**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 № 10

Выполнил: студент группы БФИ2001

Ковачев В.Е.

Проверил: Г.М. Мкртчян

Москва 2021

**Оглавление**

[**Цель** 3](#_Toc102924160)

[**Задача 1.** 4](#_Toc102924161)

[**Задача 2.** 4](#_Toc102924162)

[**Задача 3.** 4](#_Toc102924163)

[**Задача 4.** 4](#_Toc102924164)

[**Задача 5.** 4](#_Toc102924165)

[**Задача 6.** 4](#_Toc102924166)

[**Задача 7.** 4](#_Toc102924167)

[**Задача 8.** 4](#_Toc102924168)

[**Задача 9.** 4](#_Toc102924169)

[**Задача 10.** 4](#_Toc102924170)

[**Вывод** 4](#_Toc102924171)

# **Цель**

Целью курсовой работы является выполнения алгоритмических задач, описание хода решения, описания алгоритмов решения задач на языке Python и демонстрация результатов работы.

# **Задача 1.**

**Условие:**

Задан массив 𝑎1,𝑎2,…,𝑎𝑛, состоящий из 𝑛 положительных целых чисел.

Изначально вы находитесь в позиции 1, и ваше количество очков равно 𝑎1. Можно совершать два типа шагов:

шаг вправо — перейти из текущей позиции 𝑥 в 𝑥+1 и получить 𝑎𝑥+1 очков. Этот шаг можно делать только если 𝑥<𝑛. шаг влево — перейти из текущей позиции 𝑥 в 𝑥−1 и получить 𝑎𝑥−1 очков. Этот шаг можно делать только если 𝑥>1. Также нельзя совершать два или более шагов влево подряд. Вам требуется совершить ровно 𝑘 шагов. Не более 𝑧 из них могут быть шагами влево.

Какое наибольшее количество очков можно получить?

**Входные данные:**

В первой строке записано одно целое число 𝑡 (1≤𝑡≤104) — количество наборов входных данных.

В первой строке каждого набора входных данных записаны три целых числа 𝑛,𝑘 и 𝑧 (2≤𝑛≤105, 1≤𝑘≤𝑛−1, 0≤𝑧≤𝑚𝑖𝑛(5,𝑘)) — количество элементов в массиве, суммарное количество шагов, которое вы должны сделать, и максимальное количество шагов влево, которое вы можете сделать.

Во второй строке каждого набора входных данных записаны 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤104) — данный массив.

Сумма 𝑛 по всем наборам входных данных не превосходит 3⋅105.

**Выходные данные**

Выведите 𝑡 целых чисел — для каждого набора входных данных выведите наибольшее количество очков, которое можно получить, если требуется сделать ровно 𝑘 шагов, не более 𝑧 из них могут быть шагами влево и не должно быть двух шагов влево подряд.

**Пример:**

входные данные  
4  
5 4 0  
1 5 4 3 2  
5 4 1  
1 5 4 3 2  
5 4 4  
10 20 30 40 50  
10 7 3  
4 6 8 2 9 9 7 4 10 9  
выходные данные  
15  
19  
150  
56

**Примечание**

В первом наборе входных данных не разрешается ходить влево вообще. Поэтому делаем четыре шага вправо и получаем 𝑎1+𝑎2+𝑎3+𝑎4+𝑎5 очков.

Во втором наборе входных данных можно сделать один ход влево. Тогда сделаем такие шаги: вправо, вправо, влево, вправо. Получится 𝑎1+𝑎2+𝑎3+𝑎2+𝑎3 очков.

В третьем наборе входных данных можно ходить влево до четырех раз, но это все равно не оптимально, можем просто сделать четыре шага вправо и получить 𝑎1+𝑎2+𝑎3+𝑎4+𝑎5 очков.

**Листинг 1 – решение задачи 1**

|  |
| --- |
| def operation(n, k, z, a): |
| #объявление переменных метода |
| rez = 0 |
| s = 0 |
| maxi = 0 |
| for i in range(k + 1):#запускаем цикл, который выполняется пока i меньше количества шагов |
| if i < n - 1:#проверяем, что мы можем взять a[i+1] |
| maxi = max(maxi, a[i] + a[i + 1])#максимальное значение из максимального значение перехода с текущим переходом. |
| s+=a[i]#добавляем сумму текущего перехода |
| if i%2==k%2:#если чётность итерируемой переменной и количества ходов совпадает |
| temp=(k-i)//2#записываем в переменную temp результат целочисленного деления оставшихся ходов на 2. |
| if temp<=z:#если temp меньше кол-ва разрешенных ходов на лево |
| rez=max(rez,s+maxi\*temp)#берем максимум из уже найденного значения rez и суммы переменной- |
| #-s, хранящей сумму всех пройденных весов, с максимальным значением перехода умноженным на переменную temp |
| return rez #возвращаем ответ |
|  |
| for \_ in range(int(input())):#вызываем функцию для каждого набора введённых данных |
| #Ввод данных |
| n, k, z = map(int, input().split()) |
| a =list(map(int, input().split())) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(n, k, z, a)) |

