Средний уровень

Условие: совсем недавно Петя решил начать записывать обучающие видео для сдающих ЕГЭ по математике в 2025 году, но столкнулся с такой проблемой: чтобы его голос звучал приятно нужно было добиться частоты дискретизации микрофона хотя бы в 48 КГц при разрешении в 16 бит. И если последнее было для него не проблемой, то с частотой была беда – его устройство не могло выдать больше 32 КГц. Так, как Петя знает, что вы программист, то он решил сэкономить и предложил вам создать некоторый алгоритм масштабирования аудиозаписей. Его суть очень проста:

1. Берём два подряд идущих значения изначальной аудиозаписи;
2. Находим корень из абсолютной разности двух чисел и округляем его вниз;
3. Число, полученное на втором этапе, прибавляем ко второму элементу. Оно будет результатом масштабирования;
4. Затем, алгоритм берёт два **следующих** элемента и начинается заново.

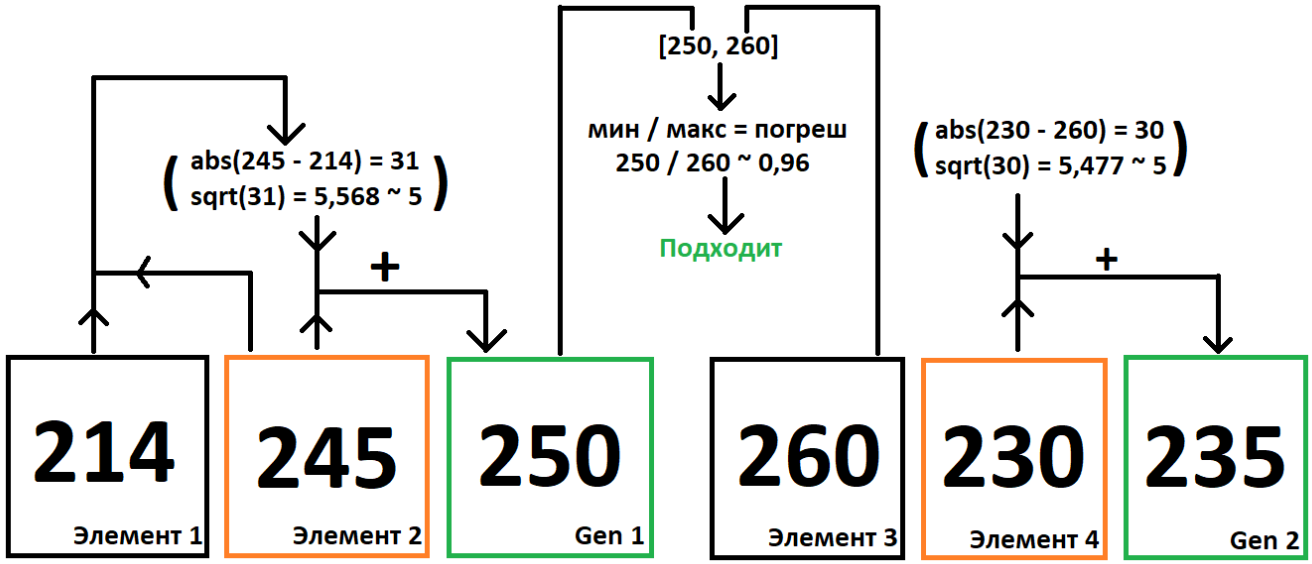
! Прошу заметить, что обрабатывать последнюю пару не требуется

Ваша задача, масштабировать предоставленную Петей аудиозапись и найти два значения: максимальное абсолютное отклонение (погрешность) сгенерированного значения от следующего (то есть, ваше сгенерированное значение сравнивается со следующим значением, которое было после пары элементов), а также количество сгенерированных значений уровень отклонения (погрешности), которых от следующего значения не больше 5%.

Решение и файл: в GitHub репозитории в файлах “*17_medium.txt*” и “*17_medium.py*”

Ответ: 267 1560

Графическое Пояснение:



```python
# floor - функция для округления дробных чисел вниз
from math import floor
# Открываем файл
f = open("17_medium.txt")
# Все данные записываем в новый массив
# Генератор преобразует входной поток в числа
data = [int(i) for i in f.readlines()]
# mmax - максимальное значение погрешности
# count - количество генераций с отклонением меньше 5%
mmax, count = 0, 0
# закрытие файла
f.close()
''' Перебираем парочками с шагом 2. Такой шаг обеспечивает
    работу не с подряд идущими, а следующими друг за другом парами. '''
for i in range(1, len(data) - 2, 2):
    ''' Generarion - генерируемое значение.
        abs возвращает модуль разности двух элементов, а
        оператор ** при значении 0.5 возвращает корень числа '''
    Generation = data[i] + floor(abs(data[i] - data[i - 1]) ** 0.5)
    # Delta - погрешность генерации. Минимальное делим на максимальное
    Delta = min(Generation, data[i + 1]) / max(Generation, data[i + 1])
    ''' Максимальное значение погрешности -
        результат разности генерации и первого элемента следующей пары '''
    mmax = max(mmax, abs(data[i + 1] - Generation))
    # Проверяем, проходит ли погрешность
    if Delta >= 0.95:
        count += 1
# Возвращаем ответ
print(mmax, count)
```