Сортировка-2. Применение.

В Python есть два способа что-нибудь отсортировать:

- метод `sort()`, который можно применять только к спискам;
- функция `sorted`, которой можно передать в виде параметра не только список, на и любой итерируемый объект (строку, например)

Ссылка на стандартную документацию.

Вот их основные отличия:

`метод sort()`	функция `sorted`
метод меняет список, к которому применяется	функция не меняет переданный список
метод ничего не возвращает (точнее, возвращает `None`)	функция возвращает список отсортированных элементов переданного ей итерируемого объекта

Примеры того, как не надо делать:

```
x = [23, 45, 3, 4, 2]
sorted(x)        # внутри функции будет создан массив из отсортированных
                 # значений массива x, но он не сохраняется и x не меняется
print(x.sort()) # здесь массив будет отсортирован, но print() выведет None
```

Примеры, где удобно использовать функцию:

```
# -*- coding: cp1251 -*-

x = [
    (34, 3, "Ivanov"),
    (10, 99, "Petrova"),
    (5, 17, "Ivanopulo"),
    (9, 1, "Petrov"),
    (100, 100 ,"Petrova")
]

# делаем массив из имён отсортированных кортежей
sorted_names = [person[2] for person in sorted(x)]
print(" ".join(sorted_names))
# Ivanopulo Petrov Petrova Ivanov Petrova

# цикл по отсортированным кортежам (переданный в качестве аргумента массив не меняется)
for num1, num2, name in sorted(x):
    print(num1, num2, name)

# 5 17 Ivanopulo
# 9 1 Petrov
# 10 99 Petrova
# 34 3 Ivanov
# 100 100 Petrova

s = "ABRACADABRA"
print(sorted(s))
# На самом деле функция превращает переданный аргумент в массив,
# а затем создаёт отсортированный результат на его основе
# ['A', 'A', 'A', 'A', 'A', 'B', 'B', 'C', 'D', 'R', 'R']
```

Несколько примеров использования своего компаратора. Все они работают одинаково как в методе, так и в функции:

```
# -*- coding: cp1251 -*-

def digit_sum(x):
    ans = 0
    while x > 0:
        ans += x % 10
        x //= 10
    return ans


def cmp_len_only(s):
    return len(s)


def cmp_len_value(s):
    return len(s), s

x = [23, 34, 3, 10, 99, 5]
x.sort(reverse=True)
print(x)   # [99, 34, 23, 10, 5, 3]

x = [7, 1, 17, 3, 2, 13, 1, 11, 12, 2]
x.sort(key=digit_sum)     # сортируем по сумме цифр числа
                          # числа с одинаковой суммой цифр
                          # сохраняют первоначальный порядок
print(x)
# [1, 1, 2, 11, 2, 3, 12, 13, 7, 17]

x = ["AQ", "ZWCP", "QUO", "XN", "ITIQA", "NRQJ", "NCH", "SAA", "EBMDZ", "BDV"]

x.sort(key=cmp_len_only)
print(x)
# Строки упорядочены по длине. Строки одинаковой длины не поменяли относительный порядок.
# ["AQ", "XN", "QUO", "NCH", "SAA", "BDV", "ZWCP", "NRQJ", "ITIQA", "EBMDZ"]

x.sort(key=cmp_len_value)
print(x)
# Строки упорядочены по длине, а равные по длине - лексикографически.
# ["AQ", "XN", "BDV", "NCH", "QUO", "SAA", "NRQJ", "ZWCP", "EBMDZ", "ITIQA"]


x = ["AQ", "ZWCP", "QUO", "XN", "ITIQA", "NRQJ", "NCH", "SAA", "EBMDZ", "BDV"]
x.sort()
x.sort(key=cmp_len_only)
print(x)
# Того же результаты можно добиться, отсортировав два раза (пользуясь устойчивостью сортировки)
# ["AQ", "XN", "BDV", "NCH", "QUO", "SAA", "NRQJ", "ZWCP", "EBMDZ", "ITIQA"]
```

