Немного теории.

1. Функция **bin(s)** формирует строковое представление числа s в двоичном виде.

Например:

- `bin(43) #Двоичное представление числа 43`
- `'0b101011' #первые два символа (0b) обозначают двоичное представление, их нужно убрать.`

Итого, чтобы получить двоичную строку, мы должны взять срез, начиная с третьего символа (с индексом 2):

- `bin(43)[2:]`
- `'101011'`

2. Функция **sorted**(s) возвращает отсортированную последовательность

- `sorted('zdffgffesdfaargg')`
- `['a', 'a', 'd', 'd', 'e', 'f', 'f', 'f', 'f', 'f', 'g', 'g', 'g', 'r', 's', 'z']`

3. Функция `int(s,2)`, преобразует строковую двоичную запись числа в десятичное число

- `int('101011',2)`
- `43`

Что повторить по Python перед решением задания 5

- **Целые числа и системы счисления:** `bin(n)`, `int(s, base)`, перевод в/из разных оснований, работа с двоичными строками.
- **Строки:** индексация/срезы (`s[2:]`, `s[:-1]`), `len`, `count`, `replace`, `join`, `zfill`, `strip`.
- **Условия и циклы:** `if/elif/else`, `for`, итерация по диапазонам `range`.
- **Списки**, методы списков, списковые включения.
- **Функции:** определение функций, возврат результата, разбор примеров с параметрами.
- **Арифметика цифр числа:** выделение цифр через `//` и `%` либо перевод в строку.
- **Форматирование:** f-строки, аккуратный вывод и отладочные принты.

Приёмы, необходимые для решения вводных заданий

Перебор диапазонов, кратность, составные условия

```
for m in range(50, 101):
    for n in range(1000, 10001):
        if n % m == 0 and n % 143 == 0:
            print(m, n)
```

Python

Перевод в двоичный вид и удаление префикса `0b`

```
nums = [45, 110, 2323, 3456]
binaries = [bin(x)[2:] for x in nums]
print(binaries)  # ['101101', '1101110', ...]
```
Python

Подсчёт единиц в двоичной записи

```
ones = [(x, bin(x).count('1')) for x in nums]
print(ones)
```
Python

Манипуляции двоичными строками

- Удалить последний символ и приписать справа ’11’: `s = s[:-1] + '11'`
- Заменить последний символ на второй слева: `s = s[:-1] + s[1]`

Инверсия битов (8-бит) и перевод обратно в десятичное

```
n = 13
b = bin(n)[2:].zfill(8)     # '00001101'
inv = b.translate(str.maketrans('01', '10'))  # '11110010'
res = int(inv, 2)           # 242
print(res)
```
Python

Работа с десятичными цифрами числа

```
x = 583
a = x // 100          # сотни
b = (x // 10) % 10    # десятки
c = x % 10            # единицы
print(a + b, b + c)
```
Python

Разбор «простого» задания (чётность, два дописанных бита)

Условие (кратко): по натуральному `N` берём двоичную запись, справа дважды дописываем бит чётности (остаток от деления суммы битов на 2): сначала для исходной записи, затем — для уже расширенной. Полученное число — это `R`. Найти **наименьшее** `N`, для которого `R > 77`.

Алгоритм

1. Пусть `b = bin(N)[2:]`.
2. `p1 = sum(int(c) for c in b) % 2`, `b1 = b + str(p1)`.
3. `p2 = sum(int(c) for c in b1) % 2`, `b2 = b1 + str(p2)`.
4. `R = int(b2, 2)`. Ищем минимальное `N`, при котором `R > 77`.