**Результат работы:**

Ввод:

4  
5 4 0  
1 5 4 3 2  
5 4 1  
1 5 4 3 2  
5 4 4  
10 20 30 40 50  
10 7 3  
4 6 8 2 9 9 7 4 10 9  
Вывод:  
15  
19  
150  
56

# **Задача 2.**

**Условие:**

Вам дан массив 𝑎 из 𝑛 целых чисел.

Вы хотите сделать все элементы 𝑎 равными нулю, применив следующую операцию ровно три раза:

Выберите отрезок, к каждому числу на этом отрезке добавьте число кратное 𝑙𝑒𝑛, где 𝑙𝑒𝑛 это длина этого отрезка (добавленные числа могут быть разными). Можно доказать, что таким образом всегда можно превратить все элементы 𝑎 в нули.

**Входные данные**:

В первой строке записано одно целое число 𝑛 (1≤𝑛≤100000): количество элементов массиве.

Во второй строке записаны 𝑛 элементов массива 𝑎, разделенные пробелами: 𝑎1,𝑎2,…,𝑎𝑛 (−109≤𝑎𝑖≤109).

**Выходные данные:**

Выведите шесть строк, описывающих три операции.

Для каждой операции, выведите две строки:

В первой строке выведите два числа 𝑙, 𝑟 (1≤𝑙≤𝑟≤𝑛): границы выбранного отрезка. Во второй строке выведите 𝑟−𝑙+1 целых чисел 𝑏𝑙,𝑏𝑙+1,…,𝑏𝑟 (−1018≤𝑏𝑖≤1018): числа, которые нужно прибавить к 𝑎𝑙,𝑎𝑙+1,…,𝑎𝑟, соответственно; 𝑏𝑖 должно делиться на 𝑟−𝑙+1.

**Пример**:  
входные данные  
4  
1 3 2 4  
выходные данные  
1 1  
-1  
3 4  
4 2  
2 4  
-3 -6 -6

**Листинг 2 – решение задачи 2**

|  |
| --- |
| def operation(n,a): |
| if n == 1:#исключение на случай ввода одного числа |
| print('1 1', -a[0], '1 1', '0', '1 1', '0', sep='\n') |
| return |
|  |
| print(1, n)#выбираем диапазон |
| for i in range(n):#вывод чисел 𝑏𝑙,𝑏𝑙+1,…,𝑏𝑟 для первого действия |
| print(-a[i] \* n, end = ' ') |
| a[i] -= a[i] \* n # изменяем значение текущего элемента массива |
| print()#переводим на следующую строку |
|  |
| print(1, n - 1)#выбираем диапазон |
| for i in range(n - 1):#вывод чисел 𝑏𝑙,𝑏𝑙+1,…,𝑏𝑟 для второго действия |
| print(-a[i], end = ' ') |
| print()#переводим на следующую строку |
|  |
| print(2, n)#выбираем диапазон |
| for i in range(1, n):#вывод чисел 𝑏𝑙,𝑏𝑙+1,…,𝑏𝑟 для третьего действия |
| print(-a[i], end = ' ') |
| print()#переводим на следующую строку |
|  |
| #Ввод данных |
| n = int(input()) |
| a = list(map(int, input().split())) |
| #Вызов функции operation для выполнения вычислений |
| operation(n,a) |

**Результат работы:**

Ввод:

4

1 3 2 4

Вывод:

1 4

-4 -12 -8 -16

1 3

3 9 6

2 4

9 6 12

# **Задача 3.**

**Условие:**

Вам задан массив 𝑎, состоящий из 𝑛 целых чисел. Индексы элементов массива начинаются с нуля (то есть первый элемент — это 𝑎0, второй — 𝑎1, и так далее).

Вы можете развернуть не более одного подмассива (последовательного отрезка) этого массива. Напомним, что подмассив 𝑎 с границами 𝑙 и 𝑟 равен 𝑎[𝑙;𝑟]=𝑎𝑙,𝑎𝑙+1,…,𝑎𝑟.

