Задача 27 (алгоритмы обработки последовательностей от простого к сложному)

Начнём с самых простых задач.

Первая часть условия всех задач, рассмотренных далее, одинакова: "*С клавиатуры вводится натуральное число N* ($1 \le N \le 10\,000$)*, а затем в следующих N строках – N целых чисел, по одному в каждой строке.*"

Конкретное содержание задачи уточняется в каждом отдельном случае.

Решения будут приводиться на языке Python. Шаблон типовой программы на языке Python, выглядит так:

```
n = int(input())  # ввод количества элементов последовательности
for i in range(0, n):
    x = int(input())  # обработка значения x
print(...)  # вывод результата
```

Задача 1. *Требуется вывести одно неотрицательное число – количество пар с чётной суммой, образованных различными элементами последовательности.*

Решение.

Чтобы не учитывать одну и ту же пару дважды, будем рассматривать только упорядоченные пары (a_i, a_j), для которых $i < j$.

В данной задаче очередное число, прочитанное из входного потока, может образовывать пары с *любым* из предыдущих элементов последовательности. Будем говорить, что все предыдущие элементы принадлежат *множеству выбора пары* для следующего элемента.

Подходящими будем называть пары, которые удовлетворяют условиям задачи. В этой задаче подходящими будут те пары, для которых сумма элементов чётна.

Обозначим через Q_k количество подходящих пар для подпоследовательности, состоящей из первых k элементов полной последовательности. Если очередное полученное число x чётно, оно образует подходящие пары (с чётной суммой) только с предыдущими *чётными* элементами последовательности, а нечётное значение x – только с предыдущими *нечётными* элементами. Поэтому нам нужно хранить на каждой итерации количество предыдущих чётных чисел E_{k-1} и количество предыдущих нечётных чисел O_{k-1}.

В блоке инициализации присваиваем переменным Q, O, E нулевые значения. После ввода очередного значения x из входного потока нужно обновить Q, а затем обновить E или O, в зависимости от чётности x:

```
n = int(input())
q = 0
e = 0
o = 0  # инициализация переменных
for i in range(0, n):
    x = int(input())
    if x % 2 == 0:
        q += e
        e += 1
    else:
        q += o
        o += 1
print(q)
```

Задача 2. *Требуется вывести одно неотрицательное число – количество пар с суммой, равной* $S = 20$*, образованных различными элементами последовательности.*

Решение.

Применим общий подход: получив из входного потока новое значение x, определим, сколько подходящих пар оно может образовать с предыдущими элементами последовательности. Обозначим через Q_k количество подходящих пар в подпоследовательности, состоящей из первых k чисел полной последовательности.

Напомним, что по условию (см. начало этого раздела) все элементы последовательности целые и положительные. Поэтому числа, большие или равные *S*, не могут образовать ни одной подходящей пары.

Если число *x* меньше *S*, оно образует подходящие пары со всеми предыдущими элементами, равными *S* – *x*. Следовательно, нам нужно хранить массив значений *D*, где *D*[*i*] – это количество предыдущих элементов последовательности, равных *i* ($1 \le i < S$).

На этапе инициализации значение переменной *Q* и все элементы массива *D* обнуляются. Они обновляются на очередной итерации только тогда, когда из входного потока получено число *x*, которое меньше *S*:

```
s=20
d = [0]*s #объявление переменных
n = int(input())
q = 0
for i in range(0, n):
    x = int(input())
    if x < s:
        q += d[s - x]
        d[x] += 1
print(q)
```

Задача 3. *Требуется вывести одно неотрицательное число – количество пар с суммой, меньшей или равной S =* 20*, образованных различными элементами последовательности.*

Решение.

Будем использовать обозначения, введённые при решении **Задачи 2**. В отличие от неё, в этой задаче подходящими считаются пары с суммой не только равной, но и меньшей, чем *S*. Поэтому число *x* < *S*, полученное из входного потока на очередной итерации, образует пары со всеми предыдущими числами в диапазоне [1, *S*–*x*]. Следовательно, в программе потребуется суммирование элементов массива *D* от *D*[1] до *D*[*S*–*x*] включительно:

```
s = 20
n = int(input())
q = 0
d = [0]*s  #инициализация переменных
for i in range(0, n):
    x = int(input())
    if x < s:
        for k in range(0, s-x+1):
            q += d[k]
        d[x] += 1
print(q)
```

Задача 4. *Требуется вывести одно неотрицательное число – количество пар с суммой, большей, чем S =* 20*, образованных различными элементами последовательности.*

Решение.

Можно свести эту задачу к предыдущей. Если мы нашли количество *Q* всех пар с суммой, меньшей или равной *S*, то количество пар с суммой, большей *S*, легко найти как *N* (*N*–1)/2 – *Q*, где *N* (*N*–1)/2 – количество всех возможных пар.

