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»

Курсовая работа

Вариант № 12

Выполнил:

студент группы БФИ2001

Левин А.С

Проверил: Симонов С.Е

Москва 2022

**Оглавление**

[**Задача 1 3**](#_Toc103117590)

[**Задача 2 4**](#_Toc103117591)

[**Задача 3 5**](#_Toc103117592)

[**Задача 4 7**](#_Toc103117593)

[**Задача 5 8**](#_Toc103117594)

[**Задача 6 9**](#_Toc103117595)

[**Задача 7 10**](#_Toc103117596)

[**Задача 8 12**](#_Toc103117597)

[**Задача 9 13**](#_Toc103117598)

[**Задача 10 14**](#_Toc103117599)

[**Вывод 16**](#_Toc103117600)

## Задача 1

На уроке геометрии Вася узнал, что треугольник называется прямоугольным, если он невырожденный, а один из его углов прямой. Вася тут же захотел нарисовать такой треугольник: он отметил на листке бумаги 3 точки с целочисленными координатами, соединил их отрезками прямых и показал Пете. Но тот сказал, что васин треугольник не прямоугольный, а почти прямоугольный: сам треугольник не является прямоугольным, но можно подвинуть одну из точек на расстояние ровно 1 так, чтобы координаты остались целыми, и треугольник стал прямоугольным. Вася просит вас помочь ему выяснить, не обманывает ли его Петя. По координатам вершин треугольника определите, является ли он прямоугольным, почти прямоугольным, или же не является ни тем, ни другим.

Входные данные В первой строке входного файла через пробел записано 6 целых чисел x1, y1, x2, y2, x3, y3 — координаты вершин треугольника. Все координаты целые и не превосходят по модулю 100. Гарантируется, что треугольник невырожденный, т. е. его площадь не равна нулю.

Выходные данные Если заданный треугольник является прямоугольным, выведите RIGHT, если он является почти прямоугольным, выведите ALMOST, и если он не является ни тем, ни другим, выведите NEITHER.

**Листинг и Ход работы**

|  |
| --- |
| def rt(p):#метод для вычиление того,является ли треугольник прямоугольным (теорема пифагора) |
| a = (p[0] - p[2]) \*\* 2 + (p[1] - p[3]) \*\* 2 |
| b = (p[0] - p[4]) \*\* 2 + (p[1] - p[5]) \*\* 2 |
| c = (p[2] - p[4]) \*\* 2 + (p[3] - p[5]) \*\* 2 |
| return a + b == c or a + c == b or b + c == a#возвращает является ли треугольтнк прямоугльным |
|  |
|  |
| def f(p): |
| if rt(p):#если треугольник прямоугльный |
| return 'RIGHT' |
| for i in range(6):#сдвигаем все точки на один |
| p[i] -= 1 |
| if rt(p):#если треугольник прямоугльный после сдвига точек |
| return 'ALMOST' |
| return 'NEITHER'#если треугольник не является ни тем ,ни другим |
| print(f(list(map(int, input().split()))))#считываем данные,отправляем в метод f |

**Задача 2**

После ужасающего лесного пожара в Берляндии была реализована программа восстановления леса, по которой были посажены N рядов по M деревьев в каждом, причем настолько ровно, что можно ввести систему координат, в которой j-тое дерево в i-м ряду будет иметь координаты (i, j). Но случилось ужасное, и едва окрепший лес загорелся. И теперь для составления плана эвакуации необходимо найти координаты дерева, которое загорится позже всех.

Возгорание произошло в K точках одновременно, т.е. в начальный момент времени загорелось K деревьев. Каждую минуту огонь перекидывается со всех горящих деревьев на те деревья, которые еще не горят и расстояние от которых до ближайшего горящего дерева равно 1.

Найдите дерево, которое загорится позже всех. Если таких деревьев несколько, выведите любое.

Входные данные В первой строке входного файла заданы два целых числа N, M (1 ≤ N, M ≤ 2000) — размеры леса. Деревья были посажены во всех точках вида (x, y) (1 ≤ x ≤ N, 1 ≤ y ≤ M), x и y — целые числа.

Во второй строке задано одно целое число K (1 ≤ K ≤ 10) — количество деревьев, горящих в начальный момент времени.

В третьей строке задано K пар целых чисел: x1, y1, x2, y2, ..., xk, yk (1 ≤ xi ≤ N, 1 ≤ yi ≤ M) — координаты точек возгорания. Гарантируется, что все точки возгорания различны.