Ваша задача — развернуть такой подмассив, чтобы сумма элементов на четных позицияx получившегося массива была максимально возможной (то есть сумма элементов 𝑎0,𝑎2,…,𝑎2𝑘 для целого числа 𝑘=⌊𝑛−12⌋ должна быть максимально возможной).

Вам необходимо ответить на 𝑡 независимых наборов тестовых данных.

**Входные данные:**

Первая строка входных данных содержит одно целое число 𝑡 (1≤𝑡≤2⋅104) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Первая строка набора содержит одно целое число 𝑛 (1≤𝑛≤2⋅105) — длину 𝑎. Вторая строка набора содержит 𝑛 целых чисел 𝑎0,𝑎1,…,𝑎𝑛−1 (1≤𝑎𝑖≤109), где 𝑎𝑖 — это 𝑖-й элемент 𝑎.

Гарантируется, что сумма 𝑛 не превосходит 2⋅105 (∑𝑛≤2⋅105).

**Выходные данные:**

Для каждого набора тестовых данных выведите ответ на отдельной строке — максимально возможная сумма элементов на четных позициях после разворота не более одного подмассива (последовательного отрезка) 𝑎.

**Пример:**

входные данные  
4  
8  
1 7 3 4 7 6 2 9  
5  
1 2 1 2 1  
10  
7 8 4 5 7 6 8 9 7 3  
4  
3 1 2 1  
выходные данные  
26  
5  
37  
5

**Листинг 3 – решение задачи 3**

|  |
| --- |
| def operation(n,l): |
| #объявление переменных метода |
| mx=0 |
| cur=0 |
| for i in range(0,n-1,2): #запускаем цикл для построения подмассива 1 способом |
| cur+=l[i+1]-l[i]#вычитаем из последующего текущей элемент и добавляем разность к cur |
| mx=max(cur,mx) #ищем максимальную выгоду от текущего варианта подмассива |
| cur=0 |
| for i in range(1,n-1,2):#запускаем цикл для построения подмассива 2 способом (начинаем с 1) |
| cur+=l[i]-l[i+1]# вычитаем из текущего последующий элемент и добавляем с cur |
| mx=max(cur,mx) #ищем максимальную выгоду от текущего варианта подмассива |
| s=0 |
| for i in range(0,n,2):#проходим по всем чётным элементам |
| s+=l[i]#суммирует все чётные элементы |
| return s+mx #возвращаем сумму чётных элементов и наибольшей выгоды от развёртывания подмассива |
|  |
| for \_ in range(int(input())):#вызываем функцию для каждого набора введённых данных |
| #Ввод данных |
| n=int(input()) |
| l=list(map(int,input().split())) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(n,l)) |

**Результат работы:**

Ввод:  
4  
8  
1 7 3 4 7 6 2 9  
5  
1 2 1 2 1  
10  
7 8 4 5 7 6 8 9 7 3  
4  
3 1 2 1  
Вывод:  
26  
5  
37  
5

# **Задача 4.**

**Условие:**

У Пети есть прямоугольная доска размера 𝑛×𝑚. Изначально на доске размещено 𝑘 фишек, 𝑖-я из них находится в клетке, на пересечении 𝑠𝑥𝑖-й строки и 𝑠𝑦𝑖-го столбца.

За одно действие Петя может сдвинуть все фишки влево, вправо, вниз или вверх на 1 ячейку.

Если фишка была в клетке (𝑥,𝑦), то после операции:

влево, ее координаты будут (𝑥,𝑦−1); вправо, ее координаты будут (𝑥,𝑦+1); вниз, ее координаты будут (𝑥+1,𝑦); вверх, ее координаты будут (𝑥−1,𝑦). Если фишка находится у стенки доски, и действие, выбранное Петей, двигает ее в направлении стены, то фишка остается на своей текущей позиции.

Обратите внимание, что несколько фишек могут располагаться в одной и той же клетке.

Для каждой фишки Петя выбрал позицию, в которой она должна побывать. Обратите внимание, что фишка не обязательно заканчивает в этой позиции.

Так как у Пети не очень много свободного времени, он готов сделать не более 2𝑛𝑚 действий, описанных выше.

Вам предстоит выяснить, какие действия должен делать Петя, чтобы все фишки побывали хотя бы раз в выбранных для них клетках. Или определить, что за 2𝑛𝑚 действий выполнить это невозможно.

**Входные данные:**

Первая строка содержит три целых числа 𝑛,𝑚,𝑘 (1≤𝑛,𝑚,𝑘≤200) — количество строк и столбцов доски и количество фишек соответственно.