Так можно добавлять в массив порядковый номер элемента, если он нужен после сортировки:
⤓

```
N = int(input())
# x = list(map(int, input().split()))
x = [3, 5, 9, 8, 1, -4, 7, 5]
x = [(x[k], k + 1) for k in range(N)]

print(x)
# [(3, 1), (5, 2), (9, 3), (8, 4), (1, 5), (-4, 6), (7, 7), (5, 8)]

x.sort()
print(x)
# [(-4, 6), (1, 5), (3, 1), (5, 2), (5, 8), (7, 7), (8, 4), (9, 3)]

print(f"index: {3}, value: {x[3][0]}, serial number before sort: {x[3][1]}")
# index: 3, value: 5, serial number before sort: 2

print(f"index: {6}, value: {x[6][0]}, serial number before sort: {x[6][1]}")
# index: 6, value: 8, serial number before sort: 4
```

A. Дан массив слов, содержащих строчные латинские буквы. Упорядочить слова по возрастанию с учётом обратного чтения, т.е. не по начальным, а по конечным буквам (упорядоченный таким образом словарь называется *обратным* и удобен при изучении особенностей строения конца слов и вообще словообразования, подборе рифм).

В единственной строке вводится N ($1 \leqslant N \leqslant 10^3$) слов из строчных букв длиной не более 50 символов, разделённых пробелами.

Input	Output
`window table ten screen`	`table screen ten window`

Определение. Сортировка называется *устойчивой*, если она не меняет местами равные элементы исходного массива.

При первом прочтении определения может возникнуть впечатление, что оно имеет мало смысла. Действительно, ведь равные элементы неразличимы и нет никакой разницы, в каком порядке они идут. Это не совсем так. Представьте себе таблицу с двумя столбцами, в первом записаны фамилии учеников, во втором класс, в котором учится каждый из учеников. Мы хотим расположить строки этой таблицы так, чтобы сначала были все ученики первого класса в алфавитном порядке, затем второго, также в алфавитном порядке, и так далее. Для этого нам достаточно сначала отсортировать таблицу по первому столбцу, а затем по второму. Если в таблице используется устойчивая сортировка, то после сортировки учеников по классам среди равных элементов (учеников из одного класса) исходный порядок (алфавитный) не нарушится.

B. *Благосостояние и стабильность.*

В некой стране есть три разных валюты — динары, лиры и тугрики. Известно, что по текущему курсу валют, одна лира равна пяти динарам, а один тугрик равен семи динарам. Журнал «Благосостояние и стабильность» каждый год публикует список самых состоятельных граждан (не более 1000). Главный казначей страны выдает редактору журнала список главных богачей в следующем формате:

`<фамилия> <количество динаров> <количество лир> <количество тугриков>.`

Список составлен в порядке влиятельности богачей. Ваша задача отсортировать список по общему благосостоянию, причем если двое граждан имеют одинаковое количество денег, в вашем списке они должны быть расположены в порядке влиятельности (сортировка должна быть *стабильной*).

В первой строке задается число N — количество людей в списке казначея ($N \leqslant 1000$) . В последующих N строках задается список казначея в описанном формате.

Выведите список фамилий самых богатых людей. Каждую фамилию нужно выводить на отдельной строке.

Input	Output

`4`
`Ivanov 34 0 0`
`Petrov 0 7 0`
`Sidorov 0 0 5`
`Rosenblum 1000 1000 100500`
	`Rosenblum`
	`Petrov`
	`Sidorov`
	`Ivanov`

C. *Результаты олимпиады*

Во время проведения олимпиады каждый из участников получил свой идентификационный номер — целое неотрицательное число. Необходимо отсортировать список участников олимпиады по количеству набранных ими баллов.

На первой строке дано число N ($1 \leqslant N \leqslant 1000$) — количество участников. В следующих N строках даны идентификационный номер и набранное число баллов соответствующего участника. Во входных данных могут встретиться строки с совпадающими идентификационными номерами. Все числа во входном файле не превышают 10^5.

В выходной файл выведите исходный список в порядке убывания баллов. Если у некоторых участников одинаковые баллы, то их между собой нужно упорядочить в порядке возрастания идентификационного номера.

Input	Output
3 101 80 305 90 200 14	305 90 101 80 200 14
3 20 80 30 90 25 90	25 90 30 90 20 80

D. *Похожие массивы*

Назовём два массива похожими, если они состоят из одних и тех же элементов (без учёта кратности). По двум данным массивам выясните, похожие они или нет.

В первой строке содержится число N ($1 \leqslant N \leqslant 100000$) — размер первого массива. Во второй строке идет N целых чисел, не превосходящих по модулю 10^9 — элементы массива. Далее аналогично задается второй массив.

