

7

7 0 0 1 4 0 6

выходные данные

7 3 2 1 4 5 6

входные данные

7

7 4 0 3 0 5 1

выходные данные

7 4 2 3 6 5 1

входные данные

5

2 1 0 0 0

выходные данные

2 1 4 5 3 \

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 1 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество боксёров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int,power\_for\_right\_edge):

if not (left\_edge <= n <= 2\*np.power(right\_edge,power\_for\_right\_edge)): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество боксёров должно принимать значение от 1 до 150000") # Поднимает исключение ValueError

def solution(zero: int, x:list ,f: list):

for a, b in zip(zero, x): #Создание итератора для объединения zero и x

f[a] = b # Заносим значение b

for i in range(len(zero) - 1):

a = zero[i]

if f[a] == a + 1: #Проверяем элемент внутри f со следующим элементом а

f[a] = f[zero[i + 1]]

f[zero[i + 1]] = a + 1

a = zero[-1] #После прохождений всех итераций вносим в а новое значение.

if f[a] == a + 1:

f[a] = f[zero[0]]

f[zero[0]] = a + 1 #Для последнего числа делаем отдельную проверку и получаем ответ

ans = ' '.join(map(str, f)) #присоединяем ответ к новой переменной

return ans

if \_\_name\_\_=='\_\_main\_\_':

print("Входные данные: \nВведите количество друзей:")

n = int(input()) #Ввод n

check\_type\_int(n) # Проверка на тип int

check\_value(n,2,10,5) # Проверка на выход за границы ограничений

print("Введите номера друзей,где 0 если друг не знает о подарке:")

f = list(map(int, input().split())) #Ввод списка f

f\_x = set(f) #Создаём множество элементов f

s = set([i for i in range(1, n + 1)]) #Создаём ещё одно множество элементов

x = s - f\_x #Получаем значение для x

zero = [i for i, a in enumerate(f) if a == 0] #Вычисляем значения для zero

x = list(x) #Создаём из x список

ans=solution(zero,x,f) #Решение задачи

print("Выходные данные:\nНомера друзей:")

print(ans)

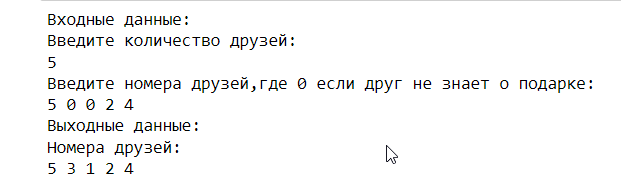


Рисунок 9 – Результат работы 9 программы

Задание 10.

Условие задачи:

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.

Пример

входные данные

3

1 3

11 11

30 20

выходные данные

3

0

4

Примечание

Первый набор входных данных разобран в условии.

Во втором наборе входных данных числа 𝑎 и 𝑏 равны изначально, а значит вам вообще не нужно выполнять операций.

В третьем наборе входных данных вам нужно применить первую, вторую, третью и четвертую операции к числу 𝑏 (𝑏 превратится в 20+1+2+3+4=30).

**Решение:** Ниже будет продемонстрировано решение этой задачи разработанное на языке Phyton в Jupyter Notebook. На рисунке 10 будет продемонстрирован результат работы программы.

def check\_type\_int (n: int):

if not isinstance(n, int): # Проверка n на соответствие типу данных int

raise TypeError("Количество боксёров должно быть целым числом") # Поднимает исключение TypeError

def check\_value(n: int, left\_edge: int, right\_edge: int):

if not (left\_edge <= n <= right\_edge): # Проверка n на соответствие вхождения в границы допустимых значений

raise ValueError("Количество боксёров должно принимать значение от 1 до 150000") # Поднимает исключение ValueError

def check\_type\_int\_for\_elems\_in\_list(n: list):

for elems in range(len(n)):

if not isinstance(n[elems], int): #Проверка элементов списка на тип данных Int

raise TypeError("Вес боксёра должен быть целым числом") # Вызывает исключение TypeError

def check\_value\_in\_list(n: list, left\_edge: int, right\_edge: int,power\_for\_right\_edge: int):

for elems in range(len(n)):

if not (left\_edge <= n[elems] <= math.pow(right\_edge,power\_for\_right\_edge)): #Проверка элементов списка на выход из границ ограничений

raise ValueError("Вес боксёра должен принимать значение от 1 до 150000") #Вызо

def solution(t: int,a:list,b:list ):

n=math.floor((-1+math.sqrt(1+8\*(b-a)))/2) #Высчитываем наибольшее число

a=a+n\*(n+1)/2 #Высчитываем переменную а

while ((a-b)%2)!=0 or a<b: # Пока выполняется условие

n+=1 # n - счётчик

a+=n #Прибавляем к а n

return n

if \_\_name\_\_ =='\_\_main\_\_':

print("Входные данные: \nВведите количество наборов данных")

temp = int(input()) #Вводим количество наборов данныз

check\_type\_int(temp) # Проверяем на int значение

check\_value(temp,1,100) #Проверяем на выход за рамки ограничений

for \_ in range(temp):

print("Введите число а и b:")

[a, b] = sorted(list(map(int, input().split()))) #Вводим и сортируем список из int элемегтов

n = solution(temp,a,b) #В переменную n вносим значения возвращаемые из функции

print("Выходные данные:")

print(n)

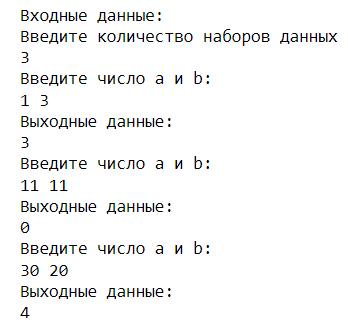


Рисунок 10 – Результат работы 10 программы.

Вывод:

В ходе данной курсовой работы были выполнены 10 задач 5 варианта. Также были реализованы функции проверки ввода, проверки на выход за границы ограничения и тп.