Следующие 𝑘 содержат по два целых чисел 𝑠𝑥𝑖,𝑠𝑦𝑖 (1≤𝑠𝑥𝑖≤𝑛,1≤𝑠𝑦𝑖≤𝑚) — начальная позиция 𝑖-й фишки.

Следующие 𝑘 содержат по два целых чисел 𝑓𝑥𝑖,𝑓𝑦𝑖 (1≤𝑓𝑥𝑖≤𝑛,1≤𝑓𝑦𝑖≤𝑚) — позиция, которую должна посетить 𝑖-я фишка хотя бы раз.

**Выходные данные:**

В первой строке выведите количество операций, чтобы каждая фишка посетила хотя бы раз позицию, которую выбрал для нее Петя.

Во второй строке выведите последовательность операций. Для обозначения операций влево, вправо, вниз, вверх используйте символы 𝐿,𝑅,𝐷,𝑈 соответственно.

Если искомой последовательности не существует, в единственной строке выведите -1.

**Примеры:**

входные данные  
3 3 2  
1 2  
2 1  
3 3  
3 2  
выходные данные  
3  
DRD  
входные данные  
5 4 3  
3 4  
3 1  
3 3  
5 3  
1 3  
1 4  
выходные данные  
9  
DDLUUUURR

**Листинг 4 – решение задачи 4**

|  |
| --- |
| def operation(n, m, k, l): |
| rez=""#создаём переменную которая будет хранить очерёдность шагов |
| #собираем все точки в одной клетке |
| rez+=(m-1)\*'L' |
| rez+=(n-1)\*'U' |
|  |
| for i in range(m-1):#с помощью цикла проходим все клетки двигаясь вправо и проходя вниз\вверх клетки |
| if i%2==0:#для смены движения вверх/вниз |
| #перемещаемся 1 раз вправо и движемся вниз |
| rez+='R' |
| rez+=(n-1)\*'D' |
|  |
| else: |
| #перемещаемся 1 раз вправо и движемся вверх |
| rez+='R' |
| rez+=(n-1)\*'U' |
| return rez #возвращаем последовательность шагов |
|  |
| #Ввод данных |
| n,m,k=map(int,input().split()) |
| for i in range(2\*k): |
| l=list(map(int,input().split())) |
| #Вызов функции operation для выполнения вычислений |
| mas=operation(n, m, k, l) |
| print(len(mas))#печать кол-ва шагов |
| print(mas)#массив шагов |

**Результат работы:**

Ввод:

3 3 2

1 2

2 1

3 3

3 2

Вывод:

10

LLUURDDRUU

# **Задача 5.**

**Условие:**

Летние каникулы начались, поэтому Алиса и Боб хотят играть и веселиться, но... Их мама не согласна с этим. Она говорит, что они должны прочитать какое-то количество книг перед всеми развлечениями. Алиса и Боб прочитают каждую книгу вместе, чтобы быстрее закончить это задание.

В семейной библиотеке есть 𝑛 книг. 𝑖-я книга характеризуется тремя целыми числами: 𝑡𝑖 — количество времени, которое Алиса и Боб должны потратить, чтобы прочитать ее, 𝑎𝑖 (равное 1, если Алисе нравится 𝑖-я книга, и 0, если не нравится), и 𝑏𝑖 (равное 1, если Бобу нравится 𝑖-я книга, и 0, если не нравится).

Поэтому им нужно выбрать какие-то книги из имеющихся 𝑛 книг таким образом, что:

Алисе нравятся не менее 𝑘 книг из выбранного множества и Бобу нравятся не менее 𝑘 книг из выбранного множества; общее время, затраченное на прочтение этих книг минимизировано (ведь они дети и хотят начать играть и веселиться как можно скорее). Множество, которое они выбирают, одинаковое и для Алисы и для Боба (они читают одни и те же книги), и они читают все книги вместе, таким образом, суммарное время чтения равно сумме 𝑡𝑖 по всем книгам, которые находятся в выбранном множестве.

Ваша задача — помочь им и найти любое подходящее множество книг или определить, что такое множество найти невозможно.

**Входные данные:**

Первая строка теста содержит два целых числа 𝑛 и 𝑘 (1≤𝑘≤𝑛≤2⋅105).

Следующие 𝑛 строк содержат описания книг, по одному описанию в строке: 𝑖-я строка содержит три целых числа 𝑡𝑖, 𝑎𝑖 и 𝑏𝑖 (1≤𝑡𝑖≤104, 0≤𝑎𝑖,𝑏𝑖≤1), где:

𝑡𝑖 — количество времени, необходимое для прочтения 𝑖-й книги;  
𝑎𝑖, равное 1, если Алисе нравится 𝑖-я книга, и 0 в обратном случае;  
𝑏𝑖, равное 1, если Бобу нравится 𝑖-я книга, и 0 в обратном случае.  
**Выходные данные:**

Если подходящего решения не существует, выведите число -1. Иначе выведите целое число 𝑇 — минимальное суммарное время, необходимое для прочтения подходящего множества книг.