В решении запрещается использовать дополнительную память, размер которой зависит от размера входных данных.

Программа должна вывести слово `YES`, если массивы похожи, и слово `NO` в противном случае.

Input	Output
3 5 7 5 3 7 5 7	YES
3 5 7 5 4 7 5 7 57	NO

E. *Несамопересекающаяся незамкнутая ломаная*

Дано N различных точек на плоскости. Построить $(N - 1)$ - звенную несамопересекающуюся ломаную, проходящую через все эти точки.

В первой строчке программе подаётся на вход натуральное число N ($1 < N < 10^3$). Затем в N строках даётся по два целых числа, разделённых пробелом: абсцисса и ордината точки.

Требуется вывести N чисел от 0 до $N - 1$ (номера заданных точек) в таком порядке, чтобы соединённые в соответствующей последовательности точки образовывали несамопересекающуюся ломаную. Если возможных порядков построения ломаной несколько, можно вывести любой.

Input	Output
6 1 2 3 4 4 1 2 1 3 2 2 -1	3 4 2 5 0 1

F. *Обувной магазин*

В обувном магазине продается обувь разного размера. Известно, что одну пару обуви можно надеть на другую, если она хотя бы на три размера больше. В магазин пришел покупатель. Требуется определить, какое наибольшее количество пар обуви сможет предложить ему продавец так, чтобы он смог надеть их все одновременно.

Сначала вводится размер ноги покупателя (обувь меньшего размера он надеть не сможет), затем количество пар обуви в магазине и размер каждой пары. Размер — натуральное число, не превосходящее 100, количество пар обуви в магазине не превосходит 1000.

Выведите единственное число — максимальное количество пар обуви.

Input	Output
60 2 60 63	2
26 5 30 35 40 41 42	3

G. *Числа*

Саша и Катя учатся в начальной школе. Для изучения арифметики при этом используются карточки, на которых написаны цифры (на каждой карточке написана ровно одна цифра). Однажды они пришли на урок математики, и Саша, используя все свои карточки, показал число A, а Катя показала число B. Учитель тогда захотел дать им такую задачу, чтобы ответ на нее смогли показать и Саша, и Катя, каждый используя только свои карточки. При этом учитель хочет, чтобы искомое число было максимальным.

Во входном файле записано два целых неотрицательных числа A и B (каждое число в одной строке). Длина каждого из чисел не превосходит 100000 цифр.

Выведите одно число — максимальное целое число, которое можно составить используя как цифры первого числа, так и цифры второго числа. Если же ни одного такого числа составить нельзя, выведите -1.

Обратите внимание на тест, в котором у Саши и Кати нет общих цифр, кроме нуля.

Input	Output
280138 798081	8810
123 456	-1
12300 405006	0
13579 4321	31

H. *Анаграммы*

Слово называется анаграммой другого слова, если оно может быть получено перестановкой его букв.

Даны два слова на отдельных строках. Слова состоят из строчных латинских букв и цифр. Длины слов не превышают 10^6.

Требуется вывести `YES` — если введенные слова являются анаграммами друг друга, `NO` — если нет.

Input	Output
sharm marsh	YES
ananas nnaass	NO

I. *Клавиатура*

Всем известно, что со временем клавиатура изнашивается, и клавиши на ней начинают залипать. Конечно, некоторое время такую клавиатуру еще можно использовать, но для нажатий клавиш приходиться использовать большую силу.

При изготовлении клавиатуры изначально для каждой клавиши задается количество нажатий, которое она должна выдерживать. Если знать эти величины для используемой клавиатуры, то для определенной последовательности нажатых клавиш можно определить, какие клавиши в процессе их использования сломаются, а какие — нет.

Требуется написать программу, определяющую, какие клавиши сломаются в процессе заданного варианта эксплуатации клавиатуры.

Первая строка входного файла содержит целое число n ($1 \leqslant n \leqslant 100$) — количество клавиш на клавиатуре. Вторая строка содержит n целых чисел — $c_1, c_2, \ldots, c_n$, где $_i$ ($1 \leqslant c_i \leqslant 10^5$) — количество нажатий, выдерживаемых i-ой клавишей. Третья строка содержит целое число k ($1 \leqslant k \leqslant 10^5$) — общее количество нажатий клавиш, и последняя строка содержит k целых чисел p_j ($1 \leqslant p_j \leqslant n$) — последовательность нажатых клавиш.

