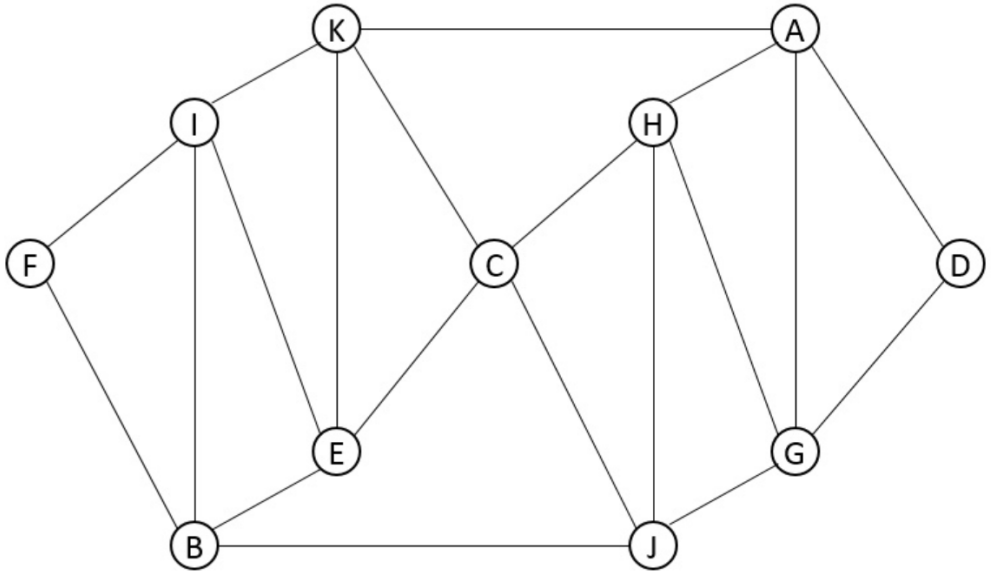
МОУ
"Средняя общеобразовательная школа №72"
Наш адрес: 410033, г.Саратов, ул.Бережная, 1

Разбор ЕГЭ по информатике март (2023)

Задание 1. Условие

На рисунке схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о протяжённости каждой из этих дорог (в километрах).

Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите суммарную протяжённость дорог в пункт C.



	П1	П2	П3	П4	П5	П6	П7	П8	П9	П10	П11
П1				13				24			
П2			10		11					16	17
П3		10		8	22		14				
П4	13		8			9		12			
П5		11	22				15			10	
П6				9			12	20	16		
П7			14		15	12			18		
П8	24			12		20			10		
П9						16	18	10		19	
П10		16			10				19		6
П11		17								6	

Задание 1. Решение

1.У нас имеется 2 пункта, из которых ведут по 2 дороги - F и D (это П1 и П11). Т.к. граф симметричный, точно знать не нужно.

2.Из П1 ведут пути в П4 и П8 (A, G или I, B). Из П11 ведут пути в П2 и П10 (A, G или I, B).

3.Заметим, что кроме F (D), пары A, G и I, B имеют по одному общему “соседу” - E (H). Для П4 и П8 таким пунктом является П6, для П2 и П10 - П5.

4.Заметим, что кроме A (B), пары G, H и I, E имеют по одному общему “соседу” - J (K). Для П4 (П8) и П6 таким пунктом является П9, для П2 (П10) и П5 - П3.

Итак, мы определили пункты E, H, J, K - это П6, П5, П9, П3. Осталось определить общий для всех пункт C и посчитать сумму длин дорог в него. Общий пункт - П7, длина дорог - 14 + 15 + 12 + 18 = **59**.

Задание 2. Условие

Логическая функция задаётся выражением:

$$(\cdots \to \cdots) \equiv \overline{(\cdots \to \cdots)}$$

На месте $\cdots$ в ней могут стоять переменные x, y, z, w. Определите, какое количество раз функция становится истинна, при любых перестановках переменных.

Примечание: переменные могут принимать значения 0 и 1.

Задание 2. Решение

Составим таблицу истинности для четырех переменных

x	y	z	w
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

1. Импликация ложна, если ложно следствие, т.е. из единицы следует ноль.

2. Эквивалентность истинна, если обе её части либо одновременно ложны, либо одновременно истинны.

3. Вторая импликация стоит под отрицанием, т.е. её значение инвертируется.