**Примеры:**

входные данные  
8 4  
7 1 1  
2 1 1  
4 0 1  
8 1 1  
1 0 1  
1 1 1  
1 0 1  
3 0 0  
выходные данные  
18  
входные данные  
5 2  
6 0 0  
9 0 0  
1 0 1  
2 1 1  
5 1 0  
выходные данные  
8  
входные данные  
5 3  
3 0 0  
2 1 0  
3 1 0  
5 0 1  
3 0 1  
выходные данные  
-1

**Листинг 5 – решение задачи 5**

|  |
| --- |
| def operation(n, k): |
| a = []#Для книг которые нравятся Алисе |
| b = []#Для книг которые нравятся Бобу |
| s = [] |
| for i in ' ' \* n: |
| t, x, y = map(int, input().split()) |
| if x & y:#если книга нравится обоим |
| s += [t] |
| elif x:#Если книга нравится Алисе |
| a += [t] |
| elif y:#Если книга нравится Бобу |
| b += [t] |
|  |
| s += [i + j for i, j in zip(sorted(a), sorted(b))]# добавляем в массив s книги которые нравятся 1 или 2 ребёнку |
| return -1 if len(s) < k else sum(sorted(s)[:k])# проверяем что подходящее множество существует или выводим -1 |
|  |
| #Ввод данных |
| n, k = map(int, input().split()) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(n, k)) |

**Результат работы:**

Ввод:

8 4

7 1 1

2 1 1

4 0 1

8 1 1

1 0 1

1 1 1

1 0 1

3 0 0

Вывод:

18

# **Задача 6.**

**Условие:**

Вы — амбициозный король, который хочет быть Императором Действительных чисел. Но перед этим вам нужно стать Императором Целых чисел.

Рассмотрим числовую ось. Столица вашей империи изначально находится в точке 0. На прямой есть 𝑛 незахваченных королевств в позициях 0<𝑥1<𝑥2<…<𝑥𝑛. Вы хотите захватить все эти королевства.

Вы можете делать два вида действий:

Вы можете переместить свою столицу (пусть ее текущая координата 𝑐1) в любое захваченное королевство (пусть его позиция 𝑐2), стоимость такого действия 𝑎⋅|𝑐1−𝑐2|. Вы можете из текущей столицы (пусть ее текущая координата 𝑐1) захватить королевство (пусть его позиция 𝑐2), стоимость такого действия 𝑏⋅|𝑐1−𝑐2|. Вы не можете захватывать королевство, если между вашей столицей и целью есть другие незахваченные королевства. Обратите внимание, что вы не можете расположить столицу в точке, где нет королевства. Другими словами, в любой момент времени ваша столица может быть только в точке 0 или в 𝑥1,𝑥2,…,𝑥𝑛. Также обратите внимание, что захват королевства не изменяет положение вашей столицы.

Выведите минимальную суммарную стоимость захвата всех королевств. Ваша столица может оказаться в итоге в любой точке.

**Входные данные:**

Первая строка содержит одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных. Далее следуют описания наборов.

Первая строка каждого набора содержит 3 целых числа 𝑛, 𝑎 и 𝑏 (1≤𝑛≤2⋅105; 1≤𝑎,𝑏≤105).

Вторая строка каждого набора содержит 𝑛 целых чисел 𝑥1,𝑥2,…,𝑥𝑛 (1≤𝑥1<𝑥2<…<𝑥𝑛≤108).

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

**Выходные данные:**

Для каждого набора входных данных выведите одно число: минимальную стоимость захвата всех королевств.

**Пример:**

входные данные  
4  
5 2 7  
3 5 12 13 21  
5 6 3  
1 5 6 21 30  
2 9 3  
10 15  
11 27182 31415  
16 18 33 98 874 989 4848 20458 34365 38117 72030  
выходные данные  
173  
171  
75  
3298918744  
**Примечание:**

Ниже приведена оптимальная последовательность действий для второго примера:\

Захватить королевство в точке 1, стоимость 3⋅(1−0)=3.  
Переместить столицу в королевство в точке 1, стоимость 6⋅(1−0)=6.  
Захватить королевство в точке 5, стоимость 3⋅(5−1)=12.  
Переместить столицу в королевство в точке 5, стоимость 6⋅(5−1)=24.  
Захватить королевство в точке 6, стоимость 3⋅(6−5)=3.  
Захватить королевство в точке 21, стоимость 3⋅(21−5)=48.  
Захватить королевство в точке 30, стоимость 3⋅(30−5)=75.  
Суммарная стоимость 3+6+12+24+3+48+75=171. Нельзя получить меньшую стоимость.