В выходной файл необходимо вывести n строк, содержащих информацию об исправности клавиш. Если i-ая клавиша сломалась, то i-ая строка должна содержать слово `yes` (без кавычек), если же клавиша работоспособна — слово `no`.

Input	Output
5 1 50 3 4 3 16 1 2 3 4 5 1 3 3 4 5 5 5 5 5 4 5	yes no no no yes

J. *Форум*

Клуб Юных Хакеров организовал на своем сайте форум. Форум имеет следующую структуру: каждое сообщение либо начинает новую тему, либо является ответом на какое-либо предыдущее сообщение и принадлежит той же теме.

После нескольких месяцев использования своего форума юных хакеров заинтересовал вопрос — какая тема на их форуме наиболее популярна. Помогите им выяснить это.

В первой строке вводится целое число N — количество сообщений в форуме ($1 \leqslant N \leqslant 1000$). Следующие строки содержат описание сообщений в хронологическом порядке.

Описание сообщения, которое представляет собой начало новой темы, состоит из трёх строк. Первая строка содержит число 0. Вторая строка содержит название темы. Длина названия не превышает 30 символов. Третья строка содержит текст сообщения.

Описание сообщения, которое является ответом на другое сообщение, состоит из двух строк. Первая строка содержит целое число — номер сообщения, ответом на которое оно является. Вторая строка содержит текст сообщения.

Сообщения нумеруются, начиная с единицы. Ответ всегда появляется позже, чем сообщение, ответом на которое он является.

При подсчёте ответов на сообщения надо учитывать ответы всех уровней, не только первого (ответы на ответы и т.п.).

Длина каждого из сообщений не превышает 100 символов.

Выведите название темы, к которой относится наибольшее количество сообщений. Если таких тем несколько, то выведите первую в хронологическом порядке.

Input	Output
2 0 topic 1 body of message 1 0 topic 2 body of message 2	topic 1

K. *Иннокентий решает задачи*

Мальчик Иннокентий решает вступительную работу в летний математический лагерь. В ней N заданий, которые можно выполнять в произвольном порядке. Разные задачи требуют разного времени для решения. При этом известно, что если задание с номером i выполнять j-м по счету, Иннокентию потребуется $T_i \times j$ времени: чем больше думаешь, тем больше устаешь. Например, если начать с первой задачи, а затем выполнить вторую, то потребуется $T_1 \times 1 + T_2 \times 2$ времени, а если выполнить сначала вторую задачу, а затем первую — то $T_2 \times 1 + T_1 \times 2$.

Подскажите Иннокентию, в каком порядке нужно решать задачи, чтобы на выполнение всей работы ушло как можно меньше времени.

В первой строке вводится число N, во второй строке — N чисел через пробел $T_1, T_2, \ldots, T_N$, разделённые пробелами. Все числа целые и удовлетворяют следующим ограничениям: $0 < N \leqslant 10, 0 < T_i \leqslant 100$.

Требуется вывести сначала минимальное время, за которое можно решить все задачи, а затем — номера задач в том порядке, в котором их нужно решать, чтобы уложиться в это время. Все числа разделяются пробелами. Если решений несколько, нужно выдать любое из них.

Input	Output
2 2 3	7 2 1

L. *Такси*

После затянувшегося совещания директор фирмы решил заказать такси, чтобы развезти сотрудников по домам. Он заказал N машин — ровно столько, сколько у него сотрудников. Однако когда они подъехали, оказалось, что у каждого водителя такси свой тариф за 1 километр.

Директор знает, какому сотруднику сколько километров от работы до дома (к сожалению, все сотрудники живут в разных направлениях, поэтому нельзя отправить двух сотрудников на одной машине). Теперь директор хочет определить, какой из сотрудников на каком такси должен поехать домой, чтобы суммарные затраты на такси (а их несёт фирма) были минимальны.

Первая строка входных данных содержит натуральное число N ($1 \leqslant N \leqslant 50000$) — количество сотрудников компании (совпадающее с количеством вызванных машин такси). Далее записано N чисел, задающих расстояния в километрах от работы до домов сотрудников компании (первое число — для первого сотрудника, второе — для второго и т.д.). Все расстояния — положительные целые числа, не превышающие 10^6. Далее записано еще N чисел — тарифы за проезд одного километра в такси (первое число — в первой машине такси, второе — во второй и т.д.). Тарифы выражаются положительными целыми числами, не превышающими 10^6.