Давайте рассмотрим второй способ – будем вычислять желаемое значение непосредственно.

Пусть из входного потока прочитано значение *x*. Оно может образовать подходящие пары со всеми предыдущими числами, большими, чем *S* – *x*. Заметим, что числа, большие или равные *S*, образуют подходящие пары со всеми предыдущими числами (на *i*-й итерации получается *i*–1 таких пар), а «маленькие» числа (меньшие *S*) – не со всеми. Для «маленько-го» числа *x*, введённого на *i*-й итерации цикла, количество подходящих пар вычисляется как $i - 1 - (D[1] + D[2] + \ldots + D[S-x])$, где *D*[*k*] – количество предыдущих элементов последовательности, равных *i*. Получаем следующую программу:

```
s = 20
d = [0]*s
n = int(input())
q = 0
for i in range(0, n):
    x = int(input())
    q += i
    if x < s:
        for k in range(1, s-x+1):
            q -= d[k]
        d[x] += 1
print(q)
```

Задача 5. *Рассматриваются все пары различных элементов последовательности, в которых хотя бы одно число кратно F = 17. Из всех таких пар нужно выбрать пару с максимальной суммой и вывести на экран оба элемента, образующие эту пару. Если таких пар несколько, то нужно вывести любую из них. Если таких пар нет, то нужно вывести два нуля.*

Решение.

Поскольку в паре должно быть хотя бы одно число, делящееся1 на *F*, возможность образования пар с предыдущими элементами последовательности будет определяться делимостью введённого значения *x* на *F*.

Если число *x* делится на *F*, оно может образовать пару с любым из предыдущих элементов. Так как нас интересует подходящая пара с максимальной суммой, нужно хранить максимальное из всех предыдущих чисел; для этого будем использовать переменную *max*.

Если число *x* не делится на *F*, оно может образовать подходящую пару только с теми из предыдущих элементов, которые делятся на *F*. Поэтому будем также сохранять (в переменной *maxF*) максимальное из предыдущих значений, делящихся на *F*.

В переменных *left* и *right* хранятся два числа, которые на каждой итерации представляют собой текущее решение задачи – подходящую пару с максимальной на данный момент суммой. Если введённое значение *x* позволяет образовать подходящую пару с большей суммой, чем *left* + *right*, значения этих переменных обновляются. Получаем следующий код для программы:

Задача 6. (Статград апрель) *Рассматриваются все пары различных элементов последовательности, разность которых делится на R = 60 и при этом хотя бы один элемент из пары больше B = 80. Требуется вывести одно неотрицательное число – количество пар, удовлетворяющих указанным условиям.*

Решение.

Разность двух чисел будет делиться на *R* тогда и только тогда, когда эти числа при делении на *R* дают равные остатки. Будем сохранять в массиве *D* количество чисел, дающих при делении на *R* все возможные остатки от 0 до *R*–1. Значение элемента *D*[*i*] – это количество предыдущих элементов последовательности, которые при делении на *R* дают в остатке *i*.

Если полученное из входного потока значение *x* меньше, чем *B*, оно может образовать пары не со всеми предыдущими числами с равными остатками, а только с теми, которые больше *B*. Поэтому нужен второй массив *G*, в котором элемент *G*[*i*] – это количество предыдущих элементов последовательности, **больших *B***, которые при делении на *R* дают в остатке *i*.

Получив из входного потока очередное значение *x*, определяем остаток от его деления на *R:*

ost = x % R

Если *x* > *B* (число *x* «большое»), то увеличиваем счётчик пар *Q* на величину *D*[*r*] (*x* образует пары со всеми предыдущими элементами с тем же остатком). Если *x* ≤ *B*, то увеличиваем *Q* на величину *G*[*r*] (учитываем только «большие» предыдущие числа с тем же остатком).

Значение *D*[*r*] увеличивается на 1 при каждой итерации цикла, а значение *G*[*r*] – только тогда, когда *x* > *B*. Получаем следующий код для программы:

Задача 7. (досрочный вариант ЕГЭ-2020) *Рассматриваются все пары различных элементов последовательности, разность которых чётна и при этом хотя бы одно число из пары кратно F = 17. Из всех таких пар нужно выбрать пару с максимальной суммой и вывести на экран оба элемента, образующие эту пару. Если таких пар несколько, то нужно вывести любую из них. Если таких пар нет, то нужно вывести два нуля.*

Решение:

По существу, эта задача является немного усложнённым вариантом рассмотренной ранее **Задачи 5**, в которую добавлено ещё одно условие – чётность разности чисел, образующих пару. Решение этой задачи с использованием комбинаторики становится очень громоздким из-за множества вложенных условных операторов, поэтому применим нашу идею о нахождении максимальной пары на конкретном шаге.

