№1 №3 №5 №7 №9 №11 №13 №14 №16 №18 №20 №22 №24 №26
№2 №4 №6 №8 №10 №12 №15 №17 №19 №21 №23 №25 №27

Шпаргалка

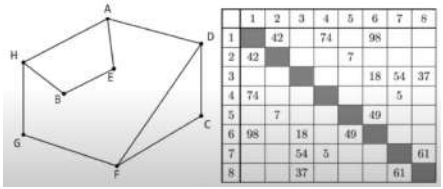
Информатика. Школково.

Задание №1

Алгоритм:

1. Ищем **«особую»** точку (из которой выходит отличное от всех остальных количество дорог)
   - ищем пункт, который **не соединён** с какой-то «особой» точкой – тогда он тоже «особый»
   - ищем пункт с уникальными соседями (из которых выходит определенное число дорог, отличное от остальных пунктов)
2. Смотрим, в какие города ведут дороги из этой точки
3. Таким образом, точка за точкой, восстанавливаем всю таблицу и уже потом смотрим расстояние между нужными городами
4. Расстояние между нужными городами — их пересечение в таблице

Типовая задача:



	1	2	3	4	5	6	7	8
1		42		74		98		
2	42				7			
3						18	54	37
4	74						5	
5		7				49		
6	98		18		49			
7			54	5				61
8			37				61	

1. Соотносим вершины одинаковых степеней:
   a. Вторая степень: B, C, E, G (2, 4, 5, 8)
   b. Третья степень: A, D, F, H (1, 3, 6, 7)
2. Находим особые точки: две вершины второй степени не связаны между собой — G, C (4, 8). Они обе связаны с двумя вершинами третьей степени, но в одном случае они связаны, а в другом — нет. Получается, что G — 4, а C — 8.
3. F связана с обеими вершинами G, C, поэтому F — 7
4. Аналогично восстанавливаем, что D — 3, A — 6, H — 1, E — 5 и B — 2
5. Далее ищем либо расстояние из одного пункта в другой, либо что требуется по условию

✅ **Подпишите число соседей (обозначьте большой цифрой) и число соседей каждого соседа (обозначьте маленькой цифрой)**

- Потом станет легче анализировать табличку (по числу соседей и числу соседей у соседей), особенно если числа соседей у всех разные

Важные моменты:

- Если нужно найти расстояние между несколькими городами — рассмотрите все возможные пути между двумя городами, тогда точно ничего не упустите.

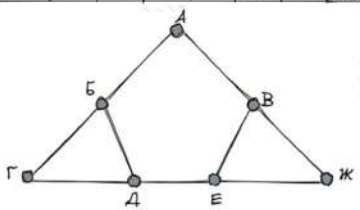
Если соответствие графа и таблицы неоднозначное, то скорее всего граф зеркальный, и это абсолютно нормально, тогда ищем ось симметрии и рассматриваем вариант с неоднозначностью городов (какая-то заметка в условии должна быть, чтобы помочь установить однозначное соответствие, например, что одна из дорог длиннее другой)

Пример (досрок):

- Требуется найти длину дорогу ДЕ.

При соотнесении номера пункта и буквы может получится два разных решения, при этом ответ получается один и тот же (граф зеркальный, его можно отразить)

Номер пункта	1	2	3	4	5	6	7
1		11	12	14			
2	11		13				
3	12	13				17	
4	14				18		
5				18		19	20
6			17		19		15
7					20	15	



Задание №2

ОЧЕНЬ ВАЖНО:

- Следите за скобками:
  - Импликация <= (и эквивалентность ==) в Python имеет приоритет выше, чем логические операции (not, and или or), однако в математической логике она выполняется после них!
    - Максимально важно всегда окружать переменные вокруг импликации в скобки, это самая частая ошибка в этом задании
  - not часто может обращаться не ко всему выражению — этого можно легко избежать, если not брать в главные скобки, вот так: (not())
- Операции алгебры логики в Python и их приоритет (математические [операции сравнения] всегда приоритетнее логических)

$$a \wedge b \rightarrow c$$

Правильно: (a and b) <= c
Неправильно: a and b <= c

Приоритет в Python	Операция	Запись в Python	Обозначения	Запись в Python через логику
1	Импликация	<=	→	(not a) or b
2	Эквивалентность	==	↔ ~ ≡	(not (a and b)) or (a and b)
3	Отрицание (не)	not	¬ā	not a
4	Конъюнкция (и)	and	∧ • &	a and b
5	Дизъюнкция (или)	or	∨ + ||	a or b

⚠ ИЗМЕНИТЬ ПРИОРИТЕТНОСТЬ ↔ ПОСТАВИТЬ СКОБКИ

1. Получаем набор x, y, z, w для которых функция ложна или истинна в зависимости от условия задачи
2. Аккуратно сопоставляем с таблицей в условии

Сопоставляем так:

- Ищем «уникальную» строчку или столбик (например, где только 0 или 1)
- Определяем, какой переменной она/он соответствует
- Ориентируясь на то, что уже определили, снова ищем «уникальность» в какой-то строчке, так доопределяем оставшиеся переменные

СПОСОБ №1

```
print("x y z w")
for x in range(2):
    for y in range(2):
        for z in range(2):
            for w in range(2):
                if (логическая функция == 0 или 1):
                    print(x, y, z, w)
```

СПОСОБ №2

```
from itertools import product
print("x y z w")
for x, y, z, w in product((0, 1), repeat=4):
    # repeat - количество переменных
    if логическая функция == 0 или 1:
        print(x, y, z, w)
```

Задание №3 начало

Общий алгоритм:

1. Есть несколько листов, на каждом устанавливаем фильтр (если дали определенный район – устанавливаем в фильтре его в соответствующем листе, если дали определенный товар – устанавливаем в фильтре его и т.д.)
2. Находим подходящие id с каждого листа (id подходящих магазинов, id подходящих продуктов, id подходящих групп, id подходящих альбомов)
3. Переносим всё это в общую таблицу, где собраны все данные и где находится искомый параметр (например, кол-во упаковок этого товара в том или ином магазине) – устанавливаем фильтр и выбираем нужные id в соответствующих столбиках
4. Готово, осталось ответить на вопрос задачи (сложить/вычесть найденные значения, например, если просили общее кол-во упаковок, найти максимум и тд)

ШКОЛКОВО

Задание №3 продолжение

✅ **ВАЖНО-1:** Обратите внимание, в каких единицах измерения вы получили ответ, а в каких просили (НА ЭТОМ СОСТАВИТЕЛИ ОЧЕНЬ ЧАСТО ЛОВЯТ), например:

- в таблице граммы, а ответ просят в кг
- в таблице рубли, а ответ просят в тысячах рублей
- в таблице упаковки товара по несколько штук, а ответе просят количество товара поштучно
- в таблице байты, а ответ просят в мегабайтах
- в таблице секунды, а ответ просят в миллисекундах

✅ **ВАЖНО-2:** Если просят найти «на сколько изменилось количество упаковок товара»:

- в столбце поступление/продажа выбираем сначала поступление
- смотрим сумму по кол-ву поступивших товаров, затем сумму по кол-ву проданных товаров
- вычитаем из количества поступивших товаров проданные товары, получаем ответ

Задание №4

ШКОЛКОВО

✅ **Двоичное дерево** — дерево, где от каждого узла левый потомок — 0, а правый потомок — 1.

Алгоритм:

1. **Условие Фано** — никакое кодовое слово не является началом другого кодового слова
2. Чтобы соблюсти условие Фано, рисуем двоичное дерево (**главное правило:** если подвесили букву, забываем об этой ветке, туда больше ничего вешать нельзя, ветка закрылась)
3. Подвешиваем уже известные буквы к их кодовым словам
4. Определяем, куда оптимальнее подвесить оставшиеся буквы (иногда придётся рассмотреть несколько вариантов)
   a. Как правило, если буква в слове повторяется несколько раз, то ей нужно назначить кодовое слово **минимальной** длины.

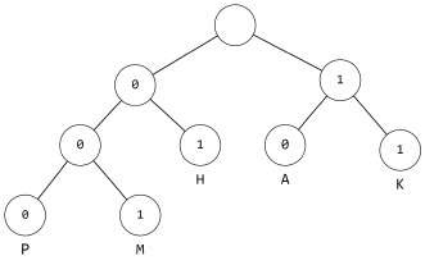
Важно обратить внимание на:

- Нужно ли использовать весь алфавит
  - Если да, то необходимо оставить одну ветку под оставшиеся буквы алфавита.
    - Если все ветки уже заняты, то это можно сделать разбив одну из уже занятых веток на две, тогда длина кодового слова этой буквы увеличивается на 1 символ.
- Есть ли буквы, которые не используются в слове, но на которые нужно оставить коды
- Если условие Фано обратное, тогда коды считываем из дерева не сверху вниз, а снизу вверх, не забывая, что граф перевёрнутый

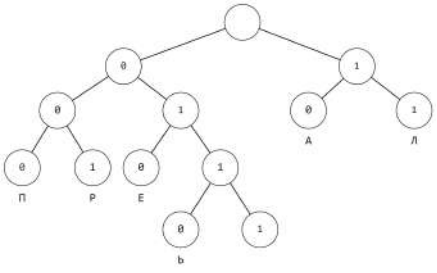
ШКОЛКОВО

Примеры задач

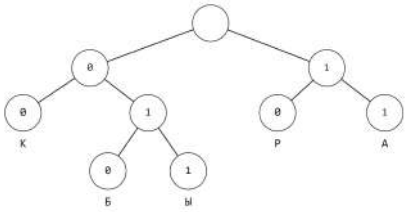
Для кодирования букв МНРАК использовали неравномерный двоичный код, удовлетворяющий условию Фано. Известно, что у букв М, Н, Р кодовые слова соответственно равны 001, 01, 000 а у остальных букв они имеют минимальную длину. Также известно, что у буквы А численное значение кодового слова меньше, чем у буквы К. Расшифруйте последовательность 11100000011001. В ответ запишите последовательность больших букв.



По каналу связи передаются сообщения, содержащие только все заглавные буквы русского алфавита. Для передачи используется двоичный код, допускающий однозначное декодирование. Укажите минимальную возможную длину закодированной последовательности ПАРАЛЛЕЛЬ.



Марафонец кодирует буквы К, Р, А, Б, Ы неравномерным двоичным кодом, который удовлетворяет обратному условию Фано. Известно, что букве К соответствует код 00, букве Р - 01, а букве А - 11. Укажите кодовое слово для буквы Б, если известно, что у него минимальное численное значение.