Реализация на Python

```
def build_R(N: int) -> int:
    b = bin(N)[2:]
    p1 = sum(int(c) for c in b) % 2
    b1 = b + str(p1)
    p2 = sum(int(c) for c in b1) % 2
    b2 = b1 + str(p2)
    return int(b2, 2)

answer = None
for N in range(1, 10**6):
    if build_R(N) > 77:
        answer = N
        break

print(answer)  # 19
```
Python

Проверка найденного минимума вручную

- `N = 19`, `b = 10011`, сумма битов = 3 ⇒ `p1 = 1`, `b1 = 100111`.
- Сумма битов в `b1` = 4 ⇒ `p2 = 0`, `b2 = 1001110`.
- `R = 0b1001110 = 78 > 77` — условие выполнено.

Ответ: `19`.

Почему это действительно минимум? Для меньших `N` (например, 18: `b=10010`) двукратное дописывание чётности даёт `R ≤ 77`. Перебор по возрастанию `N` останавливается впервые на `N=19`, что гарантирует минимальность.

Разбор «усложнённого» задания (четырёхзначное число → конкатенация сумм)

Правило преобразования: дано четырёхзначное число `abcd` (без ведущего нуля).
Считаем три суммы: `s1=a+b`, `s2=b+c`, `s3=c+d`.
Удаляем *наименьшую* из сумм (если минимумов несколько — удаляем только один из них). Две оставшиеся суммы записываем подряд в **порядке неубывания** (как строку без разделителей).

Реализация преобразования

```
def transform(x: int) -> str:
    # x = abcd, 1000 ≤ x ≤ 9999
    a = x // 1000
    b = (x // 100) % 10
    c = (x // 10) % 10
    d = x % 10
    sums = [a + b, b + c, c + d]
    m = min(sums)
    # удалить лишь ОДНО вхождение минимальной суммы
    sums.remove(m)
    sums.sort()
    return f"{sums[0]}{sums[1]}"

# Примеры
print(transform(3527))  # s: 8,7,9 → удаляем 7 → '89'
print(transform(7701))  # s: 14,7,1 → удаляем 1 → '714'
```

Python

Типичные вопросы и подходы к решению

- **Прямое вычисление результата** для конкретного входа — просто вызвать `transform(x)`.
- **Поиск числа по условию на результат** (например, найти минимальное `x`, для которого результат равен/больше/меньше заданного):

```
goal = "1314"
ans = min((x for x in range(1000, 10000) if transform(x) == goal),
default=None)
      print(ans)
```

Python

- **Перебор с фильтром** по множеству входов (учёт запрета на ведущий ноль автоматически выполняется диапазоном `1000..9999`).

Важные нюансы:
1) При равенстве нескольких минимальных сумм удаляем только ОДНУ из них.
2) Конкатенация — это строковая запись, поэтому для сумм > 9 результат может иметь 3–4 цифры (например, «714»).
3) Требуется порядок *неубывания* двух оставшихся сумм — сначала меньшая (или равная), потом большая.

Вводные задачи

1. Даны два диапазона чисел: `M [50:100]` и `N [1000:10000]`. Выведите все пары чисел, где N кратно M и N кратно 143.
2. Преобразуйте десятичные числа `45, 110, 2323, 3456` в двоичную форму и выведите результат.
3. Посчитайте сколько единиц в каждом из этих двоичных чисел
4. Дана строка `s`. Удалите ее последний символ и допишите справа ‘11’
5. Дана строка `s`. Замените ее последний символ на второй слева символ.

6. Дано восьмибитное двоичное представление числа N. Замените числа на противоположные (вместо 0 – 1, вместо 1 — 0). Выведите результат в десятичном виде. Например.

- Дано число `N = 13`. Алгоритм работает следующим образом.
- Восьмибитная двоичная запись числа `N: 00001101`.
- Все цифры заменяются на противоположные, новая запись `11110010`.
- Десятичное значение полученного числа `242`

7. Дано трехзначное число. Сложите две первых цифры и две последних. Выведите два этих числа в порядке возрастания.
8. Дано трехзначное число N. Из цифр, образующих десятичную запись N, постройте наибольшее и наименьшее возможные двузначные числа