Выходные данные Выведите одну строку, содержащую два целых числа x и y, записанные через пробел, — координаты дерева, которое загорится последним. Если таких деревьев несколько, выведите любое из них.

**Листинг и Ход работы**

|  |
| --- |
| #считывание введеных данных |
| n,m=map(int,input().split()) |
| i=int(input()) |
| l=list(map(int,input().split())) |
| t=[] |
| for x in range(i): |
| t.append([l[x\*2],l[x\*2+1]]) |
| #создаем необходимые для вычисления элементы |
| ans=0 |
| pans=[1,1] |
| for x in range(1,n+1): |
| for y in range(1,m+1): |
| tt=10\*\*7 #максимальное по условию значение |
| for ax in t: |
| tt=min(tt,abs(x-ax[0])+abs(y-ax[1])) |
| if tt>ans: |
| ans=tt |
| pans=[x,y] |
| print(\*pans)#Вывод результата |

**Задача 3**

Вы, наверное, слышали про задачу о дележе торта. Если два человека хотят справедливо разделить кусок торта, то один человек должен разрезать кусок пополам, а другой — выбрать, кому достанется какая половина. У Алисы и Боба есть несколько кусочков пирога, и, вместо того, чтобы разрезать каждый кусочек пополам, они договорились, что каждый кусочек будет съеден одним человеком.

Они решили разделить кусочки следующим образом. Сначала они определили порядок, в котором кусочки будут распределены. У них есть специальная фишка, называемая «решающий», изначально она находится у Боба. Пока не все кусочки распределены, тот, у кого находится фишка, дает очередной кусочек пирога по своему усмотрению одному участнику дележа, а фишку — другому. Так продолжается, пока все кусочки не будут распределены.

Так как все кусочки превосходного качества, каждый участник хочет максимизировать суммарный размер кусочков пирога, которые достанутся ему. Предполагая, что оба участника принимают решения оптимально, сколько пирога получит каждый?

Входные данные В первой строке находится одно целое число N (1 ≤ N ≤ 50) — количество кусочков пирога.

На следующей строке находятся N целых чисел — размеры кусочков (каждый размер лежит в пределах от 1 до 100000 включительно) в том порядке, в котором они будут распределены.

Выходные данные Выведите два целых числа. Сначала выведите суммарный размер кусочков, которые достанутся Алисе, а затем суммарный размер кусочков, которые достанутся Бобу, если оба участника действуют оптимально.

**Листинг и Ход работы**

|  |
| --- |
| #считывание введеных данных |
| n = int(input()) |
| nums = list(map(int,input().split())) |
| b = [0]\*n # массив нулей |
| dp = [0]\*n# массив нулей |
| b[-1]=nums[-1] #последнему элементу массива b присвоить полсдений элемент массива nums |
| dp[-1]=nums[-1]#последнему элементу массива db присвоить полсдений элемент массива nums |
| for i in range(n-2,-1,-1):#пролистываем массив в обратном порядке |
| dp[i]=dp[i+1]+nums[i] |
| b[i]=max(b[i+1],dp[i]-b[i+1]) |
| print(dp[0]-b[0],b[0])#Вывод результата |

**Задача 4**

Мальчик Лёша любит слушать музыку со своего смартфона. Но памяти у смартфона не очень много, поэтому Лёша слушает любимые композиции в известной социальной сети НаСвязи.

К сожалению, интернет в Екатеринозаводске не очень быстрый, и песня загружается очень медленно. Но Лёша весьма нетерпелив. Песня длится ровно T секунд. Лёша прогружает первые S секунд композиции и включает песню. Когда воспроизведение доходит до момента, который ещё не прогрузился, Лёша моментально включает песню заново (при этом загруженная часть песни остаётся в памяти телефона, и процесс скачивания продолжается с того же места), и так происходит до тех пор, когда песня скачается полностью и Лёша дослушает до конца. За q секунд реального времени интернет позволяет скачать q - 1 секунду трека.

Подскажите Лёше, сколько раз он запустит песню, включая самый первый запуск.

Входные данные В единственной строчке записаны три целых числа T, S, q (2 ≤ q ≤ 104, 1 ≤ S < T ≤ 105).

Выходные данные Выведите одно целое число — количество запусков композиции.