Задание №5 начало

✅ **Необходимые функции:**

- bin(x) — перевести число x из десятичной в двоичную систему
- oct(x) — перевести число x из десятичной в восьмеричную систему
- hex(x) — перевести число x из десятичной в шестнадцатеричную систему
- int("xxx", n) — перевод числа xxx из n-ой системы счисления в десятичную

Очень важно не путать, потому что:

- Из любой СС (от 2 до 36) можно перевести в десятичную с помощью функции
- Из десятичной можно перевести только в двоичную (bin()), восьмеричную (oct()) и шестнадцатеричную (hex())

✅ **ВАЖНО:**

- Переведённая в двоичную систему строка выглядит как 0b11101, например.
  - Нужно не забывать **отрезать первые два** символа строки, когда мы переводим десятичное число в двоичную СС, это делается так:
    - x = bin(n)[2:]

- Если какая-то часть алгоритма должна выполняться **по несколько раз**, тогда включаем этот кусок программы в цикл и пишем количество раз, сколько она должна выполняться (например, 3), в range(3) в for i in range(3):
  - Включаем только нужную часть программы внутрь цикла for
- Количество единиц в двоичном числе равно сумме его цифр, поэтому сумму цифр двоичного числа можно найти с помощью str.count('1')
- Посчитать **количество разрядов** в числе можно с помощью len(str(number))
- Если просят, чтобы число было четное/нечетное, то проверяем поданное на вход число по остатку на 2 (n % 2)
- Не путаем четность суммы цифр и самого числа
- Можно легко перебрать **не все возможные числа**, рекомендуется ставить range(x), где x достаточно большой (например, 1000000)
- При добавлении цифр к числу, используется **строковое сложение**, например string = string + '10', а не string = string + 10
- Иногда нужно **перевернуть число**, для этого можно использовать срез с шагом -1 (двигаемся из конца строки в начало) string = string[::-1]
- Иногда нужно **удалить какую-то из цифр** числа, это можно легко сделать при помощи срезов, например:
  - **Удалить первую цифру числа** можно следующим образом: string = string[1:]
  - **Удалить последнюю цифру** можно следующим образом: string = string[:-1]
  - **Удалить какую-то цифру** в числе сложнее, для этого нужно знать индекс этой цифры

ШКОЛКОВО

Задание №5 продолжение

- Индекс **первого включения** цифры в число (например, индекс первого нуля) можно найти с помощью i = string.index('0')
- Индекс **последнего включения** цифры в число можно найти с помощью этой же функции, предварительно перевернув число с помощью string[::-1], то есть i = string[::-1].index('0')
- В итоге, **убрать цифру по индексу** можно с помощью string = string[:i] + string[i + 1:]
- Если хочется **заменить цифру по индексу на какие-нибудь другие цифры** (например, 10), это можно сделать с помощью string = string[:i] + '10' + string[i + 1:]

Программа, если просят МИНИМАЛЬНОЕ (ПЕРВОЕ) ЧИСЛО, РЕЗУЛЬТАТ которого удовлетворяет условию

- Первое выведенное на экран число будет ответом
- Можно не писать break в конце, тогда программа будет выводить все подходящие варианты. Самым первым числом в выводе будет минимальное, а самым последним - максимальное
- Часто люди пугаются, когда им выводится слишком много чисел, поэтому break оптимален

```
for n in range(1000000):
    x = bin(n)[2:]

    # здесь пишется алгоритм из задания

    a = int(x, 2)
    if условие из задания:  # например, a > 127:
        print(n)
        break
```

ШКОЛКОВО

Программа, если просят найти МАКСИМАЛЬНОЕ (ПОСЛЕДНЕЕ) ЧИСЛО, РЕЗУЛЬТАТ которого удовлетворяет условию

- Последнее выведенное на экран число будет ответом

```
for n in range(1000000):
    x = bin(n)[2:]

    # здесь пишется алгоритм из задания

    a = int(x, 2)
    if условие из задания:  # например, a < 1024:
        print(n)
```

- Иногда в задаче слишком много чисел удовлетворяют условию, тогда делаем перебор не от наименьшего к наибольшему (снизу вверх), а от наибольшего к наименьшему (сверху вниз), и используем break, тогда код такой:
- В этом случае первое (и единственное) число, выведенное на экран, будет ответом

```
for n in range(10000, 0, -1):
    x = bin(n)[2:]

    # здесь пишется алгоритм из задания

    a = int(x, 2)
    if условие из задания:  # например, a < 1024:
        print(n)
        break
```

Программа, если просят найти минимальный (min()) РЕЗУЛЬТАТ:

- Если увеличить диапазон в range() несколько раз, и выводимое число не изменится, тогда, вероятнее всего, вы нашли нужное число

```
m = 1000000
for i in range(1, 1000000):
    x = bin(i)[2:]

    # здесь пишется алгоритм из задания

    a = int(x, 2)
    m = min(m, a)
print(m)
```

ШКОЛКОВО

Программа, если просят найти максимальный (max()) РЕЗУЛЬТАТ:

- Если увеличить диапазон в range() несколько раз, и выводимое число не изменится, тогда, вероятнее всего, вы нашли нужное число

```
m = -1
for i in range(1, 1000000):
    x = bin(i)[2:]

    # здесь пишется алгоритм из задания

    a = int(x, 2)
    m = max(m, a)
print(m)
```

✅ **ПОДВОХ:** Нужно именно искать минимальный максимальный из результатов, а не самих чисел, потому что не всегда результаты идут на монотонное увеличение/уменьшение

ШКОЛКОВО

Задание №6

✅ ВАЖНО:

- Смотрим, реально ли точки находятся на линии (иногда это не так)
- Смотрим, нужно ли учитывать точки на линиях или на пересечениях линий
- Если точек слишком много, их можно посчитать либо математически, либо с помощью перебора всех целочисленных координат с помощью Python
  - Может произойти такое, что не все координаты положительные, тогда не забывайте перебрать также их
- Существует прототип (Цапля), когда просят нарисовать полуокружность или сектор круга — тогда проще всего нарисовать фигуру на листочке и посчитать ручками (эффективно математически) все точки

Прога может не сработать, если:

- Вы забыли опустить хвост у черепашки
- Вы используете не того исполнителя
- Вы написали направо или налево вместо вправо или влево
- Вы перебираете не все координаты
- Вы неправильно записали неравенства, которые определяют фигуру

Пример шаблонного задания в кумире через исполнителя “Черепаха”

```
использовать Черепаха
алг
нач
  опустить хвост
    # повторяем цикл сколько-то раз
    нц N раз
      # пишем нужные команды
      вправо(градусы)
      вперед(клетки)
      влево(градусы)
      назад(клетки)
    кц
кон
```

Пример программы для пересчёта всех целочисленных координат, удовлетворяющих условию

```
count = 0
for x in range(1, 100):
    for y in range(1, 100):
        # пишем те неравенства, которые ограничивают
        # нашу фигуру в пересечении
        if неравенство1 and неравенство2 and неравенство3:
            count += 1
print(count)
```

ШКОЛКОВО

Задание №7

✅ **Надо знать:**
8 бит = 1 байт
1024 байт = 2^{10} байт = 2^{13} бит = 1 Кбайт
2^{10} Кбайт = 2^{20} байт = 2^{23} бит = 1 Мбайт
2^{10} Мбайт = 2^{20} Кбайт = 2^{30} байт = 2^{33} бит = 1 Гбайт

1. Проще всего делать все вычисления с помощью калькулятора, но если хочется проверить руками, тогда часто проще всё разложить на простые множители, а затем упрощать
2. Если просят найти “во сколько раз изображение/звуковая дорожка увеличилась/уменьшилась”, то больший объём делим на меньший, а затем сокращаем множители

ШКОЛКОВО

Информационный объём изображения

Для хранения растрового изображения нужно выделить в памяти $I = w \cdot h \cdot i$ бит, где w — ширина, h высота и i — глубина кодирования

$$I = w \cdot h \cdot i$$

Количество цветов: $N = 2^i$, где i — глубина кодирования

$$N = 2^i$$

Глубина кодирования означает, сколько битов нужно, чтобы закодировать определённое количество цветов, цвета кодируются с помощью двоичного кода. Например, рассмотрим случай, когда там нужно закодировать 7 цветов:

- 1-й цвет 000
- 2-й цвет 001
- 3-й цвет 010
- 4-й цвет 011
- 5-й цвет 100
- 6-й цвет 101
- 7-й цвет 110

Как можем наблюдать, для 7 цветов из 3 битов можно составить достаточное количество кодов, чтобы закодировать все цвета. Это значение коррелирует со степенями двойки (на каждый разряд можно поставить по 2 варианта, то есть перемножаем двойку столько раз, сколько есть доступных разрядов, и получаем количество кодов ($2 \cdot 2 \cdot 2 = 2^3 = 8$), где количество двоек (разрядов) и будет глубиной кодирования)
Если количество цветов **не является степенью двойки**, тогда глубину кодирования ищем от ближайшей степени двойки сверху (например, при N = 100):

- $2^6 = 64 < 100 < 128 = 2^7$, тогда глубина кодирования для 100 цветов будет 7
- Если вы знаете логарифмы, тогда глубина кодирования находится как **$[\log_2 N]$** обозначает округление вверх:

ШКОЛКОВО

✅ **ВАЖНО:**

1. Когда сказали «**изображение сжали на x%**», значит за 100% берем именно то, относительно чего изменилось (за 100% берем исходный объем, а не объем после сжатия).
2. «При кодировании каждого пикселя используется j бит для определения степени прозрачности» → добавляется j бит к весу одного символа (вес одного символа будет равен i + j)
3. «для каждых двух бит цвета дописывается дополнительный **бит контроля чётности**» == объем строится так: i бит на цвет + b бит четности (если нужно посчитать, сколько весит изображение без битов четности, из него нужно вычесть его **b/(i+b)-ую** часть)
4. Если размер картинки или её линейное разрешение увеличилось в 2 раза, тогда теперь картинка использует в 2 раза больше пикселей по ширине и по высоте, то есть информационный объём изображения увеличился в $2 \cdot 2 = 4$ раза (квадратичная зависимость, $x \sim n^2$)
5. Иногда требуется найти минимальное количество цветов, которое могло бы использоваться в изображении, тогда мы ищем первый цвет, который использует бит в максимальном разряде, например, если нужно найти минимальное количество цветов, которое могло бы использоваться при глубине кодирования 10 бит, то:
   - 1-й цвет: 0000000000
   - 2-й цвет: 0000000001
   - ...
   - 512-й цвет: 0111111111
   - 513-й цвет: 1000000000

Выходит, что 10 битами можно закодировать минимум 513 ($2^9 + 1$) цветов (это будет первый цвет, для кодирования которого используется десятый бит), получаем следующую формулу:

$$N = 2^{i-1} + 1$$

6. Если мы хотим найти максимальное количество цветов, то всё просто, берём

$$N = 2^i$$

Информационный объём аудиозаписи

1. Объём аудиозаписи $I = f \cdot i \cdot k \cdot t$, где f — частота дискретизации (в Гц), i — глубина кодирования в битах, k — количество каналов, t — время звучания (в секундах)

$$I = f \cdot i \cdot k \cdot t$$

Обычно в этой части задания достаточно просто перемножить все числа из условия, предварительно их переведя в нужную систему измерений (секунды, биты)

✅ ВАЖНО:

1. Очень легко проигнорировать слова стерео и моно
   - стерео означает, что каналов два (2)
   - моно означает, что каналов один (1)
   - Также встречается квадро, что означает, что каналов было четыре (4)
2. Все переменные выше имеют прямую зависимость (x~n), что означает, что если увеличить одно число в 2 раза, тогда и объём **I** вырастет в 2 раза
   a. Иногда встречается не время записи, а темп (BPM) — тогда зависимость обратная (x~1/n), то есть если темп увеличился в 2 раза, то время уменьшится в 2 раза

Информационный объём видеофайла

Видеофайл складывается из серии картинок и звука, соответственно остаётся перемножить размер одного изображения на частоту кадров и на длительность видео (в секундах).
После этого нужно прибавить к серии картинок размер аудио — получаем размер видео.