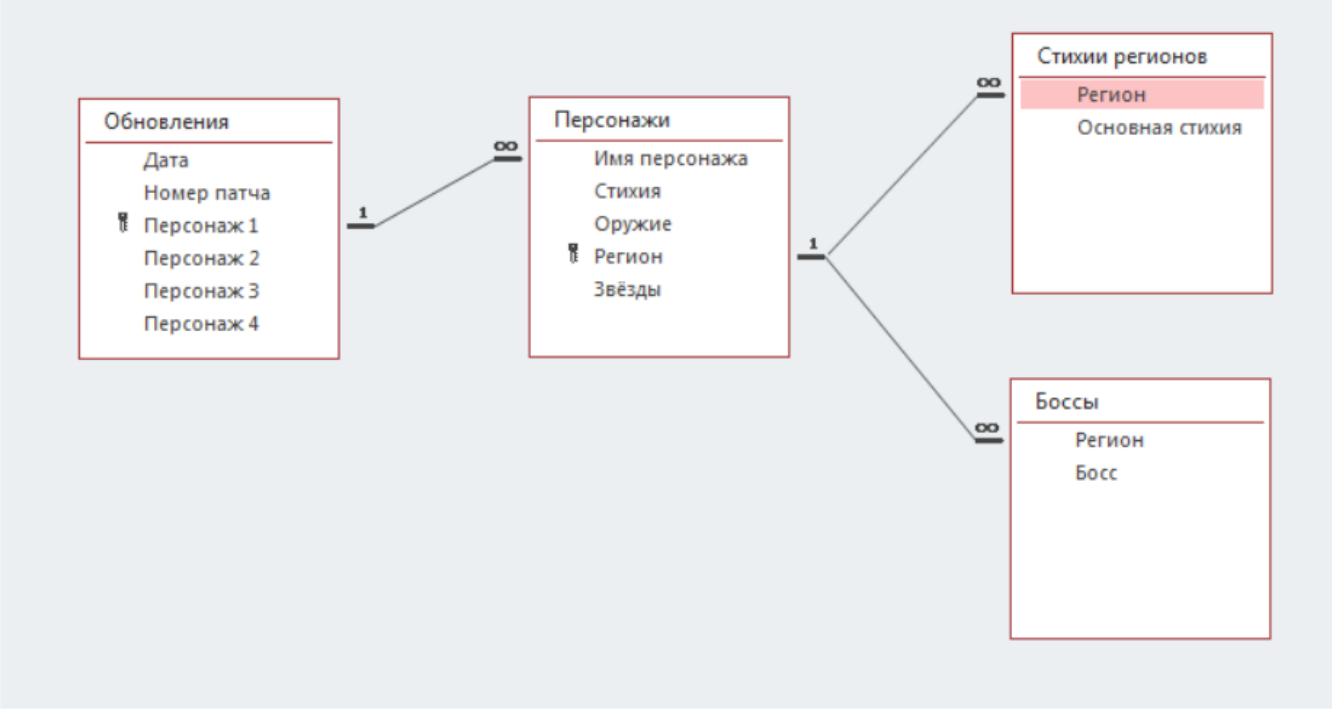
4. Из вышесказанного следует, что импликации в обеих частях должны принимать разные значения. Выделим зеленым цветом подходящие строки

5. Помним, что переменные могут находится на любых местах, т.е как xyzw, так и zwyx. Чтобы посчитать кол-во таких вариантов, нам нужна будет формула перестановок:
$P_n = n!$
Итого, 4! = 24 перестановки. Каждая из таких перестановок имеет свои 6 вариантов, найденных выше. 24 * 6 = **144**.

Задание 3. Условие

Таблица «Обновления» содержит информацию о дате и номерной версии обновления, а также о добавленных персонажах. Таблица «Персонажи» содержит информацию о некоторых характеристиках персонажей: их стихии, оружии, регионе проживания и редкости. Таблица «Стихии регионов» содержит информацию о стихиях, распространённых в определённом регионе. Таблица «Боссы» содержит информацию о боссах всех регионов.

На основании приведённых данных ответьте, в течение какого периода, в днях, длилось обновление, в котором в игру был добавлен легендарный (5 звёзд) персонаж, обладающий наиболее редко используемой (которой обладают меньше всего персонажей) стихией и **не** владеющий наиболее редко используемым (которым владеют меньше всего персонажей) оружием, проживающий в регионе со своей родной стихией, в котором находится наибольшее количество боссов.



