Разбор заданий отборочного этапа олимпиады по информатике «СПЕКТР» 2024г.

Задача 1. Дано 4 числа. Требуется выбрать из них такое, которое будет наиболее близко к среднему арифметическому оставшихся 3-х чисел по абсолютной величине. Если таких чисел несколько, то взять меньшее.

Входные данные

4 целых числа (-2 * 10^10 <= a, b, c, d <= 2 * 10^10) в одной строке, разделённые пробелом.

Выходные данные

Целое число, наиболее близкое к среднему арифметическому оставшихся чисел.

Пример

1 2 3 4

Ответ

2

Пояснение

Число	Среднее остальных	Разница
1	3	2
2	2,67	0,67
3	2,67	0,67
4	2	2

Имеем 2 подходящих числа с наименьшим отклонением от среднего. Выбираем меньшее - 2.

Разбор

Для решения задачи можно использовать 2 подхода:

1. Отсортировать полученные числа и выбирать второе или третье по минимальной разнице.
2. Перебрать все возможные варианты решения и выбрать наилучший.

Ниже приводятся оба варианта на языке Python.

```
# 1
buf = sorted([int(x) for x in input().split()])

if (abs((sum(buf) - buf[1]) / 3 - buf[1]) <
        abs((sum(buf) - buf[2]) / 3 - buf[2])):
    ans = abs((sum(buf) - buf[1]) / 3 - buf[1])

else:
    ans = abs((sum(buf) - buf[2]) / 3 - buf[2])

print(ans)


# 2
buf = [int(x) for x in input().split()]

sm = sum(buf)
best = (10 ** 11, buf[0])

for x in buf:
    if abs((sm - x) / 3 - x) < best[0]:
        best = abs((sm - x) / 3 - x), x

print(best[1])
```

Задача 2. Дано две строки s и t, состоящих из больших латинских букв (A..Z). Требуется определить, можно ли получить из строки s строку t если выбрать в исходной строке некоторые символы. Также можно заменить пару одинаковых символов строки s на один любой другой символ.

Входные данные

2 строки из больших латинских букв (A..Z) по одному на каждой строке. Количество символов в строках не превосходит 1 * 10^6.

Выходные данные

"Yes" - если можно получить из строки s выбором и заменой строку t.

"No" - если не существует возможности получить из строки s строку t.

Пример

№	Входные данные	Ответ
1	PRIVET VETPIR	Yes
2	PRIVETEE VETPIA	Yes
3	PRIVETEEV VETPIAB	No

Пояснение

Во втором примере можно набрать все символы кроме R, но у нас есть пара лишних символов E, которую можно заменить на недостающий символ R.

Разбор

Для решения задачи можно подсчитать количество символов в первой и второй строках. Затем вычитаем из числа каждого символа первой строки соответствующее количество во второй. Если получается отрицательное число, то этих символов не хватает, и их надо добавить, а если положительное, то эти символы избыточные и можно использовать их для замены половины недостающих.

Ниже приводятся вариант на языке Python.

```
first_line = input()
second_line = input()

# can use Counter

# counting a number of symbols in the first line
nums_first = {}
for letter in first_line:
    nums_first[letter] = nums_first.get(letter, 0) + 1

# counting a number of symbols in the first line
nums_second = {}
for letter in second_line:
    nums_second[letter] = nums_second.get(letter, 0) + 1

# can compare length of the strings before

# get a difference between first and second lines
for letter, value in nums_second.items():
    nums_first[letter] = nums_first.get(letter, 0) - value
```

```
nums = 0
for value in nums_first.values():
    if value < 0:
        # deficit
        nums += value
    else:
        # surplus
        nums += value // 2

print('Yes' if nums >= 0 else 'No')
```

Задача 3. Дан массив из N чисел. В данном массиве можно поменять местами любую пару соседних элементов. Требуется определить минимальное количество пар чисел, стоящих не по возрастанию, если допустимо применить одну операцию перестановки соседних элементов.

Входные данные

В первой строке указывается N - количество элементов массива (1 < N <= 1 * 10^6). Во второй строке через пробел задано N целых чисел (-1 * 10^9 <= ai <= 1 * 10^9)

Выходные данные

Минимальное число не упорядоченных по возрастанию пар соседних элементов, если допустимо выполнить одну операцию перестановки соседних элементов.

Пример 1

№	Входные данные	Ответ
1	4 1 3 2 4	0
2	4 2 1 4 3	1

Разбор

Простой наивный способ поочерёдного обмена всех пар элементов реализуется совсем просто, но, к сожалению, не проходит по времени для больших N. Ниже приведена таблица тестирования для 5 различных размеров (Intel(R) Core(TM) i7-14650HX 2.20 GHz; 32.0 GB).

```
 # |     n    |  solv   | slow
 1 |       3  |     0.0 |     0.0
 2 |      10  |     0.0 |     0.0
 3 |     100  |     0.0 |   0.001
 4 |    1000  |     0.0 |   0.029
 5 |   10000  |   0.016 |     4.1
solv = 0.016
solv_slow = 4.130
```

Уже при 10 000 элементах решение не укладывается в ограничения. Квадратичное время работы алгоритма даёт о себе знать.

Проблема заключается в том, что в наивном решении заново перепроверяются все пары, хотя изменение затрагивает только 4 элемента (2 меняемых местами и соседние слева и справа от пары).

Для быстрого решения достаточно посчитать общее количество инверсий, а затем пройти по всем парам и посмотреть, к чему может привести обмен элементов в паре. Время работы остаётся линейным.

Для удобства обработки в начало и конец массива были добавлены максимальный и минимальный элементы, что не изменяет число инверсий.

Ниже приводятся оба варианта реализации на языке Python.

```python
def solv_slow(n, buf):
    best = n

    t = 0
    for j in range(n):
        buf[t], buf[j] = buf[j], buf[t]

        k = 0
        for i in range(n - 1):
            if buf[i] >= buf[i + 1]:
                k += 1
        best = min(best, k)

        buf[t], buf[j] = buf[j], buf[t]
        t = j

    return best


def solv(n, buf):
    if n == 2:
```

```
        return 0

    k = 0
    for i in range(n - 1):
        if buf[i] >= buf[i + 1]:
            k += 1
    best = k

    buf = [min(buf) - 1] + buf + [max(buf) + 1]

    for i in range(1, n):

        was = 0
        temp = buf[i - 1: i + 3]
        for j in range(3):
            if temp[j] >= temp [j + 1]:
                was += 1

        # if swap buf[i] and buf[i + 1]:
        will = 0
        temp = [buf[i - 1], buf[i + 1], buf[i], buf[i + 2]]
        for j in range(3):
            if temp[j] >= temp [j + 1]:
                will += 1

        # update
        best = min(best, k - was + will)

    return best


def main():
    n = int(input())
    buf = [int(x) for x in input().split()]

    print(solv(n, buf))
    # print(solv_slow(n, buf))


main()
```