**Листинг и Ход работы**

|  |
| --- |
| t, s, q = map(int,input().split()) #считывание введеных данных |
| count = 1 #счетчик |
| s \*= q #загружено изначально |
| while t > s: #пока длительность больше длительности загруженной части песни |
| s \*= q #считаем сколько загрузилось в момент прослушивания |
| count += 1 #добавляем запуск к счетчику |
| print(count)#Вывод результата |

**Задача 5**

Спортсмен Вася участвует в соревнованиях по спортивному ориентированию. На соревновательной прямой расположены n контрольных пунктов с координатами x1, x2, ..., xn. Вася стартует из точки с координатой a. Ему необходимо посетить как минимум n - 1 контрольный пункт, чтобы финишировать. Участники могут посещать контрольные пункты в произвольном порядке.

Васю интересует, какие контрольные пункты и в каком порядке ему нужно посещать, чтобы пройденное им расстояние было минимально возможным. Помогите ему вычислить это расстояние.

Входные данные В первой строке входных данных даны два целых числа n и a (1 ≤ n ≤ 100 000,  - 1 000 000 ≤ a ≤ 1 000 000) — количество контрольных пунктов и начальная координата Васи соответственно.

Во второй строке записаны n целых чисел x1, x2, ..., xn ( - 1 000 000 ≤ xi ≤ 1 000 000) — координаты контрольных пунктов.

Выходные данные Выведите одно целое число — минимальное суммарное расстояние, которое нужно пройти Васе, чтобы посетить как минимум n - 1 контрольный пункт.

**Листинг и Ход работы**

|  |
| --- |
| def f(a, l, r): |
| dl, dr = a - l, r - a#создаем переменные для вычисления минимального суммарного расстояния |
| return 2 \* max(0, min(dl, dr)) + max(dl, dr)#возвращаем минимальное суммарное расстояние |
|  |
| #считывание введеных данных |
| n, a = map(int, input().split()) |
| x = sorted(map(int, input().split())) |
| print(min(f(a, x[0], x[-2]), f(a, x[1], x[-1])))#Вывод результата |

## Задача 6

Иннокентий очень любит чай и сегодня хочет выпить ровно n чашек чая. Он бы выпил и больше, но у него осталось ровно n пакетиков чая, a из которых — зеленый чай и b — черный.

Иннокентий не любит пить более, чем k раз подряд один и тот же чай (зеленый или черный). Перед вами стоит задача определить порядок заваривания пакетиков, при котором Иннокентий сможет выпить n чашек чая, при этом он не будет пить один и тот же чай более k раз подряд, либо сообщить, что это невозможно. Каждый пакетик должен быть заварен ровно один раз.

Входные данные В первой строке следуют четыре целых числа n, k, a и b (1 ≤ k ≤ n ≤ 105, 0 ≤ a, b ≤ n) — количество чашек чая, которые хочет выпить Иннокентий, максимальное количество чашек одного и того же чая, которые Иннокентий может выпить подряд, количество пакетиков с зеленым чаем и количество пакетиков с черным чаем. Гарантируется, что a + b = n.

Выходные данные Если выпить n чашек чая не удастся, выведите «NO» (без кавычек).

В противном случае, выведите строку длины n, состоящую из символов 'G' и 'B'. Если очередной символ равен 'G', то очередная кружка должна быть с зеленым чаем. Если очередной символ равен 'B', то очередная кружка должна быть с черным чаем. Если ответов несколько, разрешается вывести любой из них.

**Листинг и Ход работы**

|  |
| --- |
| n,k,a,b=map(int,input().split()) #считывание введеных данных |
| x=min(a,b)#присваеваем минимальное кол-во пакетиков из 2 видов чая |
| y=max(a,b)#присваеваем максимальное кол-во пакетиков из 2 видов чая |
| if x==a: |
| L='G' |
| H='B' |
| else: |
| H='G' |
| L='B' |
| if y>(x+1)\*k: #если н пакетиков чая не получится выпить |
| print("NO") |
| else: |
| q=y//(x+1) |
| w=y%(x+1) |
| g=H\*q |
| ans="" |
| i=0 |
| while(i<n): |
| ans+=g |
| i+=q |
| if(w): |
| ans+=H |
| w-=1 |
| i+=1 |
| ans+=L |
| i+=1 |
| print(ans[:n])#Вывод результата |

**Задача 7**

Вы — искатель приключений. Сейчас Вы путешествуете по злому храму. Одолев пару злых зомби, Вы прибыли в квадратную комнату, состоящую из плиток, выложенных в таблицу размера n × n. Строки пронумерованы от 1 до n сверху вниз, а столбцы пронумерованы с 1 до n слева направо. В дальнем углу комнаты находится дверь, закрытая злыми магическими силами. На двери написано следующее:

Очищение от всякого зла пробудит дверь! Так как Вы довольно опытный искатель приключений, Вы мгновенно понимаете, в чем тут дело. Вы замечаете, что каждая ячейка в таблице изначально является злой. Надо очистить все эти ячейки.