Задание 3. Решение

1.Чтобы найти нужного персонажа, переходим в таблицу “Персонажи” и создаём две сводные таблицы - для определения редких стихии и оружия.

2.В одной из сводных таблиц столбцами (строками) выбираем “Стихии”, а значениями - “Имя персонажа”, чтобы определить, какой стихией владеет сколько персонажей. Самые редкие - Дендро и Гео.

3.Во второй сводной таблице столбцами (строками) выбираем “Оружие”, а значениями - “Имя персонажа”, чтобы определить, каким оружием владеет сколько персонажей. Самое редкое - двуручный меч.

4.Отправляемся в таблицу “Персонажи”, где, с помощью фильтра, “отсеиваем” ненужное нам. Итак, нужна редкость 5 звёзд, стихии Гео и Дендро, оружие - НЕ двуручный меч.

5.Определяем регионы. В таблице “Стихии регионов” с помощью фильтра, “отсеиваем” регионы с родными стихиями Дендро и Гео. Это Ли Юэ и Сумеру.

6.В таблице “Боссы” создаём сводную таблицу, в которой строками (столбцами) будет “Регион”, а значениями - “Босс”. Наибольшее количество боссов в регионе Ли Юэ (среди найденных в п.5).

7.В таблице “Персонажи” вносим в фильтре найденные данные (регион - Ли Юэ). Итак, персонаж - Чжун Ли.

8.С помощью фильтров по персонажам (в таблице “Обновления”) определяем, что персонаж Чжун Ли был добавлен в игру в обновлении 1.1, которое длилось с 11 ноября по 23 декабря 2020 года (не включительно). В произвольной ячейке находим разность, между этими датами, которая равна **42** дням.

Задание 4. Условие

Для кодирования букв А, Н, Е, С, Т, З, И, О, Л, Г используется неравномерный двоичный код, удовлетворяющий условию Фано. Известно, что для кодирования гласных букв потребовалось настолько меньше знаков, чем для кодирования согласных, сколько их требуется для кодирования слова ЛЕТО, а для кодирования слова ЛОТОС потребовалось на 6 знаков больше, чем для кодирования слова ЛЕТО. Какое наименьшее количество двоичных знаков потребуется для кодирования слова АНЕСТЕЗИОЛОГ?

Задание 4. Решение

Согласные - НСТЗЛГ
Гласные - АЕИО
ЛЕТО - x
ЛОТОС - x+6 - ЛЕТО-Е+ОС
Самые частые в слове АНЕСТЕЗИОЛОГ - ЕО (2)
1.
Примем Е = 01 (2), О = 10 (2).
2. Из уравнения следует, что при равенстве Е и О, длина С равна 6 знакам.
3. ЛОТОС = 2 + 2 + 6 + ЛТ = 10 + ЛТ
ЛЕТО = 2 + 2 + ЛТ = 4 + ЛТ
Гласные - 2 + 2 + АИ = 4 + АИ
Согласные - 6 + НТЗЛГ
4. Предположим, что Л = 001 (3), Т = 110 (3).
Тогда ЛЕТО = 0010111010 (10), ЛОТОС = 0011011010 + С (16).
Согласные - 13 + НЗГ, гласные - 4 + АИ.
5. Пусть А = 0001 (4), И = 1111 (4).
Тогда гласные - 4 + 4 + 4 = 12, согласные - 13 + НЗГ.
Но для кодирования согласных требуется на 10 знаков больше, чем для кодирования гласных.
12 + 10 = 22
22 - 13 = 9. В дереве нет места, чтобы закодировать НЗГ, суммарно, девятью знаками. Необходимо увеличить длину АИ, чтобы под НЗГ оставалось больше знаков.

Задание 4. Решение

6. Пусть А = 00001 (5), И = 11110 (5). На освободившиеся места 0001 и 1111 поместим Н и З. Тогда гласные - 4 + 5 + 5 = 14, согласные - 13 + 4 + 4 + Г = 21 + Г.

14 + 10 = 24

24 - 21 = 3. Всё ещё не хватает.

7. Осталась одна согласная. Следовательно, заменять мы имеем право только одну гласную, чтобы код был минимальной длины. Пусть А = 111101 (6). На освободившееся место 00001 поместим Г. Гласные - 4 + 6 + 5 = 15, согласные - 21 + 5 = 26.