**Листинг 6 – решение задачи 6**

|  |
| --- |
| def operation(n, a, b, vh): |
| sm = sum(vh)#суммируем все значения королевств |
| rs=sm \* b #находим самый дорогой дорогой вариант действий |
| s=0 |
| for j in range(n):#проходим по всем королевствам |
| s += vh[j]#суммируем значение пройденных королевств |
| rs = min(rs, (a + b)\*vh[j] + (sm-s - (n-j-1)\*vh[j])\*b)#находим до какого момента нам выгоднее - |
| #- перемещаться на захваченное королевство, а не захватывать напрямую |
| return rs #возвращаем полученное значение |
|  |
| for i in range(int(input())):#вызываем функцию для каждого набора введённых данных |
| #Ввод данных |
| n,a,b = map(int, input().split()) |
| vh = list(map(int, input().split())) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(n, a, b, vh)) |

**Результат работы:**

Ввод:

4  
5 2 7  
3 5 12 13 21  
5 6 3  
1 5 6 21 30  
2 9 3  
10 15  
11 27182 31415  
16 18 33 98 874 989 4848 20458 34365 38117 72030

Вывод:

173

171

75

# **Задача 7.**

**Условие:**

Вам даны две последовательности целых чисел 𝑎1,…,𝑎𝑛 и 𝑏1,…,𝑏𝑚. Для каждого 𝑗=1,…,𝑚 найдите наибольший общий делитель чисел 𝑎1+𝑏𝑗,…,𝑎𝑛+𝑏𝑗.

**Входные данные:**

В первой строке записано два целых числа 𝑛 и 𝑚 (1≤𝑛,𝑚≤2⋅105).

Во второй строке записано 𝑛 целых чисел 𝑎1,…,𝑎𝑛 (1≤𝑎𝑖≤1018).

В третьей строке записано 𝑚 целых чисел 𝑏1,…,𝑏𝑚 (1≤𝑏𝑗≤1018).

**Выходные данные:**

Выведите 𝑚 целых чисел. 𝑗-е из этих чисел должно быть равно НОД(𝑎1+𝑏𝑗,…,𝑎𝑛+𝑏𝑗).

**Пример:**  
входные данные  
4 4  
1 25 121 169  
1 2 7 23  
выходные данные  
2 3 8 24

**Листинг 7 – решение задачи 7**

|  |
| --- |
| from math import gcd #подключаем для нахождения НОД |
|  |
| def operation(n, m, a, b): |
| g = 0 #переменная для НОД первого массива |
| for i in a[1:]:#цикл для получения НОД первого массива |
| g = gcd(g,i-a[0])#находим НОД первого массива |
| for i in range(m):#цикл для получения НОД второго массива |
| print(gcd(g,a[0]+b[i]),end = " ")#находим НОД для an+bj и НОДа первого массива, получая числа равные НОД(𝑎1+𝑏𝑗,…,𝑎𝑛+𝑏𝑗). |
|  |
| #Ввод данных |
| n,m = map(int,input().split()) |
| a = list(map(int,input().split())) |
| b = list(map(int,input().split())) |
| #Вызов функции operation для выполнения вычислений |
| operation(n, m, a, b) |

**Результат работы:**

Ввод:  
4 4

1 25 121 169

1 2 7 23

Вывод:

2 3 8 24

# **Задача 8.**

**Условие:**

Вам заданы два массива целых чисел 𝑎 и 𝑏 длины 𝑛.

Вы можете развернуть не более одного подмассива (последовательного отрезка) массива 𝑎.

Ваша задача состоит в том, чтобы перевернуть такой подмассив, чтобы сумма ∑𝑖=1𝑛𝑎𝑖⋅𝑏𝑖 была максимально возможной.

**Входные данные:**

Первая строка содержит одно целое число 𝑛 (1≤𝑛≤5000).

Вторая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤107).

Третья строка содержит 𝑛 целых чисел 𝑏1,𝑏2,…,𝑏𝑛 (1≤𝑏𝑖≤107).

**Выходные данные:**

Выведите одно целое число — максимально возможную сумма после разворота не более одного подмассива (последовательного отрезка) 𝑎.