В первой строке программа должна вывести минимальную стоимость доставки всех сотрудников. Во второй строке программа должна вывести N чисел. Первое число — номер такси, в которое должен сесть первый сотрудник, второе число — номер такси, в которое должен сесть второй и т.д., чтобы суммарные затраты на такси были минимальны. Если вариантов рассадки сотрудников, при которых затраты минимальны, несколько, выведите любой из них.

Input	Output
3 10 20 30 50 20 30	1700 1 3 2
5 10 20 1 30 30 3 3 3 2 3	243 5 1 3 2 4

M. *Большое число*

Маша написала на длинной полоске бумаги большое число и решила похвастаться своему старшему брату Серёже этим достижением. Но только она вышла из комнаты, чтобы позвать брата, как её сестра Дуня вбежала в комнату и разрезала полоску бумаги на несколько частей. В результате на каждой части оказалось одна или несколько идущих подряд цифр. Теперь Маша не может вспомнить, какое именно число она написала. Только помнит, что оно было очень большое. Чтобы утешить сестру, Серёжа решил выяснить, какое максимальное число могло быть написано на полоске бумаги перед разрезанием. Помогите ему.

Входные данные представляют собой одну или более строк, каждая из которых содержит последовательность цифр. Количество строк во входном файле не превышает 100, каждая строка содержит от 1 до 100 цифр. Гарантируется, что хотя бы в одной строке первая цифра отлична от нуля.

Выведите одну строку — максимальное число, которое могло быть написано на полоске перед разрезанием.

Input	Output
2 20 004 66	66220004
3	3

Указание: здесь может быть полезно реализовать свою сортировку (можно даже квадратичную, размеры массива вполне позволяют) и придумать для неё свой алгоритм проверки того, надо ли менять элементы (строки) местами или нет.

N. *Решение задач*

Вася готовится к олимпиаде. Учитель дал ему N ($1 \leqslant N \leqslant 10^5$) задач для тренировки. Для каждой из этих задач известно, каким умением a_i нужно обладать для её решения. Это означает, что если текущее умение Васи больше либо равно заданного умения для задачи, то он может ее решить.

Кроме того, после решения i-й задачи Васино умение увеличивается на число b_i.

Исходное умение Васи равно A. Решать данные учителем задачи он может в произвольном порядке. Какое максимальное количество задач он сможет решить, если выберет самый лучший порядок их решения?

Сначала вводятся два целых числа N, A ($1 \leqslant N \leqslant 10^5, 0 \leqslant A \leqslant 10^9$) — количество задач и исходное умение. Далее идут N пар целых чисел a_i, b_i ($1 \leqslant a_i \leqslant 10^9, 1 \leqslant b_i \leqslant 10^9$) — соответственно сколько умения нужно для решения i-й задачи и сколько умения прибавится после её решения.

Выведите одно число — максимальное количество задач, которое Вася сможет решить.

Input	Output
3 2 3 1 2 1 1 1	3

O. *Покрытие отрезками*

Даны N отрезков на прямой и координаты M точек. Для каждой из точек отрезка $[1, L]$ определите — каким количеством данных отрезков они покрываются?

Во входном файле даны сначала L, N, M ($1 \leqslant L \leqslant 10000, 1 \leqslant N \leqslant 10^4, 1 \leqslant M \leqslant 10^5$).

Далее идут N пар чисел $l \leqslant r$ от 1 до L — левые и правые концы отрезков. Затем перечислены M чисел от 1 до L.

Выведите M чисел — количество отрезков, покрывающую каждую из указанных M точек.

Input	Output
39 4 7 3 21 3 15 2 20 3 17 4 17 33 5 9 25 37	4 3 0 4 4 0 0

Указание: Для решения задачи можно сохранить в массиве все границы отрезков (и левые и правые), сохранив информацию о типе границы. Например, для каждой границы можно хранить пару чисел $(x, \pm 1)$: $+1$ для левой границы и -1 для правой. Упорядочить массив по возрастанию координаты (подумайте, важно ли упордяочивать по типу границы?).

Пользуясь тем, что $L < 10^4$ (т.е. невелико), посчитаем **для каждой точки** от 1 до L количество накрывающих её отрезков. Затем выведем результат для тех точек, которые даны в условии.