15 + 10 = 25

25 != 26. Не сходится - на гласные меньше.

8. Увеличив И на один символ - И = 111100 (6), убеждаемся в выполнении условия: гласные - 4 + 6 + 6 = 16. Согласные - 26.

16 + 10 = 26

26 = 26. Кроме того, у нас есть свободное место для С - код 000001 (6).

9. Посчитаем длину слова АНЕСТЕЗИОЛОГ:

6 + 4 + 2 + 6 + 3 + 2 + 4 + 6 + 2 + 3 + 2 + 5 = **45**.

Е	01	З	1110
О	10	Г	00001
Л	001	А	11110
Т	110	И	111110
Н	0001	С	000001

Задание 5. Условие

(А. Игнатюк) Компьютер Барбарик преобразует четырехзначное число N в число R по следующим правилам:
1) Строиться восьмеричная запись числа N, которая называется O.
2) Если количество цифр в восьмеричной записи числа N четно, то в середину этой записи дописывается остаток деления суммы цифр числа O, записанного в троичной системе счисления, на 3.
3) Если количество цифр в восьмеричной записи числа N нечетно, то в предпоследнюю позицию ставится остаток деления наибольшего общего делителя чисел O и N на 5. Также на вторую слева позицию (счет начинается с 1 справо налево) ставится остаток деления количества цифр числа O, записанного в шестеричной системе счисления, на 6.

В ответ укажите количество чисел N, для которой компьютер Барбарик выдаст числа, сумма цифр которых не менее 20.

Задание 5. Решение

1) Для решения задачи понадобится работа с НОД и 3, 6, 8 системами счисления. Для первых двух необходимо самостоятельно написать перевод из 10 системы; для третьей можно воспользоваться встроенной функцией oct(); для нахождения НОД можно воспользоваться функцией gcd из библиотеки math

2) Переходя к решению задачи, сначала необходимо создать перебор четырехзначных чисел, каждое из которых будет переведено в 8 СС. Затем, в зависимости от длины получившегося числа, необходимо перейти к следующим действиям.

3) При четной длине в середину 8-ричной записи необходимо вписать остаток деления на 3 текущего числа при его записи в 3СС

4) При нечетной длине на вторую слева и на вторую справа позиции необходимо вписать остаток деления на 6 текущего числа при его записи в 6СС и НОД между изначальным числом, записанным в 10СС и его остатком деления на 5 соответственно

5) Далее необходимо проверить условие, поставленное в задаче для выбора ответа

Задание 6. Условие

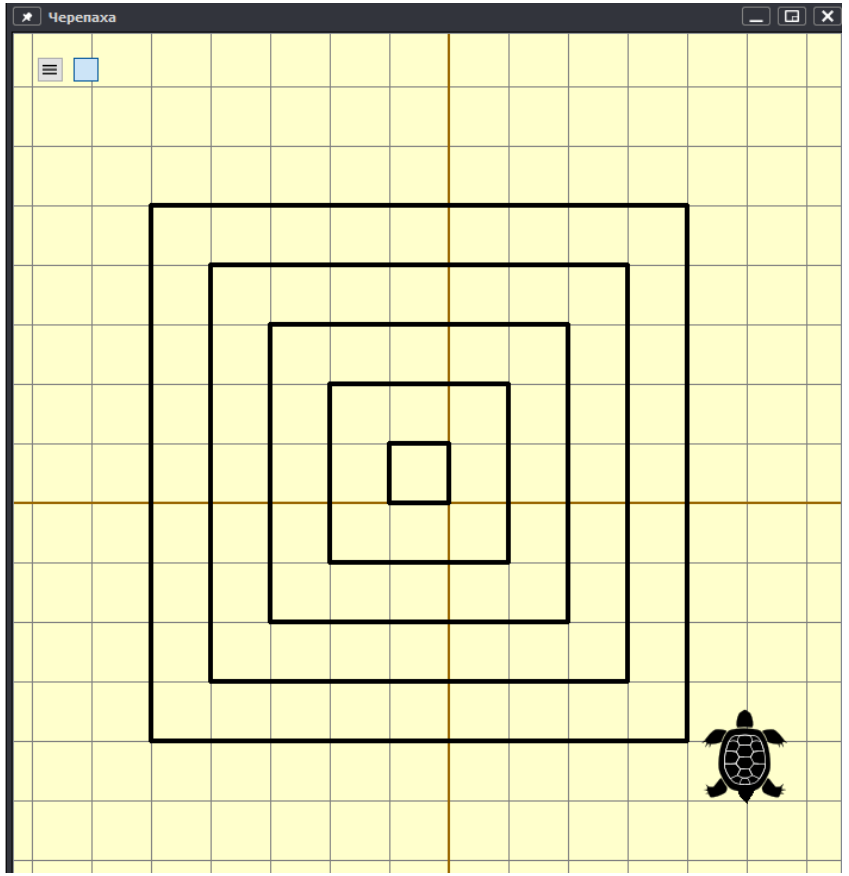
```
Целое X
X присвоить 0
Опустить хвост
Повтори N раз [
Повтори 4 раза [ Вперед 1 + 2*X Влево 90]
X прибавить 1
Поднять хвост
Вправо 180
Повтори 2 раза [Вперед 1 Влево 90]
Опустить хвост
]
```