Передача по каналу связи

Если в задаче присутствует канал данных, то у него есть определённая скорость передачи данных. Как правило, это даётся, чтобы найти время, за которые передавались данные разного объёма при одной и той же скорости, а затем сравнить их объём и из этого найти недостающую величину.

$$I = V \cdot t$$

ШКОЛКОВО

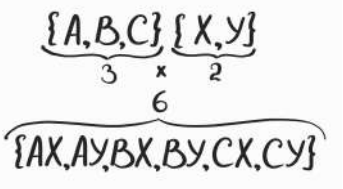
Такие задания очень просто решаются, есть представлять, как зависят величины при одинаковой скорости, то есть:

$$I = I / t = const$$

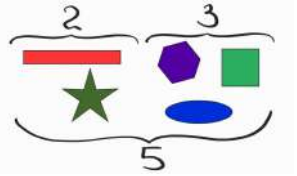
Во сколько раз медленнее передаётся файл, во сколько же раз он больше (обратная зависимость, x~1/n)

Задание №8 начало

- **Правило умножения:** если элемент a можно выбрать n способами, а элемент b — m способами (независимо от a), тогда все пары a и b можно выбрать $n \cdot m$ способами



- **Правило сложения:** если элемент a можно выбрать n способами, а элемент b — m способами (независимо от a), тогда все пары a и b можно выбрать $n + m$ способами

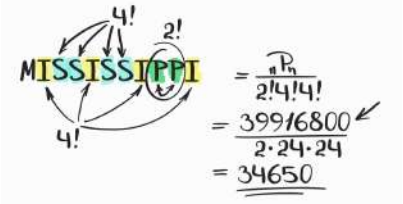


- ***n!*** — количество способов переставить n объектов без повторений
  - Например, при 3-х различных элементах мы можем поставить на первое место один из 3-х элементов, затем один из 2-х, затем один (1) оставшийся, всего $3 \cdot 2 \cdot 1 = 3! = 6$ вариантов:
    - 123
    - 132
    - 213
    - 231
    - 312
    - 321

$$\frac{n!}{k_1! k_2! k_3!}$$

— количество способов переставить n объектов с повторениями, где $k_{1..3}$ — количества повторяющихся элементов

Задание №8 продолжение



- Мы делим на возможное количество перестановок одинаковых элементов между собой, потому что такие случаи уже будут учитываться
- **Сочетания** позволяют найти количество способов выбрать k объектов из n БЕЗ учёта порядка

$$C_n^k = \frac{n!}{(n-k)! \cdot k!}$$

- **Размещения** позволяют найти количество способов выбрать k объектов из n С учётом порядка

$$A_n^k = \frac{n!}{(n-k)!}$$

Прога для поиска КОЛИЧЕСТВА СЛОВ/ЧИСЕЛ, подходящих под определенные условия

Первый вариант:

```
h = 0
letters = "ABCDEFGHI3" # здесь пишем все буквы/цифры из задания БЕЗ ПОВТОРОВ

# количество циклов for равно длине СЛОВА (не количеству возможных букв)
for a in letters:
    for b in letters:
        for c in letters:
            for d in letters:
                for e in letters:
                    s = a + b + c + d + e # здесь складываем ВСЕ буквы в строку
                    if все условия из задания:
                        h += 1
print(h)
```

Второй вариант:
Через модуль ITERTOOLS

- product("ABCDEF", repeat = ...) – размещения с повторениями (эквивалентное вложенному циклу for, в repeat устанавливаем длину слова/разрядность числа, здесь буквы будут повторяться)
- s = " ".join(("A", "B")) превращает кортеж ("A", "B") в строку через разделитель " ". В данном примере после исполнения функции получается s = "AB"

```
from itertools import product

ans = 0
# в repeat устанавливаем длину слова/разрядность числа
for i in product("ABCDEF", repeat = 5):
    s = "".join(i)
    if все условия из задания:
        ans += 1
print(ans)
```

- permutations("ABCDEF") – перестановки (тут буквы не будут повторятся, если в слове не повторяются)
- permutations("ABCDEF", 3) – все слова из трёх букв, которые можно составить из всех букв слова, второй аргумент отвечает за длину возвращаемых кортежей

```
from itertools import permutations

ans = 0
for i in permutations("ABCDEF"):
    s = "".join(i)
    if все условия из задания:
        ans += 1
print(ans)
```

Прога для поиска НОМЕРА СТРОЧКИ для какой-то определенной комбинацией букв

- Сначала определите, строчку с какой именно комбинацией надо искать

```
letters = "ваши буквы в АЛФАВИТНОМ порядке"

# нумерация в задании с единицы
n = 1

# количество циклов for равно длине СЛОВА
for a in letters:
    for b in letters:
        for c in letters:
            for d in letters:
                for e in letters:
                    s = a + b + c + d + e
                    if s == "ваше слово":
                        print(n)
                    n += 1
```

Прога для поиска СЛОВА под каким-то определённым номером

- Сначала определите, слово на каком номере вам нужно найти

```
letters = "ваши буквы в АЛФАВИТНОМ порядке"

# нумерация в задании с единицы
n = 1

# количество циклов for равно длине СЛОВА
for a in letters:
    for b in letters:
        for c in letters:
            for d in letters:
                for e in letters:
                    s = a + b + c + d + e
                    if n == номер нужного слова:
                        print(s)
                    n += 1
```

✅ ВАЖНО:

1. Если можете, решите задание двумя способами — комбинаторным и программным
2. **Число не может начинаться с 0!!!** (поэтому 0 при выборе/генерации первой цифры убираем)
3. Если две чётные и две нечётные цифры/ согласная и гласная не должны стоять рядом, то четная и нечетная/гласная и согласная чередуются.
   a. Это позволяет легко рассмотреть все варианты руками, потому что все строки будут формата XOXOXOXO, где, например, X — нечётная цифра или согласная буква, а O — чётная цифра или гласная буква
4. Внимательно читайте, могут ли повторятся буквы/цифры

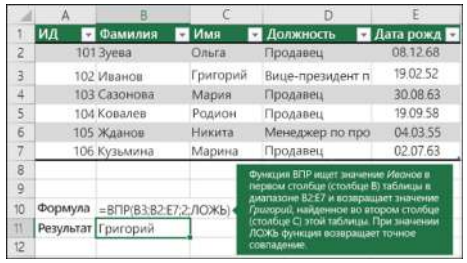
Задание №9

- Диапазон можно выбрать с помощью X1:X2
- В формулах может использоваться либо,, либо ; (чаще всего на ЕГЭ используются версии Excel-я, где используются точки с запятой)
- Перечислить отдельные клетки можно с помощью X1; X2; X3
- Выбрать весь столбец можно с помощью X:X, где X — буква столбца
- Для быстрого перемещения в конец столбца/ строки или между заполненными диапазонами можно использовать Ctrl+стрелка, а для выделения диапазона Ctrl+Shift+стрелка
- Если у вас вылезает какая-то непонятная ошибка, проверьте, написали ли вы диапазоны ЛАТИНСКИМИ буквами, а названия функций РУССКИМИ буквами
  - Особенно стоит обратить внимание на те функции, которые начинаются с буквы С, потому что С на английской и на русской раскладке находятся на одной клавише
- При выделении каких-то диапазонов может автоматически считаться сумма с краю/снизу, что позволяет лишний раз не мучаться с формулами

Полезные формулы

Полезные формулы	Синтаксис
Найти сумму числа	СУММ(X1:X2)
Посчитать количество непустых ячеек	СЧЁТ(X1:X2)
Подсчёт среднего арифметического значения	СРЕДНЕЕ(X1:X2)
Найти максимальное значение	МАКС(X1:X2)
Найти минимальное значение	МИН(X1:X2)
Проверка условия	ЕСЛИ(условие; значение_если_истина; значение_если_ложь)
Логическое И для условий	И(условие1; условие2)
Логическое ИЛИ для условий	ИЛИ(условие1; условие2)
Посчитать количество непустых ячеек в диапазоне, которые удовлетворяют заданному условию	СЧЁТЕСЛИ(X1:X2; условие)
Сумма значений в диапазоне, которые удовлетворяют заданному условию	СУММЕСЛИ(X1:X2; условие)
Остаток от деления одного числа на другое	ОСТАТ(X1; X2)
Модуль числа	ABS(X1)
Наиболее часто встречаемое число	МОДА(X1:X2)
Среднее число, если все числа в диапазоне расположить по порядку	МЕДИАНА(X1:X2)
Квадратный корень из числа	КОРЕНЬ(X1)
N-ое наибольшее/ наименьшее число из диапазона	НАИБОЛЬШИЙ(X1:X2; N) НАИМЕНЬШИЙ(X1:X2; N)
ВПР	ВПР(искомое значение; место для его поиска; номер столбца в диапазоне с возвращаемым значением; возврат приблизительного или точного совпадения — указывается как 1/ ИСТИНА или 0/ЛОЖЬ)

ШКОЛКОВО



	A	B	C	D	E
1	ИД	Фамилия	Имя	Должность	Дата рожд
2	101	Зуева	Ольга	Продавец	08.12.68
3	102	Иванов	Григорий	Вице-президент п	19.02.52
4	103	Сазонова	Мария	Продавец	30.08.63
5	104	Ковалев	Родион	Продавец	19.09.58
6	105	Жданов	Никита	Менеджер по про	04.03.55
7	106	Кузьмина	Марина	Продавец	02.07.63
8					
9					
10	Формула	=ВПР(B3;B2:E7;2;ЛОЖЬ)			
11	Результат	Григорий			
12					

Функция ВПР ищет значение *Иванов* в первом столбце (столбце B) таблицы в диапазоне B2:E7 и возвращает значение *Григорий*, найденное во втором столбце (столбце C) этой таблицы. При значении ЛОЖЬ функция возвращает точное совпадение.

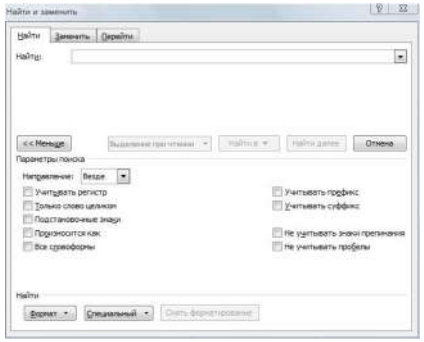
Задание №10

Сноска — текст, находящийся в самом низу страницы, отдельно от основного
Регистр у букв — размер букв: заглавная или строчная.
- ✅ Учитывать регистр

Формы слова — варианты одного и того слова.
- ✅ Все словоформы
  - Для существительных и прилагательных — разный падеж или число
  - Для глаголов — разное число, лицо, наклонение, время
- Если просят найти только само слово целиком, то нажимаем ✅ Только слово целиком

✅ ВАЖНО:
- Не забывайте прокликать все найденные слова – это единственная самопроверка в этой задаче
- Обратите внимание, чтобы слова не находились в сносках, если это требуется по заданию, чаще всего выполняется поиск во всём документе
- Если нужно найти только лишь строку, входящую в слово (исключая само слово, например ход только в составе ходить), то проще всего:
  a. Найти количество всех слов, содержащих нужную строку
  b. Найти количество включений этого слова целиком
  c. Вычесть из первого значения второе

Данное окно можно открыть с помощью Ctrl + F в Word и LibreOffice.