**Примеры:**  
входные данные  
5  
2 3 2 1 3  
1 3 2 4 2  
выходные данные  
29  
входные данные  
2  
13 37  
2 4  
выходные данные  
174  
входные данные  
6  
1 8 7 6 3 6  
5 9 6 8 8 6  
выходные данные  
235  
**Примечание:**

В первом примере можно перевернуть подмассив [4,5]. Тогда 𝑎=[2,3,2,3,1] и 2⋅1+3⋅3+2⋅2+3⋅4+1⋅2=29.

Во втором примере не нужно использовать операцию. 13⋅2+37⋅4=174.

В третьем примере можно перевернуть подмассив [3,5]. Тогда 𝑎=[1,8,3,6,7,6] и 1⋅5+8⋅9+3⋅6+6⋅8+7⋅8+6⋅6=235.

**Листинг 8 – решение задачи 8**

|  |
| --- |
| def operation(n,a,b): |
| rez = 0 #переменная для хранения максимальной суммы |
| for i in range(n): |
| rez += a[i] \* b[i] #находим начальное значение для переменной рез до развертывания подмассива |
| base = rez # создаём переменную которая сохранит базовое значение для массива до развёртывания подмассива |
| for i in range(n): |
| for j in (i - 1, i): # цикл который позволяет получить варианты подмассивов начинающихся с- |
| # - нечётного/чётного индекса для создания и проверки всех вариантов подмассива |
| cur = base #присваиваем новой переменной значение rez что-бы была возможность получить знаечние после изменений обратно |
| k = i + 1 |
| while j >= 0 and k < n: # запускаем цикл который проверит все варианты подмассива с текущего индекса |
| cur -= (a[j] - a[k]) \* (b[j] - b[k]) # рассчитываем выгоду от расширения подмассива на 2 текущих значения |
| rez = max(rez, cur) # изменяем значение результирующий переменной если cur Больше её текущего значения |
| j -= 1 # расширяем подмассив на 1 влево |
| k += 1 # расширяем подмассив на 1 вправо |
| return rez #возвращаем результат |
|  |
| #Ввод данных |
| n = int(input()) |
| a = list(map(int, input().split())) |
| b = list(map(int, input().split())) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(n,a,b)) |

**Результат работы:**

Ввод:

5

2 3 2 1 3

1 3 2 4 2

Вывод:

29

# **Задача 9.**

**Условие:**

Жил-был в одной стране Царь по имени Цопа. После очередной царской реформы полномочия Царя стали настолько широкими, что, в частности, он стал собственноручно заниматься финансовой отчётностью.

Известен суммарный доход A его Царства по итогам 0-го года, и суммарный доход B по итогам n-го года (оба числа могут быть отрицательными, что означает — в этот год экономика Царства была убыточной).

Царь хочет продемонстрировать финансовую стабильность, для этого ему надо подобрать единый коэффициент X — коэффициент роста дохода Царства за один год. Этот коэффициент должен удовлетворять уравнению:

A·Xn = B. Разумеется, Царь не собирается делать такую работу вручную, и требует от вас написать программу, ищущую этот коэффициент X.

Следует отметить, что дробные числа крайне не любят в экономических структурах Царства, поэтому как входные данные, так и искомый коэффициент должны быть целыми. Искомый коэффициент X может оказаться равным нулю или даже быть отрицательным.

**Входные данные:**

В единственной строке записаны три целых числа A, B, n, удовлетворяющих условиям: |A|, |B| ≤ 1000, 1 ≤ n ≤ 10.

**Выходные данные:**

Выведите единственное целое число — искомый целый коэффициент X, или фразу «No solution», если такого коэффициента не существует, или он не целый. Если ответов несколько, выведите любой.

**Примеры:**

входные данные  
2 18 2  
выходные данные  
3  
входные данные  
-1 8 3 выходные данные  
-2  
входные данные  
0 0 10  
выходные данные  
5  
входные данные  
1 16 5  
выходные данные  
No solution

**Листинг 9 – решение задачи 9**

|  |
| --- |
| def operation(a,b,n): |
| for x in range(-1001,1000):#проходим по всему диапазону возможных значений |
| if a\*x\*\*n==b:#подставляем значение в форму |
| return x #если значение подходит возвращаем его |
| return 'No solution'#если не нашлось подходящего значения |
|  |
| #Ввод данных |
| a,b,n=map(int,input().split()) |
| #Вызов функции operation для выполнения вычислений |
| print(operation(a,b,n)) |

**Результат работы:**

Ввод:  
-1 8 3  
Вывод:

-2

# **Задача 10.**

**Условие:**

У Алисы есть торт, и она собирается его разрезать. Она выполнит следующую операцию 𝑛−1 раз: выберет кусочек пирога (изначально весь торт является одним куском) с весом 𝑤≥2 и разрежем его на два более маленьких кусочка с весами ⌊𝑤2⌋ и ⌈𝑤2⌉ (⌊𝑥⌋ и ⌈𝑥⌉ обозначают округление вниз и вверх, соответственно).