Определите при каком минимальном значении N сумма площадей получившихся фигур будет превышать 1_000_000_000

Здесь будет работа с переменными. Кумир все-таки тоже является языком программирования, я подумал, что было бы неплохо рассмотреть как работать с переменными в нем.

Задание 6. Решение

```
использовать Черепаха
алг
нач
 . цел x
 . x := 0
 . опустить хвост
 . нц 5 раз
 . . нц 4 раза
 . . . вперед (2 * x + 1)
 . . . влево (90)
 . . кц
 . . x := x + 1
 . . поднять хвост
 . . вправо (180)
 . . нц 2 раза
 . . . вперед (1)
 . . . влево (90)
 . . кц
 . . опустить хвост
 . кц
кон
```



Написал алгоритм для N = 5

Обратите внимание, что перед тем как вводить переменную нужно указать ее тип данных (в данном случае цел)
:= оператор присваивания в Кумир

Можно заметить, что для N = 5 получилось 5 квадратов со сторонами 1, 3, 5, 7, 9. Сумма их площадей будет 1 + 9 + 25 + 49 + 81 = 165. Значит, для общего случая сумма площадей будет равна $\sum_{i=0}^{n}(2 * i + 1)^2$

Значит, остается только подобрать значение N, при котором сумма превысит 10**9. Для этого можно написать программу

```
s = 0
n = 0
for x in range(1000):
    n += 1
    s += (2 * x + 1) ** 2
    if s >= 10**9:
        print(n)
        break
```

```
Pyth
AMD6
Type
>>>
====
909
>>>
```

Задание 7

Человеческий мозг в среднем способен хранить около 3 ПетаБайт информации. Сколько дней можно непрерывно смотреть видео-ролики с разрешением 7680 x 4320 пикселей, частотой 60 кадров в секунду в цветовой палитре, содержащей 16.7 миллионов цветов, пока не заполнится вся память в мозге? При этом все видео-ролики сопровождаются звуковой дорожкой, записанной в 8-канальном режиме с частотой дискретизации 88 КГц и разрешением 24 бита. В качестве ответа укажите целую часть получившегося числа.

Вспомним единицы хранения информации

1 Байт = 2**3 бит
1 КБайт = 2**10 байт
1 МБайт = 2**10 КБайт
1 Гбайт = 2**10 МБайт
1 Тбайт = 2**10 ГБайт
1 Пбайт = 2**10 ТБайт = 2**53 бит

Объем видеоролика за 1 секунду:
Изображения: 7680 * 4320 * 24 (16.7 млн цветов стандартная палитра RGB, в которой 24 бит на цвет) * 60
Звук: 8 * 88000 * 24

Итого, чтобы найти объем видео в секундах, который может запомнить человек, нужно 3 ПетаБайта разделить на сумму объемов изображения и звука за 1 секунду. А чтобы перевести ответ в количество дней, нужно еще разделить его на 86400 (1 день = 24 часа * 60 минут * 60 секунд = 86400 секунд)

$$\frac{3 * 2^{53}}{(7680 * 4320 * 24 * 60 + 8 * 88000 * 24) * 86400}$$ = 6, 54... дня, значит ответ **6**

Задание 8. Условие

Правильной скобочной последовательностью (ПСП) называется строка, в которой каждой открывающейся скобке соответствует ровно одна закрывающаяся. Например, строка ((( ))) является ПСП, строка ((( ))) ( ) тоже является ПСП. А строка ((( ) ПСП не является. Вычислите, сколько существует ПСП, состоящих из 18 скобок?