Применим общий подход: предположим, что очередное значение *x* введено из входного потока и проверим, какие подходящие пары оно может образовать с предыдущими элементами.

С точки зрения формирования подходящих пар все элементы входной последовательности можно разделить на четыре группы:

1. делятся на 17, чётные;
2. делятся на 17, нечётные;
3. не делятся на 17, чётные;
4. не делятся на 17, нечётные.

Для получения чётной разности можно составлять пару из двух чётных или из двух нечётных чисел. Кроме того, хотя бы одно из них должно быть кратно 17, то есть входить в группу 1 или в группу 2. Таким образом, допустимы следующие комбинации групп:

1 + (1 или 3), 2 + (2 или 4), 3 + 1 и 4 + 2.

Поскольку нас интересует пара с максимальной суммой, нужно хранить максимумы предыдущих элементов последовательности, которые могут образовать пару с элементами каждой из четырёх групп. В переменных maxh и maxn будем хранить максимумы среди чётных и нечетных чисел соответственно, а в переменных maxh17 и maxn17 будем хранить максимумы среди четных и нечетных чисел кратных 17 соответственно. В переменных left и right будем хранить максимальную пару на данном шаге.

Задача 8. *Рассматриваются все пары различных элементов последовательности, произведение которых кратно F = 39. Из всех таких пар нужно выбрать пару с максимальной суммой и вывести на экран оба элемента, образующие эту пару. Если таких пар несколько, то нужно вывести любую из них. Если таких пар нет, то нужно вывести два нуля.*

Решение.

В отличие от предыдущих задач, здесь важно, что число *F* имеет всего два простых делителя, 3 и 13 (при большем количестве простых делителей решение существенно усложняется). Поэтому все числа с точки зрения выполнения условия задачи можно разделить на четыре группы:

1. делятся на 13, делятся на 3;
2. делятся на 13, не делятся на 3;
3. не делятся на 13, делятся на 3;
4. не делятся на 13, не делятся на 3.

Для того чтобы произведение двух чисел было кратно 39, необходимо и достаточно, чтобы оно содержало сомножители 3 и 13. Поэтому возможны следующие комбинации групп:
1 + (1 или 2 или 3 или 4), 2 + (1 или 3), 3 + (1 или 2) и 4 + 1.

Будем использовать следующие обозначения:

max – максимум из всех чисел (для образования пар с числами из группы 1);

max3 – максимум из чисел, которые делятся на 3 (для образования пар с числами из группы 2);

max13 – максимум из чисел, которые делятся на 13 (для образования пар с числами из группы 3);

max39 – максимум из чисел, которые делятся как на 13, так и на 3 (для образования пар с числами из группы 4).

Задача 9. *Рассматриваются все пары различных элементов последовательности,* ***номера которых в последовательности отличаются не менее, чем на L = 4****, а произведение кратно F = 39. Из всех таких пар нужно выбрать пару с максимальной суммой и вывести на экран оба элемента,*

образующие эту пару. Если таких пар несколько, то нужно вывести любую из них. Если таких пар нет, то нужно вывести два нуля. Предполагается, что N > L.

Решение.

При решении предыдущих задач мы предполагали, что множество выбора пары для очередного значения *x* – это все предыдущие элементы последовательности. В данной задаче это не так: согласно условию, очередной элемент последовательности не может образовывать пары с ближайшими *L* – 1 предшествующими элементами. Правда, потом все они войдут во множество выбора пары, но до этого момента их нужно где-то сохранить. С этой целью будем использовать вспомогательный массив *A*.

Первые *L* элементов последовательности не могут образовывать пары с предыдущими элементами, поскольку для таких пар разность номеров элементов меньше, чем *L*. Поэтому сначала нужно просто заполнить вспомогательный массив первыми *L* значениями из входного потока.

В начале программы создаётся пустой массив *A*, который сразу заполняется первыми *L* значениями из входного потока.

Основной цикл теперь выполняется *N* – *L* раз, а не *N*, как было ранее, так как первые *L* элементов последовательности уже прочитаны и добавлены в массив до начала этого цикла.

Затем нужно добавить *первый элемент массива* к множеству выбора пар, так как следующий элемент последовательности (пока не прочитанный) уже может образовать с ним пару.

Затем читаем новое значение *x* из входного потока, проверяем пары, которые с ним можно составить, а после этого записываем его на место использованного элемента в массиве, после чего сдвигаем нумерацию во вспомогательном массиве: *ia = (ia + 1) % L*, не забывая что нумерация элементов в массиве от 0 до L-1.