Единственный известный Вам способ очищения плиток — заклинание «Очищение». Заклинание накладывается на одну плитку, а затем все ячейки в той же строке, а также все ячейки в том же столбце (включая выбранную плитку) очищаются! Разрешается очищать клетку несколько раз.

Вы хотели бы очистить все n × n ячеек, накладывая заклинание очищения как можно меньше раз. В теории все просто, но вы только что заметили, что есть еще особо злые плитки. На них заклинание очищения нельзя накладывать, даже после того, как они были очищены. Но эти плитки все же можно очистить, если наложить заклинание очищения на ячейку в той же строке или столбце.

Пожалуйста, найдите способ очистить все ячейки минимальным количеством накладываемых заклинаний. Если такого способа нет, выведите -1.

Входные данные В первой строке записано единственное целое число n (1 ≤ n ≤ 100). В следующих n строках записано по n символов: j-ый символ i-ой строки обозначает ячейку, расположенную в строке i и столбце j. Особо злые плитки заданы символом «E», остальные — символом «.».

Выходные данные Если очистить все клетки не удастся, то выведите -1. В противном случае, если Ваше решение накладывает x заклинаний очищения, выведите x строк. Каждая строка должна состоять из двух целых чисел, обозначающих номер строки и столбца ячейки, на которую следует наложить заклинание очищения.

**Листинг и Ход работы**

|  |
| --- |
| #считывание введеных данных |
| n = int(input()) |
| maze = [input().strip() for \_ in range(n)] |
|  |
| def go(by\_row,maze): |
| maze = list(zip(\*maze)) |
| can = True |
| for i in range(n): |
| if '.' not in maze[i]: |
| can = False |
| if can: |
| for i in range(n): |
| for j in range(n): |
| if maze[i][j] == '.': |
| print(i + 1, j + 1) if by\_row else print(j + 1, i + 1) |
| break |
| return can #Возвращаем результат функции |
|  |
| if not go(0,maze) and not go(1,maze): |
| print(-1)#Вывод результата |

**Задача 8**

Принцесса собирается сбежать из пещеры дракона. Это нужно как следует спланировать.

Принцесса бегает со скоростью vp миль в час, дракон летает со скоростью vd миль в час. Дракон обнаружит побег через t часов и немедленно бросится в погоню. Затея кажется безнадежной, но принцесса заметила, что дракон очень жаден и не очень умен. Чтобы задержать его, принцесса решает прихватить с собой несколько драгоценностей из его сокровищницы. Когда дракон догоняет принцессу, она может бросить на землю драгоценность; тогда дракон остановится, подберет упавшее, вернется в пещеру и потратит f часов в самой пещере на возвращение вещицы на место и наведение порядка в сокровищнице, после чего вновь отправится в погоню, начиная все с самого начала.

Предполагая, что принцесса будет бежать по прямой без остановок, сколько драгоценностей ей нужно взять с собой, чтобы успеть добраться до королевского замка, расположенного на расстоянии c миль от пещеры дракона? Если дракон догоняет принцессу в тот момент, когда она добегает до замка, считается, что она успела раньше (дополнительная драгоценность не нужна).

Входные данные Входные данные содержат целые числа vp, vd, t, f и c (1 ≤ vp, vd ≤ 100, 1 ≤ t, f ≤ 10, 1 ≤ c ≤ 1000), каждое число записано в отдельной строке.

Выходные данные Выведите минимальное количество драгоценностей, необходимых для того, чтобы побег удался.