Эту задачу можно решить как минимум 3-мя способами.

1) Полный перебор всех возможных вариантов и проверка каждого на ПСП. Всего вариантов 2**17 (очевидно, что нет смысла начинать строку с закрывающейся скобки). Здесь главной сложностью будет проверка каждого варианта на ПСП.
2) «Умный» перебор. Генерируем заранее «адекватные» строки и делаем минимальную проверку.
3) Математический способ. Через специальную функцию

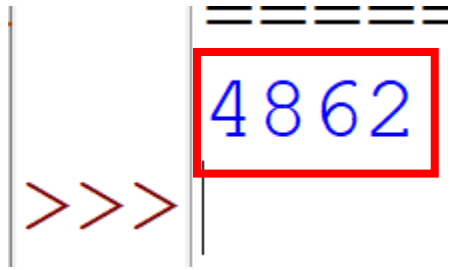
Мы рассмотрим все 3 способа.

Задание 8. Полный перебор

```
#проверка строки на ПСП
def bracket_check(stroka):
    #создаем массив, куда будем накапливать символы
    stack = []
    for i in range(len(stroka)):
        #добавляем в массив символ из строки
        stack.append(stroka[i])
        #если два последних символа равны ()
        if len(stack) >= 2 and stack[-1] == ')' and stack[-2] == '(':
            #удаляем их из массива, потому что это правильная комбинация
            stack.pop(-1)
            stack.pop(-1)
    #в правильной ПСП массив в конце должен остаться пустым, т.к. все правильные комбинации будут удалены из него
    return stack == []

def gen_psp(s):
    if len(s) == 18:
        return bracket_check(s)
    else:
        return gen_psp(s + ')') + gen_psp(s + '(')

print(gen_psp('('))
```



Задание 8. «Умный» перебор

```
def psp(x):
    #diff - разность между количеством ( и )
    diff = x.count('(') - x.count(')')
    if len(x) == 18:
        #для ПСП в конце diff == 0
        return diff == 0
    else:
        #если ) больше, чем (, то дальше идти нет смысла
        #например, из строки ))))))) уже не соберется ПСП
        if diff < 0:
            return 0
        #если ( и ) одинаковое число, то можем ставить только (
        #иначе мы можем создать ситуацию ()(), из которой ПСП тоже не получится
        elif diff == 0:
            return psp(x + '(')
        #если ( больше, чем ), то мы можем ставить в конец строки любую скобку
        else:
            return psp(x + '(') + psp(x + ')')
print(psp('('))
```

```
>>> =====
4862
```

В этом способе мы не используем алгоритм проверки ПСП, а сразу генерируем потенциально подходящие строки

Задание 8. Математический способ

Обратимся к задаче 5711 из общего банка заданий КЕГЭ.

№ 5711 (Уровень: Средний)

(Д. Тараскин) Числа Каталана - числовая последовательность, которая часто встречается в задачах комбинаторики. В частности, этим числом можно охарактеризовать количество правильных скобочных последовательностей длины $2n$.
Это значение можно посчитать по рекуррентному соотношению:

$$C_0 = 1 \quad \text{и} \quad C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i} \text{ для } n \geqslant 1.$$

Оказывается есть функция, которая может посчитать количество ПСП нужной длины. Значит, чтобы ответить на вопрос задачи, нужно посчитать значение функции от 9.

```
def katalan(x):
    if x == 0:
        return 1
    else:
        return sum([katalan(i) * katalan(x - 1 - i) for i in range(x)])
print(katalan(9))
```

```
>>> 
4862
```

Задание 9. Условие

В файле электронной таблицы записано M столбиков, каждый из которых содержит в своей строке 3 натуральных числа. Необходимо найти количество строк, где числа могут являться сторонами треугольника, в котором только один угол от 1 до 60 градусов (не включительно).