ШКОЛКОВО

Задание №11

Алгоритм:
1. **Считаем мощность алфавита.** Если посимвольное кодирование, то считаем кол-во символов, которые можно использовать в пароле; если кодирование не посимвольное, то считаем кол-во различных значений, которые необходимо закодировать
2. **Находим, сколько бит на один символ выделить** (находим, какая степень двойки не меньше мощности алфавита), через логарифмы это можно сделать так, где *[x]* — округление вверх:
$$i = \lceil \log_2 N \rceil$$
3. **Находим, сколько бит на весь пароль нужно выделить** (кол-во символов в пароле умножаем на вес одного символа):
$$I = i \cdot k$$
4. **Переводим вес пароля в байты** (округляем вверх до целого числа байт!!!) и далее по заданию.
5. Если вместе с паролем хранится дополнительная информация, тогда добавляем её вес (I_{extra}) к весу одного пароля, и далее считаем общий вес всех паролей, умножая вес одного пароля на количество
6. Далее может потребоваться найти какое-то из нескольких значений, это могут быть:
   a. сколько дополнительных байтов информации используется для каждого из паролей (I_{extra})
   b. сколько дополнительных байтов информации выделено всего на хранение дополнительной информации ($I_{extra} \cdot n$)
   c. сколько всего байтов занимают все пароли на диске (K)

Это всё находится из выражения:
$$K = (I + I_{extra}) \cdot n$$

✅ ВАЖНО:
1. Внимательно следите, кодирование пароля равномерное или неравномерное (можно ли одну часть пароля закодировать одним числом байт, а вторую – другим)
2. Не путайте длину пароля и количество символов в алфавите
3. Не забывайте учесть десятичные цифры, если таковые даны по заданию
4. Может произойти такое, что все пароли кодируются целым минимально возможным числом бит, тогда переводить пароль в байты на этапах алгоритма не нужно
5. Может произойти такое, что какая-то часть пароля кодируется отдельно как число (например, хранится двоичное число), тогда действуем как при не посимвольном кодировании из первого пункта алгоритма

Задание №12

Исполнитель “Редактор”
- Задания на исполнителя “Редактор” спокойно решаются шаблонно
- Не забывайте писать 1 в replace("old", "new", 1), иначе программа заменит ВСЕ числа в строке, а не единственное
- replace() возвращает строку, а не изменяет исходную
  - Правильно: string = string.replace()
  - Неправильно: string.replace()
- Если дана строка, в которой все цифры расположены в произвольном порядке, – можно располагать их в любом порядке (подряд удобнее всего)
  - Обычно такое происходит, потому что в программе реализуется какая-то сортировка чисел
- Не путаем or (или) и and (и)
- Для генерации строк из n-ого количества нескольких цифр используем строковое сложение и умножение, например: "1" * 20 + "2" * 50
  - Например, если изначальная строка состоит из нескольких многократно повторяющихся разных цифр, где количество повторений одной из цифр неизвестно, то её можно генерировать в каждой из итераций цикла с помощью, например "1" * 20 + "2" * i + "3" * 30

Прога, если начальная строка известна

```
string = "строка из условия"

# здесь пишем именно что дано в алгоритме из условия
while "комбинация1" in string or/and "комбинация2" in string:
    if "комбинация1" in string:
        string = string.replace("комбинация1", "новая комбинация1", 1)
    else:
        string = string.replace("комбинация2", "новая комбинация2", 1)

print(string)
```

Прога, если начальная строка НЕИЗВЕСТНА, нужно найти количество цифр, из которых состояла строка

```
# перебираем разное изначальное количество единиц
for i in range(...):
    # генерируем изначальную строку
    string = "1" * i

    # здесь пишем именно что дано в алгоритме из условия
    if "комбинация1" in string:
        string = string.replace("комбинация1", "новая комбинация1", 1)
    else:
        string = string.replace("комбинация2", "новая комбинация2", 1)

    if string = "строка в результате":
        print(i) # количество символов в изначальной строке
```

Исполнитель “Чертёжник”

✅ ВАЖНО:
- Задания на исполнитель “Чертёжник” рассчитаны на решение систем уравнений, так как отслеживаются две координаты (x, y), и расписано, как они изменяются
- Задача — трансформировать алгоритм в такой формат, который можно было бы решить математически
- Если не хватает одной или двух переменных — вам повезло, такое задание решается просто обычной системой уравнений

Рассмотрим следующий алгоритм в качестве примера, учитывая, что после завершения алгоритма чертёжник вернулся в изначальную точку, и мы хотим найти максимальное число N.

```
НАЧАЛО
  сместиться на (-30, -110)
  ПОВТОРИ N РАЗ
    сместиться на (a, b)
    сместиться на (76, -93)
  КОНЕЦ ПОВТОРИ
  сместиться на (0, 5)
КОНЕЦ
```

ШКОЛКОВО

Задание №12 продолжение

Раз чертёжник вернулся в изначальную точку, конечные координаты будут равны (0, 0).

$$\begin{cases} x = -30 + N \cdot (a + 76) = 0 \\ y = -110 + N \cdot (b - 93) + 5 = 0 \end{cases}$$

Перенесём все числа в правые части:

$$\begin{cases} N \cdot (a + 76) = 30 \\ N \cdot (b - 93) = 105 \end{cases}$$

Разложим числа 30 и 105 на множители

$$\begin{cases} N \cdot (a + 76) = 2 \cdot 3 \cdot 5 \\ N \cdot (b - 93) = 3 \cdot 5 \cdot 7 \end{cases}$$

Раз для выбранных N должны подобраться целые значения, то N должно быть делителем чисел 30 и 105. Максимальное такое число — 15.

Стратегия решения данного задания — составить систему уравнений, отслеживающую координаты (x, y), а затем решить эту систему в целых числах для N на основе НОД() получившихся чисел.

```
for N in range(16):
    for a in range(-300, 300):
        for b in range(-300, 300):
            if -30 + N * (a + 76) == 0 and -110 + N * (b - 93) + 5 == 0:
                # Берём максимальное N
                print(N)
```

Задание №13

✅ Кол-во дорог в город А – это сумма количества дорог, ведущих в города, дороги из которых ведут в А.

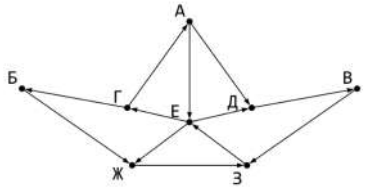
Алгоритм:

1. Внимательно читаем вопрос
   a. Если просят количество путей – ищем количество путей
   b. Если просят, чтобы пути проходили/не проходили через какую-то вершину – считаем только такие пути
   c. Если просят длину дороги – ищем длину, а не количество путей
   d. Обращаем внимание, если находится путь максимальный или минимальной длины
   e. Если попросили, чтобы пути проходили через определенный город – зачеркиваем все пути, что не проходят через него
   f. Если попросили, чтобы пути НЕ проходили через определенный город – зачеркиваем все пути, что ведут в этот город и что выходят из него.
2. Подписываем каждую вершину как сумму кол-ва дорог, которые ведут в ее (у первой вершины степень всегда 1, потому что из самого города попасть в себя же можно одним лишь способом — остаться на месте)
3. Проделываем так до пункта назначения.

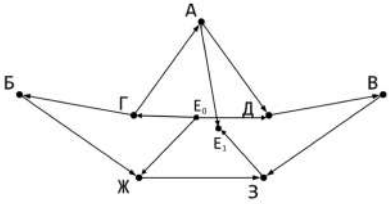
Поиск кол-ва путей в циклическом графе (начало и конец в одном и том же городе)

Алгоритм

1. Разбиваем город, который является и началом, и концом на два
   a. Один город будет иметь только исходящие пути — это начальный город
   b. Другой город будет иметь только входящие пути — это конечный город



Граф до разбиения

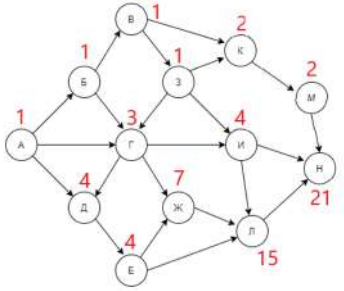


Граф после разбиения

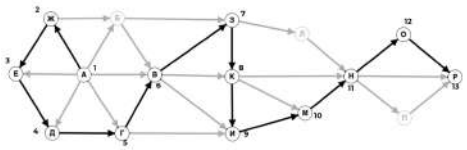
2. Выполняем этапы обычного алгоритма
3. Если какая-то дорога ведёт в тупик (в город, который мы уже посетили), тогда убираем те дороги, которые нельзя посетить, они бесполезны

Примеры графических решений

Подсчёт количества путей в графе



Поиск максимального количества городов



Исключаем заведомо более короткие дороги, или дороги, имеющие одинаковую длину до одного и того же пункта (нам не важно, какой путь, нужно найти максимальный путь)

Задание №14

Универсальная функция перевода из десятичной СС в любую другую (основания от 2 до 36)

- number - число из условия
- base - основание СС

```
alphabet = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# number - число/выражение из условия
# base - основание СС
def convert_base(number, base):
    s = ""
    while number > 0:
        s = s + alphabet[number % base]
        number = number // base
    return s
```

Функция перевода из любой системы счисления в десятичную СС

```
result = int(str(number), base)
```

Функция, чтобы посчитать количество включений цифры в число

```
c = result.count("5") # пишем нужную цифру
```

Функция перевода из любой системы счисления в десятичную СС

```
for x in "0123456789ABCDEF":
    if (int("цифры числа до нужной цифры" + x + "остальные цифры", 16) + \
            int("другое число", 16)) % кратность из условия == 0:
        print(x)
```

Задание №15

x	y	x∨0	x∧0	x∨1	x∧1	x̄	x∧y	x∨y	x≡y	x→y	x̄∨y
0	0	0	0	1	0	1	0	0	1	1	1
0	1	0	0	1	0	1	0	1	0	1	1
1	0	1	0	1	1	0	0	1	0	0	0
1	1	1	0	1	1	0	1	1	1	1	1

ОЧЕНЬ ВАЖНО:

- Следите за скобками:
  - Импликация <= (и эквивалентность ==) в Python имеет приоритет выше, чем логические операции (not, and или or), однако в математической логике она выполняется после них!
    - Максимально важно всегда окружать переменные вокруг импликации в скобки, это самая частая ошибка в этом задании
  - not часто может обращаться не ко всему выражению — этого можно легко избежать, если not брать в главные скобки, вот так: (not())
- Операции алгебры логики в Python и их приоритет (математические [операции сравнения] всегда приоритетнее логических)

$$a \wedge b \rightarrow c$$

Правильно: (a and b) <= c
Неправильно: a and b <= c

Приоритет в Python	Операция	Запись в Python	Обозначения	Запись в Python через логику
1	Импликация	<=	→	(not a) or b
2	Эквивалентность	==	↔ ~ ≡	(not (a and b)) or (a and b)
3	Отрицание (не)	not	¬ā	not a
4	Конъюнкция (и)	and	∧ • &	a and b
5	Дизъюнкция (или)	or	∨ + |	a or b