P. *Закраска прямой*

На числовой прямой окрасили N отрезков. Известны координаты левого и правого концов каждого отрезка (L_i и R_i). Найти длину окрашенной части числовой прямой.

В первой строке находится число N, в следующих N строках — пары L_i и R_i. L_i и R_i — целые, $-10^9 \leqslant L_i \leqslant R_i \leqslant 10^9, 1 \leqslant N \leqslant 15000$.

Вывести одно число — длину окрашенной части прямой.

Input	Output
1 10 10	0
2 10 20 20 40	30

Указание: Можно упорядочить все отрезки по координате левого конца. Сделать пару переменных `L_bound` и `R_bound`: левую и правую границы текущего непрерывного отрезка, а также переменную с текущим значением ответа на вопрос задачи. В начале `L_bound` и `R_bound` — это границы первого отрезка.

Далее в цикле надо просмотреть слева направо остальные отрезки и обновлять переменные `L_bound`, `R_bound` и ответ. Например, если левый конец очередного отрезка больше, чем `R_bound`, то в этом случае надо обновить `L_bound` (начался новый кусок, левее уже ничего не будет, поскольку отрезки упорядочены по левой границе).

Q. *Задача о минимальном покрытии*

На прямой задано некоторое множество отрезков с целочисленными координатами концов $[L_i, R_i]$. Выберите среди данного множества подмножество отрезков, целиком покрывающее отрезок $[0, M]$, (M — натуральное число), содержащее наименьшее число отрезков.

В первой строке указано число M ($1 \leqslant M \leqslant 5000$). В каждой последующей строке записана пара чисел L_i и R_i ($|L_i|, |R_i| \leqslant 5 \cdot 10^4$), задающая координаты левого и правого концов отрезков. Список завершается парой нулей. Общее число отрезков не превышает 10^5.

В первой строке выходного файла выведите минимальное число отрезков, необходимое для покрытия отрезка $[0, M]$.

Далее выведите список покрывающего подмножества, упорядоченный по возрастанию координат левых концов отрезков. Список отрезков выводится в том же формате, что и во входе. Завершающие два нуля выводить не нужно.

Если покрытие отрезка $[0, M]$ исходным множеством отрезков $[L_i, R_i]$ невозможно, то следует вывести единственную фразу `No solution`.

Input	Output
1 -1 0 -5 -3 2 5 0 0	No solution
1 -1 0 0 1 0 0	1 0 1
500 -1000 13 5000 50000 15 499 13 18 18 500 15 18 -5000 5 -8000 -3 0 0	3 -1000 13 13 18 18 500

Указание: Здесь можно упорядочить все отрезки по координате левого конца. Сделать пару переменных `current_R = 0` и `max_R = 0`: текущую правую границу и правую границу интервала, занятого выбранным на данный момент набором отрезков.

Далее надо научиться выбирать очередной отрезок, который должен попасть в искомый оптимальный набор. Это будет отрезок, который начинается не позднее, чем `max_R` и имеет самое большое значение координаты правого конца.

Если сразу после того, как в оптимальный набор выбран очередной отрезок, начало следующего оказалось правее `mac_R`, то необходимого накрытия не существует (остальные отрезки начинаются ещё правее).

R. *Выравнивание забора*

Забор на дачном участке Глеба состоит из деревянных планок, каждая высотой h_i. Для того, чтобы использовать одну из сторон забора для прыжков в высоту, его необходимо выровнять — сделать все планки одной и той же высоты.

Приглашённые рабочие объявили, что изменение высоты i-ой планки на один метр в любую сторону (уменьшения или увеличения) будет стоить a_i рублей. При этом изменять высоту на нецелую величину рабочие не умеют.

Помогите Глебу выровнять забор, заменив все высоты на какую-то одну, так, чтобы общая плата за работы была минимальна.

Требуется написать программу, которая будет определять оптимальную высоту нового забора и сумму денег, в которую обойдется ее выравнивание.

В первой строке входного файла задано одно число n ($1 \leqslant n \leqslant 10^5$) — количество планок в заборе. Во второй строке задано n целых чисел h_i ($1 \leqslant h_i \leqslant 10^6$), где h_i — высота i-ой планки. В третьей строке так же заданы n целых чисел a_i ($1 \leqslant a_i \leqslant 10^6$) — цена изменения высоты i-ой планки на один метр.