Для решения этой задачи потребуется вспомнить одну важную теорему из геометрии, а именно – теорему косинусов. Ну и конечно же, вспомнить про неравенство треугольника, т.к. его можно составить отнюдь не их любых трех отрезков

Задание 9. Решение

1) Проверим неравенство треугольника (Столбец D):
=ЕСЛИ(И(A1+B1>C1;B1+C1>A1;A1+C1>B1);1;0)

2) Распишем теорему косинусов для каждого угла. В формулах это будет выглядеть так (Столбцы E, F, G):
=(A1*A1-B1*B1-C1*C1)/(-2*B1*C1)
=(B1*B1-A1*A1-C1*C1)/(-2*A1*C1)
=(C1*C1-B1*B1-A1*A1)/(-2*A1*B1)

3) Далее каждое получившиеся значение проверяем на принадлежность в заданном в условии диапазоне. Для этого найденный косинус угла должен удовлетворять неравенство 0.5 (cos 60) < COS(...) < 1 (cos 0). Проверяется это с помощью следующих формул (Столбцы H, I, J):
=ЕСЛИ(И(E1>0,5;E1<1);1;0)
=ЕСЛИ(И(F1>0,5;F1<1);1;0)
=ЕСЛИ(И(G1>0,5;G1<1);1;0)

4) После этого необходимо выбрать строки, которые:
А) Имеют цифру 1 в столбце с определением существования треугольника
Б) Имеют только одну цифру 1 при проверке диапазона углов
Осуществить данный отбор можно с помощью формулы (Столбец K):
=ЕСЛИ(И((H1+I1+J1)=1;D1=1);1;0)

Задание 9. Таблица

	A	B	C	D	E	F	G	H	I	J	K	L
1	90	71	71	1	0,1965879786	0,6338028169	0,6338028169	0	1	1	0	
2	89	44	51	1	-0,7540106952	0,9458030403	0,9264555669	0	1	1	0	
3	65	27	53	1	-0,2400419287	0,9150943396	0,6111111111	0	1	1	0	
4	68	25	94	0	1,029148936	1,003989362	-1,055	0	0	0	0	
5	68	61	47	1	0,2277642135	0,4868585732	0,7396335583	0	0	1	1	
6	44	67	60	1	0,7652985075	0,1982954545	0,4791383989	1	0	0	1	
7	86	37	77	1	-0,0171990172	0,9027484144	0,4456316782	0	1	0	1	
8	6	57	4	0	7,081140351	-66,60416667	4,779239766	0	0	0	0	
9	18	42	86	0	1,223145072	1,92377261	-3,510582011	0	0	0	0	
10	86	68	77	1	0,3014705882	0,6569767442	0,5207763338	0	1	1	0	
11	61	77	49	1	0,610787172	0,03228504517	0,7716627635	1	0	1	0	
12	21	95	50	0	1,166736842	-2,897142857	1,745864662	0	0	0	0	
13	64	97	30	0	1,067525773	-1,14921875	1,015222294	0	0	0	0	
14	78	12	76	1	-0,0899122807	0,9881916329	0,2414529915	0	1	0	1	
15	8	81	56	0	1,061838624	-3,751116071	2,69212963	0	0	0	0	
16	87	30	96	1	0,4421875	0,9509698276	-0,1431034483	0	1	0	1	
17	32	37	69	0	1	1	-1	0	0	0	0	
18	37	34	82	0	1,167682927	1,143210283	-1,668918919	0	0	0	0	
19	53	68	50	1	0,6345588235	0,129245283	0,6843784684	1	0	1	0	
20	24	6	74	0	5,558558559	1,693693694	-16,88888889	0	0	0	0	
21	31	26	77	0	1,40959041	1,30163385	-2,662531017	0	0	0	0	
22	45	76	26	0	1,120192308	-1,314102564	1,041666667	0	0	0	0	
23	68	6	41	0	-5,908536585	1,12428264	3,650735294	0	0	0	0	

Зеленым цветом выделены примеры подходящих под условия строк (столбец D проверяет можно ли составить треугольник, столбцы H – J проверяют, что угол принадлежит диапазону от 1 до 60 градусов)