⚠ ИЗМЕНИТЬ ПРИОРИТЕТНОСТЬ ↔ ПОСТАВИТЬ СКОБКИ

Полезные формулы

	Дизъюнкция	Конъюнкция
Двойного отрицания	$\bar{\bar{x}} = x$	$\bar{\bar{x}} = x$
Переместительный	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
Сочетательный	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
Распределительный	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Идемпотентности	$x \vee x = x$	$x \wedge x = x$
Исключения третьего	$x \vee \bar{x} = 1$	$x \wedge \bar{x} = 0$
Поглощения	$x \vee (x \wedge y) = x$	$x \wedge (x \vee y) = x$
де Моргана	$\overline{(x \vee y)} = \bar{x} \wedge \bar{y}$	$\overline{(x \wedge y)} = \bar{x} \vee \bar{y}$

ШКОЛКОВО

Задание №15 продолжение

Методика решения руками

Алгоритм

1. Упрощаем выражение по максимуму с помощью законов алгебры логики, таким образом получая систему для друзей
2. Берём отрицание от всего выражения, получая систему для врагов
3. Анализируем систему и ищем такое максимальное/минимальное значение A, чтобы противодействовать врагам (чтобы их система никогда не выполнялась)

Рассмотрим пример выражения ДЕЛ(x, A) -→ (НЕДЕЛ(x, 21) ИЛИ ДЕЛ(x, 35)), нужно найти минимальное A, когда друзья побеждают (функция всегда принимает истинные значения)

- Упрощаем до НЕДЕЛ(x, A) ИЛИ НЕДЕЛ(x, 21) ИЛИ ДЕЛ(x, 35) используя законы алгебры логики, получаем систему для друзей

$$\begin{cases} x \not\vdots A \\ x \not\vdots 21 \\ x \vdots 35 \end{cases}$$

- Берём отрицание от всего выражения, получаем систему для врагов
  - Заменяем совокупности на системы и наоборот
  - Заменяем отрицания на неотрицания и наоборот

$$\begin{cases} x \vdots A \\ x \vdots 21 \\ x \not\vdots 35 \end{cases}$$

- Друзья хотят, чтобы система была ложная, для этого нужно чтобы все числа, которые делятся на 21 и на A, делились на 35

$$\begin{cases} x \vdots A \\ x \vdots 3 \cdot 7 \\ x \not\vdots 5 \cdot 7 \end{cases}$$

- Если мы возьмём A = 5, тогда все числа, которые делятся на 5 и 21 не должны делиться на 35, что невозможно, получили противоречие

✅ ВАЖНО:

1. Следите за тем, какими являются переменные A, x, y:
   - натуральными (от 1 и больше)
   - целыми и неотрицательными (0, 1 и больше)
   - целыми и положительными (= натуральные)
   - целые (тогда рассматриваются все числа от -N до N)

От этого зависит, с чего начинать диапазон в range()

- Очень частая ошибка — написать == вместо = или наоборот, например при назначении флага нужно писать именно flag = False, а не flag == False
- Если нужно найти значения, когда функция всегда ложная, пишем not в return в def f():

ПРАВИЛО +/- 1:

- Длина отрезка: конец - начало
- Кол-во чисел на отрезке: конец - начало + 1

Неравенства

```
def f(x, y, A):
    return функция из условия
for A in range(1, 200):

    # предполагаем, что с этим A всё хорошо
    flag = True

    for x in range(1, 200):
        for y in range(1, 200):
            # в условии нужно истинно поэтому если
            # при этом A - ложно, выходим с флагом False
            if not f(x, y, A):
                flag = False
                break
        if not flag:
            break
    if flag:
        # максимальный ответ будет присвоен на последней операции
        ans = A

print(ans)
```

Поразрядная конъюкнция

- Поразрядная конъюнкция в Python записывается как &

```
def f(x, a):
    return функция из условия


for a in range(0, 300):
    p = True
    for x in range(0, 300):

        # в условии нужно истинно поэтому если
        # при этом A - ложно, выходим с флагом False
        if not f(x, a):
            p = False
            break
    if p:
        print(a)
```

Числовой отрезок

$[a_1, b_1]$ — числовой отрезок

Способ 1

```
# функция для проверки, входит ли число в отрезок
# использовать при написании функции f()
def inside(x, A):
    return A[0] <= x <= A[1]


def f(x, a):
    P = [a1, b1]
    Q = [a2, b2]
    K = [a3, b3]
    return функция из условия
borders = [0, 0]
minim = 100000

# воспользуемся более точным приближением при подборе
k = 5

for a in range(0, 80 * k):
    for b in range(a, 80 * k):
        A = [a / k, b / k]

        # предполагаем, что при этом A всё хорошо
        good = True
        for x in range(0, 100 * k):
            # в условии нужно истинно поэтому если
            # при этом A - ложно, выходим с флагом False
            if not f(x / k, A):
                good = False
                break

        # всё хорошо, отрезок такой походит
        if good:
            if A[1] - A[0] <= minim:
                # сохраняем длину
                minim = A[1] - A[0]
                # сохраним сам отрезок
                borders = A.copy()

print(minim)
print(borders)
```

Способ 2

```
# задаём отрезки
p = range(a1, b1 + 1)
q = range(a2, b2 + 1)

# для максимального отрезка: создаём множество со значениями от 1 до 1000
a = set(range(1, 1000))
# для минимального отрезка: создаём пустое множество
a = set()

for x in range(1, 1000):
    if логическая функция:  # или not логическая функция
        # для максимального отрезка:
        # выкидываем такие иксы из отрезка
        a.remove(x)

        # для минимального отрезка:
        # добавляем такие иксы в отрезок
        a.add(x)
# количество элементов в отрезке
print(len(a))

# максимальная длина отрезка
print(sorted(a)[-1] - sorted(a)[0])
```

Способ 3 (в действительных числах)

```
# задаем отрезки
p = [i / 10 for i in range(a1, b1 + 1)]
q = [i / 10 for i in range(a2, b2 + 1)]

# для максимального отрезка: создаём множество
# со значениями от 1 до 1000
a = set([i / 10 for i in range(10000)])

# для минимального отрезка: создаём пустое множество
a = set()

for x in [1 / 10 for i in range(10000)]:
    if логическая функция:  # или not логическая функция
        # для максимального отрезка:
        # выкидываем такие иксы из отрезка
        a.remove(x)

        # для минимального отрезка:
        # добавляем такие иксы в отрезок
        a.add(x)
# количество элементов в отрезке
print(len(a))

# максимальная длина отрезка
print(sorted(a)[-1] - sorted(a)[0])
```

Задание №16 начало

✅ **Рекурсивная функция** — функция, которая в своей записи содержит себя же.

Шаблонная прога (одна функция)

```
def F(n):
    if условие 1:
        return результат 1
    if условие 2:
        return результат 2
    if условие 3:
        return результат 3

print(F(число из условия))
```

Пример проги (две функции)

Функции могут вызывать как себя, так и другую функцию

```
def F(n):
    if n < 4:
        return n
    return F(n - 3) + G(n - 3) + 2

def G(n):
    if n < 3:
        return n
    return G(n - 1) + F(n - 1)

print(F(9))
```

ШКОЛКОВО

Задание №16 продолжение

✅ **ВАЖНО:**

- Классическая стратегия решения этого задания — скопировать алгоритм в программный код
- Если кол-во вызовов функции сильно большое, то можно включить меморизацию памяти с помощью @cache из библиотеки functools (аналогично заданиям 19-21)
- Размер стека можно увеличить с помощью import sys и sys.setrecursionlimit(большое число)
- ВНИМАНИЕ: Альтернативно, можно понять алгоритм и ручками посчитать, какое значение будет получаться от определенных аргументов вручную

Задание №17

✅ **ВАЖНО:**

- В начале не пишут N – кол-во чисел в файле (не нужно считывать первое число, оно не показывает кол-во чисел в файле)
- Обращайте внимание, есть ли в файле отрицательные числа (тогда, возможно, придётся использовать модуль (abs()), например, если необходимо посмотреть на последнюю цифру произведения пары)
- Смотрите на то, числа в пределах какого диапазона находятся в файле – это поможет определить, как инициализировать макс/мин.
- Обращайте внимание, среди всех ли пар нужно искать мин/макс или только из подходящих
- Если в цикле есть обработка i + a элемента, где a — числовое значение, то в range() обязательно нужно вычесть a, чтобы программа не получила ошибку выхода за границу цикла
- Может получиться такое, что рассматриваются не соседние пары, а ВСЕ пары, тогда используйте два вложенных цикла:

```
# сначала перебираем все варианты взять одно число в паре
for i in range(len(a)):
    # затем перебираем все варианты взять другое число в паре,
    # не включая уже рассмотренные числа
    for j in range(i + 1, len(a)):
        if a[i] + a[j] ...: # пишем условие
```

Считывание информации из файла

Вариант 1

```
f = open("17.txt")
a = []
for n in f:
    a.append(int(n))
f.close()
```

Вариант 2

```
with open("17.txt") as f:
    a = list(map(int, f.readlines()))
```

Вариант 3

```
f = open("17.txt")
a = [int(x) for x in f]
f.close()
```

Прога

Будьте аккуратны с тем, что вам требуется найти — в коде ниже приводятся разные случаи

```
# инициализируем, если ищем количество пар/чисел
count = 0

# инициализируем, если ищем сумму пар/чисел
result = 0

# инициализируем, если нужен максимум
m = -1

# инициализируем, если нужен минимум
m = 100000000

# считываем числа из файла любым способом в a
with open("17.txt") as f:
    a = list(map(int, f.readlines())

            for i in range(len(a) - 1):
    if a[i] + a[i + 1]...:  # пишем условие по заданию
        count += 1
    result += a[i]  # или a[i + 1], зависит от
                    # условия и того, что делаем

    # если ищем максимум
    m = max(m, a[i] + a[i + 1])

    # если нужен минимум
    m = min(m, a[i] + a[i + 1])

    print(count, result, m)
```

ШКОЛКОВО

Задание №18

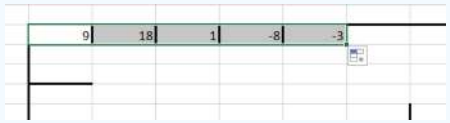
✅ Стандартная формула:

- = значение в клетке из исходной таблицы + МАКС(сосед 1 в новой таблице; сосед 2 в новой таблице)
- Например, = B2 + МАКС(B12; A13) – ходим из левого верхнего в правый нижний, ходы: вниз или вправо

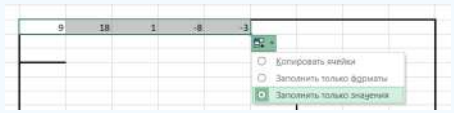
✅ Лайфхак:

Как растянуть формулу и не потерять стенки:

1. Тянем ячейку с нужной формулой за правый нижний край на область копирования:



2. В появившемся квадратике “Параметры автозаполнения” выбираем “Заполнить только значения - ГОТОВО



✅ **ВАЖНО:**

- Очень внимательно читайте, откуда и куда ходит робот (из левой нижней в правую верхнюю или из левой верхней в правую нижнюю и тд)
- Если в файле есть стенки, то просто обходим их, заполняем всю таблицу стандартной формулой, а в конце вернемся к стенкам и поменяем для них формулу
  - Если из-за стенки, например, нельзя походить влево, тогда оставляем только ход сверху)
- Если у вас вылезает какая-то непонятная ошибка, проверьте, написали ли вы диапазоны ЛАТИНСКИМИ буквами, а названия функций РУССКИМИ буквами
  - Особенно стоит обратить внимание на те функции, которые начинаются с буквы С, потому что С на английской и на русской раскладке находятся на одной клавише

ШКОЛКОВО

- Если у вас не сходится ответ, рекомендуется попытаться проделать задание чисто с нуля — мало ли что-то сломалось, пока вы редачили таблицу
- Полезно делать пробел перед первой строкой и первым столбцом — тогда формула будет ссылаться на пустые клетки, а не за границу, и не нужно будет дополнительно редактировать формулы около краёв
- Если в задании поиска максимальной суммы нужно обязательно посетить некоторые клетки, добавляем к каждой из нужных ячеек очень большое число, а затем из финального результата вычитаем очень большое число X количество клеток, которых обязательно посещать
  - Для минимальной суммы отнимаем какое-то очень большое число, затем его прибавляем
- Если в задании нельзя посещать какие-то клетки, тогда устанавливаем в них очень маленькое (отрицательное) число для максимальной суммы, тогда эти клетки не будут посещаться
  - Для минимальной суммы устанавливаем очень большое число
- Если задание нестандартное (например, используются шахматные фигуры), тогда алгоритм решения задания такой же, просто нужно учитывать, как ходит исполнитель.
  - В шаблонной формуле берём максимум/минимум из всех клеток в новой таблице, из которых можно попасть в текущую клетку:
    - = значение в клетке из исходной таблицы + МАКС(клетка 1 в новой таблице; клетка 2 в новой таблице; ... клетка N в новой таблице)

ШКОЛКОВО

Задание №19-21

✅ **ВАЖНО:**

- END — конец игры
- WIN1 — первый игрок выигрывает первым ходом
- LOSE1 — второй игрок выигрывает первым ходом (первый проигрывает)
- WIN2 — первый игрок выигрывает вторым ходом
- LOSE2 — второй игрок выигрывает вторым ходом (первый проигрывает)
- any() используется если достаточно, что один из ходов выигрывающий/проигрывающий
- all() используется если нужно, что все ходы выигрывающие/проигрывающие
- Если что-то не получается:
- Проверьте, что в конце скрипта вы ищете нужное значение
- Проверьте, что вы не написали лишних пробелов в строках выше и что они идентичных друг другу
- Расположите ходы в порядке от самого увеличивающего до самого уменьшающего
- Проверьте, не забыли ли вы использовать @lru_cache(None) или @cache (для Python 3.9+)
- Если вылезает ошибка RecursionError, тогда нужно установить лимит рекурсии на очень большое число с помощью import sys и sys.setrecursionlimit(очень большое число)
- Проверьте, что в задаче ищется именно сумма, а не произведение, тогда sum(h) нужно заменить на h[0] * h[1]

Скрипт для одной кучи

```
from functools import lru_cache


def moves(h):
        # ходы +1, +2, *3, записываем те, которые нужны
    return h * 3, h + 2, h + 1
```

ШКОЛКОВО

```
@lru_cache(None)
def f(h):
        # игра заканчивается, если в куче 59+ камней
    if (h >= 58):
        return "END"
    if any(f(x) == "END" for x in moves(h)):
        return "WIN1"
    if all(f(x) == "WIN1" for x in moves(h)):
        return "LOSE1"
    if any(f(x) == "LOSE1" for x in moves(h)):
        return "WIN2"
    if all(f(x) == "WIN1" or f(x) == "WIN2" for x in moves(h)):
        return "LOSE2"


for s in range(1, 58):
    if f(s) == "LOSE2": # здесь пишем в зависимости от условия
        print(s)
```

Скрипт для двух куч

```
from functools import lru_cache


def moves(h):
    a, b = h
# ходы *5 и +4, записываем те, которые нужны
    return (a * 5, b), (a, b * 5), (a + 4, b), (a, b + 4)
@lru_cache(None)
def f(h):
# игра заканчивается, если в куче 59+ камней
    if sum(h) >= 58:
        return "END"
    elif any(h(x) == "END" for x in moves(h)):
        return "WIN1"
    elif all(h(x) == "WIN1" for x in moves(h)):
        return "LOSE1"
    elif any(h(x) == "LOSE1" for x in moves(h)):
        return "WIN2"
    elif all(h(x) == "WIN1" or h(x) == "WIN2" for x in moves(h)):
        return "LOSE2"

for i in range(1, 58):
    if f((10, s)) == "LOSE2": # здесь пишем в зависимости от условия
        print(s)
```

Скрипт на уменьшение камней в кучах

```
from functools import lru_cache


def moves(heap):
    a, b = heap
    m = []
# ходы -1, //2 (целочисленное деление на 2)
    if a >= 1:
        m += [(a - 1, b), (a // 2, b)]
    if b >= 1:
        m += [(a, b - 1), (a, b // 2)]
    return m
@lru_cache(None)
def game(heap):
    if sum(heap) <= 12:
        return "END"
    elif any(game(x) == "END" for x in moves(heap)):
        return "WIN1"
    elif all(game(x) == "WIN1" for x in moves(heap)):
        return "LOSE1"
    elif any(game(x) == "LOSE1" for x in moves(heap)):
        return "WIN2"
    elif all(game(x) == "WIN1" or game(x) == "WIN2" for x in moves(heap)):
        return "LOSE2"


for s in range(100, 3, -1):
    if game((9, s)) == "LOSE1": # здесь пишем в зависимости от условия
        print(s)
```

ШКОЛКОВО

Условие проигрыша

```
from functools import lru_cache


def moves(heap):
    return heap + 2, heap * 3
@lru_cache(None)
def game(heap):
    if 64 <= heap <= 84:
        return "END"
    elif heap > 84:
        return "WIN1"
    elif any(game(x) == "END" for x in moves(heap)):
        return "WIN1"
    elif all(game(x) == "WIN1" for x in moves(heap)):
        return "LOSE1"
    elif any(game(x) == "LOSE1" for x in moves(heap)):
        return "WIN2"
    elif all(game(x) == "WIN1" or game(x) == "WIN2" for x in moves(heap)):
        return "LOSE2"
for s in range(63, 0, -1):
    if game(s) == "LOSE1":
        print(s)
        break
```

Ограничение по ходам (нельзя повторять предыдущий ход)

```
from functools import lru_cache


def moves(heap):
    h, k = heap
    m = []
    if k != 0:
        m += [(h + 1, 0)]
    if k != 1:
        m += [(h + 2, 1)]
    if k != 2:
        m += [(h * 3, 2)]
    return m
@lru_cache(None)
def game(heap):
    if heap[0] >= 50:
        return "END"
    elif any(game(x) == "END" for x in moves(heap)):
        return "WIN1"
    elif all(game(x) == "WIN1" for x in moves(heap)):
        return "LOSE1"
    elif any(game(x) == "LOSE1" for x in moves(heap)):
        return "WIN2"
    elif all(game(x) == "WIN1" or game(x) == "WIN2" for x in moves(heap)):
        return "LOSE2"


for s in range(1, 58):
    if game((s, -1)) == "LOSE1":
        print(s)
```

Задание №22

Алгоритм решения через Excel

1. Разбиваем столбец с ID поставщиков данных на несколько столбцов по разделителю ; (например, это будут столбцы C, D)
2. Создаём столбец, в котором будем отслеживать, когда процессы заканчивают работу (например, столбец E)
3. Копируем формулу =B2 на все ячейки столбца E, которые имеют 0 в графе ID поставщиков данных (это можно сделать с помощью фильтра)
4. Создаём столбцы для отслеживания того, когда закончится каждая из зависимостей (например, F соответствует C, а G — D)
5. Заполняем строки с зависимыми процессами (строки, где нет нулей в ID поставщиков данных) формулами
   c. Для столбца F: =СУММЕСЛИ(A:A; C2; E:E)
   d. Для столбца G: =СУММЕСЛИ(A:A; D2; E:E)
6. Находим время, когда завершатся все процессы, взяв =МАКС() из диапазона

Скрипт для перенумерации ID (если они страшные)

- Копируем всю таблицу из Excel и вставляем её в качестве input(), затем получаем новую таблицу с исправленными формулами

```
n = 100 # количество строк
a = [input() for i in range(n)]
for i in range(n):
    s = a[i].split()
    id_old = s[0]
    id_new = str(i + 1)
    for j in range(n):
        a[j] = a[j].replace(id_old, id_new)

for i in range(n):
    print(a[i])
```

Скрипт для генерации формул

- Копируем третий столбец с зависимостями и вставляем его в качестве input(), затем копируем получившийся результат обратно в Excel

```
n = 100 # количество строк
for i in range(1, n + 1):
    s = input().split(";")
    if s[0] == "0":
        formula = "=B" + str(i)
    else:
        formula = "=МАКС("
        for x in s:
            formula = formula + "D" + x + ";"
        formula = formula[:-1] + ") + B" + str(i)
    print(formula)
```

Последний шаг

- Ищем максимальное значение в столбце D с помощью формулы =МАКС(D1:D100)

Код для решения задачи полностью

```
from functools import lru_cache
n = int(input()) # количество строк
time = [0 for i in range(n + 1)]
depends = [[] for i in range(n + 1)]

@lru_cache(None)
def lazy_dp(k):
    if depends[k][0] == 0:
        return time[k]
    else:
        m = -1
        for i in depends[k]:
            m = max(m, lazy_dp(i))
    return m + time[k]

for i in range(1, n + 1):
    a = list(map(int, input().split()))
    time[i] = a[0]
    del a[0]
    depends[i] = a.copy()
ans = -1
for i in range(1, n + 1):
    ans = max(ans, lazy_dp(i))
print(ans)
```

ШКОЛКОВО

Задание №23

Разные вариации алгоритмов в динамике

- ЗДЕСЬ ПУТАЮТСЯ ВСЕ ВО ВСЁМ

направление цикла	цикл for	используется когда
прямое	Запускаем for от меньшего числа к большему	число увеличивается
обратное	Запускаем for от большего числа к меньшему	число уменьшается

порядок рассматривания элементов	рассматриваемый элемент i	рассматриваемые операции
прямой	начальный	прямые (оставляем те операции, которые находятся в условии)
обратный	конечный	обратные (берём обратные операции тем, которые в условии)

прямая операция	обратная операция
сложение	вычитание
вычитание	сложение

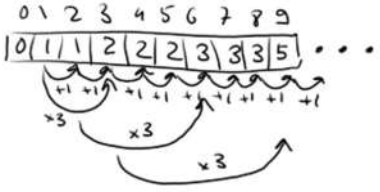
✅ **ВАЖНО:**

- Частая ошибка: a[start] = 1, а не ~~a[1] = start~~
- Если траектория должна содержать какую-то точку, то все пути, что не проходят через эту точку, нужно исключить (обнулить), когда до этой точки дошёл цикл
- Если траектория НЕ должна содержать какую-то точку, то эту точку нужно исключить (обнулить значение кол-ва путей в неё — это можно делать на каждой итерации)
- Создавайте достаточно большой массив на значения (лучше сильно больше)
- Траектория это последовательность чисел, через которые проходит программа (i), а не количества путей в последовательности (~~a[i]~~)
- Если что-то не получается, полезно распечатать массив a и посмотреть, что там находится
- Условия из задачи (например, что можно делить только числа, кратные на 2) относятся к начальным элементам
- Если вы делаете операцию умножения обратной, нужно учитывать, кратен ли текущий индекс числу, на которой вы пытаетесь делить

Реализации алгоритмов

✅ **Типовое задание**

- Исполнитель N преобразует число на экране. У исполнителя есть две команды:
- Умножить на 3
- Прибавить 1
- Сколько существуют программ, для которых при числе 1 результатом будет 28, при этом траектория вычислений содержит число 5 и не содержит число 17?