Выведите в выходной файл два целых числа: конечную высоту забора и сумму денег, которую придется потратить на его выравнивание. Если вариантов выравнивания одинаковой стоимости несколько, то выведите минимально возможную высоту.

Input	Output
6 6 7 8 8 7 7 10 6 3 1 1 4	7 14
5 7 5 7 9 8 10 8 7 8 5	7 37
5 8 5 10 9 7 2 5 4 8 4	9 34

S. *Белоснежка и N гномов*

У Белоснежки N гномов, и все они очень разные. Она знает, что для того, чтобы уложить спать i-го гнома нужно a_i минут, и после этого он будет спать ровно b_i минут. Помогите Белоснежке узнать, может ли она получить хотя бы минутку отдыха, когда все гномы будут спать, и если да, то в каком порядке для этого нужно укладывать гномов спать.

Например, пусть есть всего два гнома, $a_1 = 1, b_1 = 10, a_2 = 10, b_2 = 20$. Если Белоснежка сначала начнет укладывать первого гнома, то потом ей потребуется целых 10 минут, чтобы уложить второго, а за это время проснется первый. Если же она начнет со второго гнома, то затем она успеет уложить первого и получит целых 10 минут отдыха.

Первая строка входного файла содержит число N ($1 \leqslant N \leqslant 10^5$), вторая строка содержит числа $a_1, a_2, \ldots a_n$, третья — числа $b_1, b_2, \leqslant b_n$ ($1 \leqslant a_i, b_i \leqslant 10^9$).

Выведите в выходной файл N чисел — порядок, в котором нужно укладывать гномов спать. Если Белоснежке отдохнуть не удастся, выведите число -1.

Input	Output
2 1 10 10 20	2 1
2 10 10 10 10	-1

T. *Гражданская оборона*

Вдоль длинной прямой улицы расположены N домов. Вдоль той же дороги расположены M киосков с мороженым, которое так любят жители города.

Чтобы жители меньше уставали, необходимо для каждого дома определить ближайш к нему киоск с мороженым.

В первой строке вводится число N — количество домов ($1 \leqslant N \leqslant 10^5$). Вторая строка содержит N различных целых чисел, i-е из этих чисел задает расстояние от начала дороги до i-го дома. В третьей строке входных данных задается число M — количество киосков с мороженым ($1 \leqslant M \leqslant 10^5$). Четвертая строка содержит M различных целых чисел, i-е из этих чисел задает расстояние от начала дороги до i-го киоска. Все расстояния положительны и не превышают 10^9. Киоск с мороженым вполне может располагаться одном из домов.

Выведите N чисел — для каждого дома выведите номер ближайшего к нему киоска с мороженым. Киоски пронумерованы от 1 до M в том порядке, в котором они заданы во входных данных. Если два киоска равноудалены, можно вывести номер любого из них.

Input	Output
4 1 2 6 10 2 7 3	2 2 1 1

U. *Триангуляция*

Вася нарисовал выпуклый N-угольник и провел в нем несколько диагоналей таким образом, что никакие две диагонали не пересекаются внутри N-угольника. Теперь он утверждает, что весь N-угольник оказался разбит на треугольники. Напишите программу, которая проверяет истинность Васиного утверждения.

Входные данные

В первой строке вводятся два числа N и M. N — количество вершин N-угольника ($3 \leqslant N \leqslant 1000$) и M — количество диагоналей, проведенных Васей. Далее на вход программы поступают M пар чисел, задающих диагонали (каждая диагональ задается парой номеров вершин, которые она соединяет). Гарантируется, что каждая пара чисел задает диагональ (то есть две вершины различны и не являются соседними), а также что никакие две пары не задают одну и ту же диагональ. Никакие две диагонали не пересекаются внутри N-угольника. Вершины N-угольника нумеруются числами от 1 до N.

Выходные данные

Если Васино утверждение верно, то программа должна выводить единственное число 0. В противном случае необходимо вывести сначала число K — количество вершин в какой-нибудь не треугольной части. Далее должно быть выведено K чисел — номера вершин исходного N-угольника, которые являются вершинами этой K-угольной части в порядке обхода этой части.

Input	Output
3 0	0
4 1 1 3	0
6 2 1 3 5 3	4 1 3 5 6