**Листинг и Ход работы**

|  |
| --- |
| #считывание введеных данных |
| vp = int(input()) |
| vd = int(input()) |
| t = int(input()) |
| f = int(input()) |
| c = int(input()) |
| if vd <= vp:#если скоростью принцессы больше скорости дракона |
| print(0) |
| else: |
| count = 0 #для подсчета количества |
| while vp \* (t + ((vp \* t) / (vd - vp))) < c:# пока принцесса не добежала до нужной точки |
| count += 1 #добавляем одну драгоценность |
| t += (2 \* vp \* t) / (vd - vp) + f #время которое тратит дракон после выбрасывания драгоценностей |
| print(count)#Вывод результата |

**Задача 9**

Маленький Слоник любит играть с цветными карточками.

У него есть n карточек, каждая имеет ровно два цвета (цвет на передней стороне и цвет на задней стороне). Изначально все карточки лежат на столе передней стороной вверх. За один шаг Маленький Слоник может перевернуть на другую сторону любую карточку. Маленький Слоник считает набор карточек на столе веселым, если хотя бы половина карточек имеет одинаковый цвет (для каждой карточки рассматривается цвет стороны, которой она лежит вверх).

Помогите Маленькому Слонику найти минимальное количество шагов необходимых для превращения набора из n карточек в веселый.

Входные данные В первой строке задано единственное целое число n (1 ≤ n ≤ 105) — количество карточек. Следующие n строк содержат описание всех карточек, по одной карточке на строку. Карточки описываются парой целых положительных чисел, не превосходящих 109, — цветами на обеих сторонах. Первое число в строке — это цвет на передней стороне карточки, второе — на задней. Цвет на передней стороне может совпадать с цветом на задней.

Числа в строках разделяются единичными пробелами.

Выходные данные В единственной строке выведите единственное целое число — искомое минимальное количество шагов. Если превратить набор в веселый невозможно, выведите -1.

**Листинг и Ход работы**

|  |
| --- |
| d=dict()#создаем словарь |
| n=int(input())#ввод кол-во карточек |
| for i in range(n):#массив проходящий по всем карточкам |
| f,b=map(int,input().split())#вводим данные |
| if f not in d: |
| d[f]=[0,0] |
| d[f][0]+=1 |
| if f!=b: |
| if b not in d: |
| d[b]=[0,0] |
| d[b][1]+=1 |
| n=(n+1)//2 |
| m=2\*n |
| for aa,a in d.items(): |
| if sum(a)>=n: |
| m=min(m,n-a[0]) |
| if m==2\*n:#если набор карточек невозможно превратить в веселый |
| print(-1) |
| else: |
| print(max(0,m))#Вывод результата |

**Задача 10**

Вася живет посередине Программистской ветки метро. У него есть две девушки: Даша и Маша, живущие на разных концах ветки, каждая не ведающая о существовании другой.

Когда у Васи появляется свободное время, он едет к одной из своих девушек. Он спускается в метро в некоторый момент времени, ждет первый поезд и едет на нем до конца ветки к соответствующей девушке. Однако поезда ходят с разной частотой: в Дашином направлении станцию проезжает один поезд в a минут, а в Машином — один поезд в b минут. В случае, если поезда подошли одновременно, Вася едет в направлении с меньшей частотой хождения поездов, то есть к той девушке, в чью сторону поезда ходят реже (см. пояснение к третьему примеру).

Известно, что поезда начинают свое хождение одновременно до появления Васи. То есть расписание поездов таково, что существует момент времени, когда два поезда приезжают одновременно.

Помогите Васе посчитать, к какой девушке он будет попадать чаще.

Входные данные В первой строке записано два целых числа a и b (a ≠ b, 1 ≤ a, b ≤ 106).

Выходные данные Выведите «Dasha», если Вася будет чаще попадать к Даше, «Masha», если к Маше, и «Equal», если одинаково часто к обеим девушкам.

**Листинг и Ход работы**

|  |
| --- |
| from math import gcd #импортируем модуль для вычисления НОД |
| a,b = list(map(int,input().split()))#считывание введеных данных |
| g = gcd(a,b)#находим НОД из а и b |
| a = a//g#присваиваем а значение равное частному от деления a на НОД от а и b |
| b = b//g#присваиваем b значение равное частному от деления b на НОД от а и b |
| if abs(a-b)==1:#если разницы нет |
| print("Equal") |
| elif a<b:#если чаще к Даше |
| print("Dasha") |
| else:#если чаще к Маше |
| print("Masha") |

## Вывод

В ходе данной курсовой работы, я научился выполнять разные алгоритмические задачи.