Динамика. Число увеличивается, рассматривается начальный элемент

```
a = [0] * 1000 # создаём массив на все ходы
a[1] = 1 # инициализуем количество путей из начала на один

# идём до "связующего узла" невключительно, через который
# должны пройти все
for i in range(1, 5):
    a[i + 1] += a[i]
    a[i * 3] += a[i]

# обнуляем все остатки путей, которые прошли не через 5
# (идём от 5 невключительно)
for i in range(5 + 1, 100):
    a[i] = 0

# продолжаем от "связующего узла" включительно до конца
# невключительно
for i in range(5, 28):
    # обнуляем элемент в начале итерации
    if i == 17:
        a[i] = 0
    a[i + 1] += a[i]
    a[i * 3] += a[i]

print(a[28])
```

Динамика. Число увеличивается, рассматривается конечный элемент

```
a = [0] * 1000 # создаём массив на все ходы
a[1] = 1 # инициализуем количество путей из начала на один

for i in range(1 + 1, 28 + 1):
    a[i] += a[i - 1]
    a[i] += a[i // 3] * (i % 3 == 0)
    if i == 5:
        for i in range(1, 5):
            a[i] = 0
    if i == 17:
        a[i] = 0

print(a[28])
```

ШКОЛКОВО

Рекурсия. Число увеличивается

- В рекурсии нет обратного подхода, где рассматривается конечный элемент, потому что функции на вход подаются начальные значения

```
def f(start, finish, flag, flag_number, exit_number):
    if start == finish and flag:
        return 1
    if start > finish or start == exit_number:
        return 0
    if start == flag_number:
        # поднимаем флаг, если дошли до требуемого значения
        flag = True
    x = f(start + 1, finish, flag, flag_number, exit_number)
    y = f(start * 3, finish, flag, flag_number, exit_number)
    return x + y

print(f(1, 28, False, 5, 17))
```

✅ **Типовое задание (число уменьшается)**

- Исполнитель N преобразует число на экране. У исполнителя есть три команды:
- Если число делится на 3, поделить его на 3
- Если число делится на 2, поделить его на 2
- Вычесть 1
- Сколько существуют программ, для которых при числе 40 результатом будет число 3?

Динамика. Число уменьшается, рассматривается начальный элемент

```
a = [0] * 1000 # создаём массив на все ходы
a[40] = 1 # инициализуем количество путей из начала на один

# идём от начала включительно до конца невключительно
for i in range(40, 3, -1):
    if i % 3 == 0:
        a[i // 3] += a[i]
    if i % 2 == 0:
        a[i // 2] += a[i]
    a[i - 1] += a[i]

print(a[3])
```

Динамика. Число уменьшается, рассматривается конечный элемент

```
a = [0] * 1000 # создаём массив на все ходы
a[40] = 1 # инициализуем количество путей из начала на один

# идём от начала невключительно до конца включительно
for i in range(39, 3 - 1, -1):
    a[i] = a[i * 3] + a[i * 2] + a[i + 1]

print(a[3])
```

ШКОЛКОВО

Условие, когда путь “содержит столько-то команд”

✅ **Типовое задание (число уменьшается)**

- Исполнитель N преобразует число на экране. У исполнителя есть две команды:
1. Умножить на 3
2. Прибавить 1
- Сколько существуют программ, для которых при исходном числе 1 результатом является число 28, при этом программа содержит 9 команд, а траектория обязательно не проходит через число 11 и проходит через 13?

Динамика

```
# создаём двумерный массив:
# в первой ячейке будем хранить программы
# во второй - команды
a = [[0] * 10 for i in range(100)]
a[1][0] = 1

for i in range(1, 13):
    for j in range(9):
        if i == 11:
            a[i][j] = 0
        a[i + 3][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
        a[i + i][j + 1] += a[i][j]

for i in range(14, 100):
    for j in range(9):
        a[i][j] = 0

for i in range(13, 28):
    for j in range(9):
        a[i + 3][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
        a[i + i][j + 1] += a[i][j]

print(a[28][9])
```

Рекурсия

```
def f(start, finish, flag, command):
    if start > finish:
        return 0
    if start == 11:
        return 0
    if start == 13:
        flag = True
    if command < 0:
        return 0
    if start == finish and flag and command == 0:
        return 1
    return f(start * 3, finish, flag, command - 1) + \
           f(start + 1, finish, flag, command - 1) + \
           f(start + start, finish, flag, command - 1)

print(f(1, 28, False, 9))
```

ШКОЛКОВО

Задание №24

✅ **Типовое задание**

- Обратите внимание, в файле одна строка или несколько строк (от этого зависит, как необходимо считывать)
  - Одна строка считывается с помощью s = f.readline()
  - Много строк проще всего считать, поместив их в массив с помощью s = readlines(), а затем пройтись циклом for через них
- Внимательно следите за тем, где обнуляется переменная (нужно ли обнулять перед каждой новой итерацией или нет)
- Внимательно подумайте, счетчик инициализировать 1 или 0
  - Как правило, 1 для пар символов, потому что тогда один символ уже учитывается
  - Если ищете макс длину цепочки, сравниваем парочку соседних символов (s[i] == s[i+1]), начальный счетчик берем за 1 (т.к. если пара 1 – то буквы в ней 2, если пар 2, то буквы в ней 3 и т.д. – БУКВ НА 1 БОЛЬШЕ ЧЕМ ПАР)
- «Если таких результатов несколько, выведите последний/первый встречающийся»
  - Последний встречающийся → знак не строгий при поиске макс/мин
  - Первый встречающийся → знак строгий при поиске макс/мин
- Не забывайте, что иногда можно идти нестандартными путями в задаче, например, воспользоваться .split() и считать разбитые подцепочки
- Помните, что есть использовать .split() для задачи, где нужно подсчитать максимальную длину, не содержащую что-то, рассмотрите боковые символы от разбитой цепочки
  - Например, в строке BBBAABBBBBBAA максимальная подстрока без пар AA — ABBBBBBA, а не BBBBBB

ШКОЛКОВО

```
f = open('24.txt') # откроем файл

# считать одну строку
s = f.readline()

# считать все строки и поместить их в массив
a = f.readlines()

n = len(s) # длина строки s
n = s.count("AR") # число подстрок "AR" внутри строки s
n = s.find("AR") # поиск подстроки "AR" слева,
                 # вычисляет индекс первого вхождения,
                  #иначе -1
n = s.rfind("AR") # поиск подстроки "AR" справа, вычисляет
                  # индекс последнего вхождения, иначе -1

# замена одного вхождения слева подстроки
# "AR" на подстроку "MO"
s = s.replace("AR", "MO", 1)
```

Прототип поиска максимальной длинной цепочки одинаковых символов

```
f = open("24.txt")
s = f.readline()
current_length = 1
max_length = 0
for i in range(len(s) - 1):
    if s[i] == s[i + 1]:
        current_length += 1
        max_length = max(max_length, current_length)
    else:
        current_length = 1

print(max_length)
```

Пример использования .split()

✅ Текстовый файл состоит не более чем из 10^6 заглавных букв латинского алфавита. Найдите количество последовательностей, каждая из которых содержит не менее 15 символов, начинается и заканчивается буквой A и не содержит букв A и R (кроме начальной и конечной).

```
f = open("24.txt").read()
ar = f.split("A")
count = 0
# подстроки длины не менее 15 содержат A,
# значит длина без A должно быть не менее 13
if f[0] == "A":
    if (len(ar[0]) >= 13) and (ar[0].count("R") == 0):
        count += 1

if f[len(f) - 1] == "A":
    if (len(ar[len(ar) - 1]) >= 13) and (ar[len(ar) - 1].count("R") == 0):
        count += 1

for i in range(1, len(ar) - 1):
    if (len(ar[i]) >= 13) and (ar[i].count("R") == 0):
        count += 1
```

Найти и вывести самый повторяющийся символ

✅ **Пример:**
Найдите самый часто повторяющийся символ между символами A и R (в данном порядке). Если таких символов несколько, следует взять тот, что раньше по алфавиту.

Через массив с алфавитом

```
n = open("24.txt").readline()
letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
array = [0] * 26

for i in range(1, len(n) - 1):
    if n[i - 1] == "A" and n[i + 1] == "R":
        array[letters.index(n[i])] += 1

for i in range(len(array)):
    if array[i] == max(array):
        print(letters[i])
        break
```

Через ord(), chr()

```
n = open("24.txt").readline()
array = [0] * 200
for i in range(1, len(n) - 1):
    if n[i - 1] == "A" and n[i + 1] == "R":
        array[ord(n[i])] += 1
print(array)
for i in range(len(array)):
    if array[i] == max(array):
        print(chr(i))
        break
```

Задание №25 начало

✅ **Важно:**

- **Простое число** — число, которое имеет всего 2 делителя — 1 и само число.
  - Число 1 не является простым, потому что у него всего 1 делитель.
- **Нетривиальные делители** — все делители числа, кроме 1 и самого числа.
- **Тривиальные делители числа** — 1 и само число.