После того как Алиса разрежет торт на 𝑛 кусочков, она расположит эти 𝑛 кусочков на столе в произвольном порядке. Обозначим за 𝑎𝑖 вес 𝑖-го кусочка.

Вам дан массив 𝑎. Определите, существует ли некоторый начальный вес торта и последовательность операций разрезания такие, чтобы веса кусочков получались равными 𝑎.

**Входные данные:**

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

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

Вторая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤109).

Гарантируется, что сумма 𝑛 для всех наборов входных данных не превышает 2⋅105.

**Выходные данные:**

Для каждого набора входных данных выведите одну строку: выведите YES, если массив 𝑎 может быть получен с помощью действий Алисы, иначе выведите NO.

Вы можете выводить буквы в любом регистре (например, YES, Yes, yes, yEs будут приняты как положительный ответ).

**Пример:**

входные данные  
14  
1  
327  
2  
869 541  
2  
985214736 985214737  
3  
2 3 1  
3  
2 3 3  
6  
1 1 1 1 1 1  
6  
100 100 100 100 100 100  
8  
100 100 100 100 100 100 100 100  
8  
2 16 1 8 64 1 4 32  
10  
1 2 4 7 1 1 1 1 7 2  
10  
7 1 1 1 3 1 3 3 2 3  
10  
1 4 4 1 1 1 3 3 3 1  
10  
2 3 2 2 1 2 2 2 2 2  
4  
999999999 999999999 999999999 999999999  
выходные данные  
YES  
NO  
YES  
YES  
NO  
YES  
NO  
YES  
YES  
YES  
YES  
NO  
NO  
YES  
Примечание:

В первом примере можно получить массив 𝑎, выполнив 0 операций с тортом веса 327.

Во втором примере невозможно получить массив 𝑎.

В третьем примере можно получить массив 𝑎, выполнив 1 операцию с тортом веса 1970429473:

Разрезать торт пополам, веса будут равны [985214736,985214737]. Обратите внимание, что начальный вес торта может быть больше 109. В четвертом примере можно получить массив 𝑎 выполнив 2 операции над тортом веса 6:

Разрезать торт пополам, веса кусочков будут равны [3,3]. Разрезать один из кусочков веса 3, веса кусочков будут равны [1,2,3], что совпадает с [2,3,1] с точностью до порядка.

**Листинг 10 – решение задачи 10**

|  |
| --- |
| from collections import Counter #1.Импортируем Counter для подсчета количества появлений неизменяемых элементов в последовательности |
|  |
| def operation(Sum): |
| if d[Sum]>0:#условие для конечной ветки рекурсии, которое возвращает результат |
| d[Sum]-=1 |
| return True |
| if Sum==1:#условие для конечной ветки рекурсии, которое возвращает результат |
| return False |
| return operation(Sum//2) and operation((Sum+1)//2)#метод рекурсивно вызывает себя два раза- |
| #- c разными параметрами, для проверки возможных вариантов разрезания торта |
|  |
| for \_ in range(int(input())):#вызываем функцию для каждого набора введённых данных |
| #Ввод данных |
| n=int(input()) |
| a=list(map(int,input().split())) |
| #создаём обьект Counter |
| d=Counter(a) |
| #Вызов функции operation для выполнения вычислений |
| print("YES" if operation(sum(a)) else "NO") |

**Результат работы:**

Ввод:

14  
1  
327  
2  
869 541  
2  
985214736 985214737  
3  
2 3 1  
3  
2 3 3  
6  
1 1 1 1 1 1  
6  
100 100 100 100 100 100  
8  
100 100 100 100 100 100 100 100  
8  
2 16 1 8 64 1 4 32  
10  
1 2 4 7 1 1 1 1 7 2  
10  
7 1 1 1 3 1 3 3 2 3  
10  
1 4 4 1 1 1 3 3 3 1  
10  
2 3 2 2 1 2 2 2 2 2  
4  
999999999 999999999 999999999 999999999

Вывод:

YES  
NO  
YES  
YES  
NO  
YES  
NO  
YES  
YES  
YES  
YES  
NO  
NO  
YES

# **Вывод**

В ходе выполнения курсовой работой были освоены навыки решения алгоритмических задач.