- Нечетное число делителей означает, что число является квадратом (проверяем только числа, являющиеся квадратами)
- Иногда проще сначала сгенерировать подходящее под условие число, а затем проверить, что оно из нужного промежутка (а не наоборот)
  - В этом случае запускаем циклы по переменным, из которых генерируются числа, а не по самим числам
  - Затем проверяем, входит ли сгенерированное число в промежуток

Сложные случаи:

- Если нас просят найти числа, имеющие только лишь какое-то количество (n) нечётных делителей, тогда это число имеет формат p ** (n - 1), где p — простое число
  - Делители такого числа будут иметь формат 1, p ** 1, p ** 2, p ** 3 ...
  - Из этого же можно получить числа, имеющие такое же количество чётных делителей, просто умножаем каждое число на 2, то есть эти числа будут иметь формат 2 * p ** (n - 1)
- Если просят найти числа, имеющие какое-то количество (n) нечётных делителей (нам всё равно на другие делители), тогда это число имеет формат 2 ** k * p ** (n - 1), где p — простое число
  - Домножая сгенерированное число на 2 количество нечётных делителей не меняется

Оптимизированная функция нахождения количества делителей числа от O(x)

```
def count_div(n):
    # 1 и само число и так делители числа
    k = 2
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            k += 1
            if n // i != i:
                k += 1
    return k
```

Задание №25 продожение

Функция нахождения отсортированного массива нетривиальных делителей

```
def dividers(n):
    a = []
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            a.append(i)
            if n // i != i:
                a.append(n // i)
    a.sort()
    return a
```

Функция проверки числа на простоту

```
def is_prime(n):
    if n == 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

Функция нахождения минимального делителя (после 1)

```
def min_divider(n):
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return i
```

Генерация чисел из простых множителей и двоек (решение задачи на 3 нечётных делителя из диапазона [A, B])

```
for i in range(1, 100):
    for j in range(0, 100):
        if is_prime(i):
            if A <= i ** 2 * 2 ** j <= B:
                ...
```

ШКОЛКОВО

Маски

✅ Назовём **маской числа** последовательность цифр, в которой также могут встречаться следующие символы:

- Символ «?» означает ровно одну произвольную цифру;
- Символ «*» означает любую последовательность цифр произвольной длины
  - В том числе «*» может задавать и пустую последовательность. Например, маске 1234?5 соответствуют числа 123405 и 12300405.

✅ **Типовой пример**

Среди натуральных чисел, не превышающих 10^9, найдите все числа, соответствующие маске 1234*7?8, делящиеся на число 137 без остатка.
В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им результаты деления этих чисел на 137.

```
# пустая звёздочка
for r in range(10):
    s = int("12347" + str(r) + "8")
    if s % 137 == 0:
        print(s, s // 137)

# односимвольная звёздочка
for j in range(10):
    for a in range(10):
        s = int("1234" + str(j) + "7" + str(a) + "8")
        if s % 137 == 0:
            print(s, s // 137)

# двухсимвольная звёздочка
for i in range(10):
    for x in range(10):
        for y in range(10):
            s = int("1234" + str(i) + str(x) + "7" + str(y) + "8")
            if s % 137 == 0:
                print(s, s // 137)
```

ШКОЛКОВО

✅ **ВАЖНО:**

- Следите за порядком вывода чисел, чтобы выводилось от меньшего к большему!!!!! (в программе выше мы учли возрастание)
- Как правило, напрямую итерировать по всем числам и затем проверять их на соответствие маске будет слишком долго, поэтому ищите более лёгкие обходные пути

Задание №26 начало

✅ **ВАЖНО:**

Чтобы в Excel разделить текст по разным столбцам, жми клавиши:
Данные → Текст по столбцам → С разделителями → С разделителями → Пробел → Готово

Сортировка одного столбца

- Данные → Сортировка

Сортировка нескольких столбцов

- Данные → Сортировка → Добавить уровень

Удалить повторяющиеся данные

- Данные → Удалить дубликаты

✅ Все прототипы этих задач на сортировку, и большинство из них можно сделать руками, просто отсортировав значения в excel или python
Сортировка по возрастанию: a.sort()
Сортировка по убыванию: a.sort(reverse = True)

Подарки

```
f = open("26.txt")
n = int(f.readline())
a = sorted([int(s) for s in f], reverse=True)
ans = 1
x = a[0]
for i in range(1, len(a)):
        # разница между коробками как минимум 5
    if (x - a[i]) >= 5:
        ans += 1
        x = a[i]
print(ans, x)
```

ШКОЛКОВО

Места в кинотеатре/лесополоса

✅ Внимательно читайте, занятые или свободные места

- Обращайте внимание, нужно написать максимальный или минимальный номер ряда/места, от этого зависит, по какому из значений нужно проводить сортировку
- Если мест слишком много и их не удаётся просмотреть руками, можно использовать формулы, чтобы обозначить те последовательности занятых/свободных мест, которые удовлетворяют условию
- Номера мест, находящихся через одно место, различаются на два.
- Смотрите, какое место от вас требуется написать (первое, среднее, последнее...)

Архив файлов

✅ Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов. Известно, какой объём занимает файл каждого пользователя.
По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Обращайте внимание на величины, которые нужно найти и какие ограничения есть в самой задаче (минимальное/максимальное количество пользователей или минимальный/максимальный размер файла при максимальном/минимальном количество пользователей)

```
f = open("26.txt")
lines = f.readlines()

s, n = map(int, lines[0].split())
array = list(map(int, lines[1:]))
array = sorted(array)

users_count = 0
maximum_file = 0
current_summ = 0
i = 0
while i < n:
    if current_summ + array[i] > s:
        break
    current_summ += array[i]
    i += 1
users_count = i
current_summ -= array[i - 1]

while i < n:
    if current_summ + array[i] > s:
        break
    i += 1
current_summ += array[i - 1]
maximum_file = array[i - 1]

print(users_count, maximum_file)
```

Олимпиады

✅ По итогам проведения олимпиады по программированию каждый участник получил определённое количество баллов. По регламенту олимпиады победителя присуждают K лучших участников, а призёра присуждают M лучших участников, следующих за ними. По заданной информации о результатах каждого из участников определите по модулю разность суммы баллов победителей и призёров данной олимпиады.

Задание №26 продолжение

```
f = open("26.txt")
n, k, m = map(int, f.readline().split())

a = []
for i in range(n):
    a.append(int(f.readline()))
a.sort(reverse=True)

ans1 = 0
for i in range(k):
    ans1 += a[i]

ans2 = 0
for i in range(k, k + m):
    ans2 += a[i]
print(abs(ans1 - ans2))
```

UNIX-время

✅ Во многих компьютерных системах текущее время хранится в формате «UNIX-время» – количестве секунд от начала суток 1 января 1970 года. В одной компьютерной системе проводили исследование загруженности. Для этого в течение месяца с момента UNIX-времени 1632930743 фиксировали и заносили в базу данных моменты старта и финиша всех процессов, действовавших в этой системе.
Вам необходимо определить, какое наибольшее количество процессов выполнялось в системе одновременно на двух неделях, начавшихся в момент UNIX-времени 1632932743, и в течение какого суммарного времени (в секундах) выполнялось такое наибольшее количество процессов.

ШКОЛКОВО
ШКОЛКОВО

```
start = 1632932743  # начальное время
finish = start + 2 * 7 * 24 * 60 * 60
file = open("26.txt", "r")
lines = file.readlines()
n = int(lines[0])
# перебираем каждую стоку кроме первой (в которой N)
# и берем по одному числу из каждой
start_array = [int(line.split()[0]) for line in lines[1:]]
finish_array = [int(line.split()[1]) for line in lines[1:]]
for i in range(len(finish_array)):
    if finish_array[i] == 0:
        finish_array[i] = start + 2000000000000000000000000

start_array = sorted(start_array)
finish_array = sorted(finish_array)

i = 0
j = 0
maxim = 0
k = 0
while i < n and j < n:
    if start_array[i] < finish_array[j]:
        k += 1
        if k > maxim and start_array[i] >= start and start_array[i] < finish:
            maxim = k
        i += 1
    else:
        k -= 1
        j += 1

while j < n:
    k -= 1
    j += 1

i = 0
j = 0
k = 0
ans_count = 0
for sec in range(start, finish):
    while i < len(start_array) and start_array[i] <= sec:
        k += 1
        i += 1
    while j < len(finish_array) and finish_array[j] <= sec:
        k -= 1
        j += 1
    if k == maxim:
        ans_count += 1

print(maxim, ans_count)
```

ШКОЛКОВО
ШКОЛКОВО

Задание №27

- Если вы пытаетесь решать это задание, подразумевается, что вы проверили решения предыдущих заданий, потому что гораздо проще ошибиться в предыдущих заданиях, чем решить это
- Здесь приводятся несколько самых распространённых прототипов

Последовательности (префиксные суммы)

```
# найти максимальную сумму подряд идущих чисел, кратных 42

f = open("27.txt")
n = int(f.readline())
s = 0
ans = -1000000

# минимальные префиксные суммы элементов
# кратных всем числам до 42 включительно
mps = [1000000] * 42

# так как минимальная префиксная сумма, кратная 42, является нулём
mps[0] = 0

for i in range(n):
    # считаем текущую сумму
    s += int(f.readline())

    # динамически перезаписываем ответ
    ans = max(ans, s - mps[s % 42])

    # кладём минимальную префиксную сумму в нужную ячейку
    mps[s % 42] = min(mps[s % 42], s)

print(ans)
```

ШКОЛКОВО

Пары из последовательности

```
# найдите количество пар чисел, где произведение чисел будет кратно 8,
# но некратно 16, а сумма будет иметь остаток 4 при делении на 10

f = open("27.txt")
n = int(f.readline())

sv = [[0] * 10 for i in range(8 + 1)]
ans = 0

for i in range(n):
    x = int(f.readline())

    if x % 16 != 0:
# [8, 4, 2, 1] - делители 16, кроме 16
        for j in [8, 4, 2, 1]:
            if j * x % 8 == 0 and j * x % 16 != 0:
                for k in range(10):
                    if (k + x) % 10 == 4:
                        ans += sv[j][k]

        for j in [8, 4, 2, 1]:
            if x % j == 0:
                sv[j][x % 10] += 1
                break

print(ans)
```

ШКОЛКОВО
ШКОЛКОВО
ШКОЛКОВО

Движение по окружности (мусорки)

```
# найти индекс пункта на кольцевой дороге,
# где будет дешевле всего поставить пункт сбора мусора

f = open("27.txt")
n = int(f.readline())
a = [int(i) for i in f]
s = [0] * n

# задаём первую последовательность
current_sum = 0
for i in range(n // 2):
    current_sum += a[i]
s[0] = current_sum

# задаём следующие полуокружности
for i in range(1, n):
    s[i] = s[i - 1] - a[i - 1] + a[(i - 1 + n // 2) % n]

cost = 0
for i in range(n):
    # считаем стоимость доставки по кругу
    cost += (min(i, n - i) * a[i])
minim = cost
for i in range(1, n):
    # пересчитываем стоимость доставки,
    cost = cost - s[i] + s[(i + n // 2) % n]

    # передвигая точки старта
    if minim > cost:
        minim = cost
        ans = i

# добавляем +1 в ответ, так как индексация с нуля
print(ans + 1)
```

ШКОЛКОВО

Очереди

```
# найти максимальную сумму, кратную 9
# элементы которой находятся на расстоянии не меньше 7

f = open("27.txt")
n = int(f.readline())
nums = [0] * 9

ans = 0
line = []
for i in range(6):
    line += [int(f.readline())]

for i in range(6, n):
    x = int(f.readline())

    ans = max(ans, x + nums[(9 - x % 9) % 9])

    nums[line[0] % 9] = max(nums[line[0] % 9], line[0])

    for j in range(5):
        line[j] = line[j + 1]
    line[5] = x

print(ans)
```

ШКОЛКОВО