Задание 10

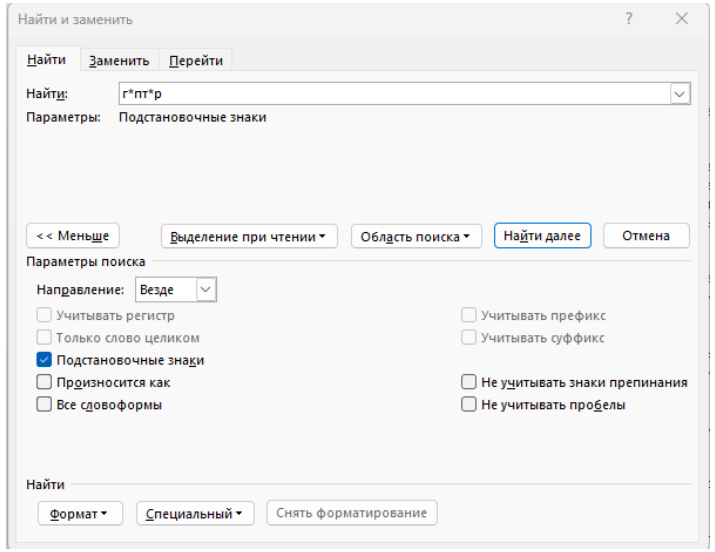
Назовём магической строкой такую, которая начинается с буквы “г”, заканчивается буквой “р” и имеет в середине сочетание “пт” С помощью текстового редактора найдите длину самой короткой магической строки в первой главе произведения Элиезера Юдковского “Гарри Поттер и методы рационального мышления”.

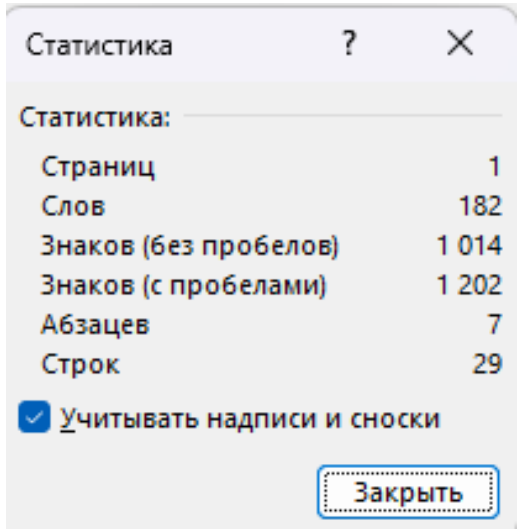
Решение

1. При поиске, в меню расширенного поиска применяем подстановочные знаки. Ищем строку г*пт*р.

2. Пролистываем строки, с помощью кнопки “Найти далее”, и параллельно смотрим, сколько строка занимает символов, нажимая на количество слов (в нижней строке).

3. Всего в тексте 2 большие строки, поиск в которых, в конце концов, приводит к ответу - **38**.





Задание 11

Постоянная Авогадро — физическая величина, численно равная количеству атомов в 1 моле любого вещества.

$N(A) = 6{,}02 \cdot 10^{23}$

Студенты медицинского университета изучают молекулу аспирина $C_9H_8O_4$. Суперкомпьютер может хранить в базе данных названия каждого атома изучаемой молекулы. Как известно, в таблице Менделеева содержится 118 химических элементов. На хранение информации о названии каждого атома выделено минимально возможное количество бит. Сколько ПетаБайт выделит суперкомпьютер для хранения данных обо всех атомах молекулы аспирина? В качестве ответа укажите целую часть получившегося числа.

Основная сложность этой задачи в том, чтобы определить количество атомов в молекуле аспирина. Количество атомов, из определения N(A), напрямую зависит от количества моль вещества. По формуле можно сказать, что 1 молекула аспирина содержит 9 моль углерода, 8 моль водорода и 4 моль кислорода. Итого 21 моль вещества.

Объем памяти на 1 атом = 7 бит (2**7 > 118)

Итого, объем (в битах) для хранения данных обо всех атомах будет = 7 * 21 * 6,02 * 10**23

А чтобы перевести в ПетаБайты (см задачу 7), нужно этот объем поделить на 2**53

$\frac{7 * 21 * 6{,}02 * 10^{23}}{2^{53}}$ = 9824807634.117859,

значит ответ **9824807634**

(по заданию нужно записать целую часть числа)

Задание 12. Условие

Исполнитель РЕДАКТОР 2.0 работает со строками, состоящими из цифр. Логика его работы очень проста: первые 3 найденные цифры он заменяет на их сумму и начинает обход заново. И так он делает до тех пор, пока в строке будет хотя бы 3 цифры.
Какое количество замен выполнит РЕДАКТОР в строке, состоящей из 1..12..23…34…45…56…67…78…89…9 (в каждой последовательности одинаковых цифр их содержится ровно 100 штук) и какое число останется после выполнения алгоритма? В качестве ответа укажите эти два значения без разделителя
К примеру, если бы количество замен было 123, а оставшееся число 45, то ответом считалась бы последовательность 12345.

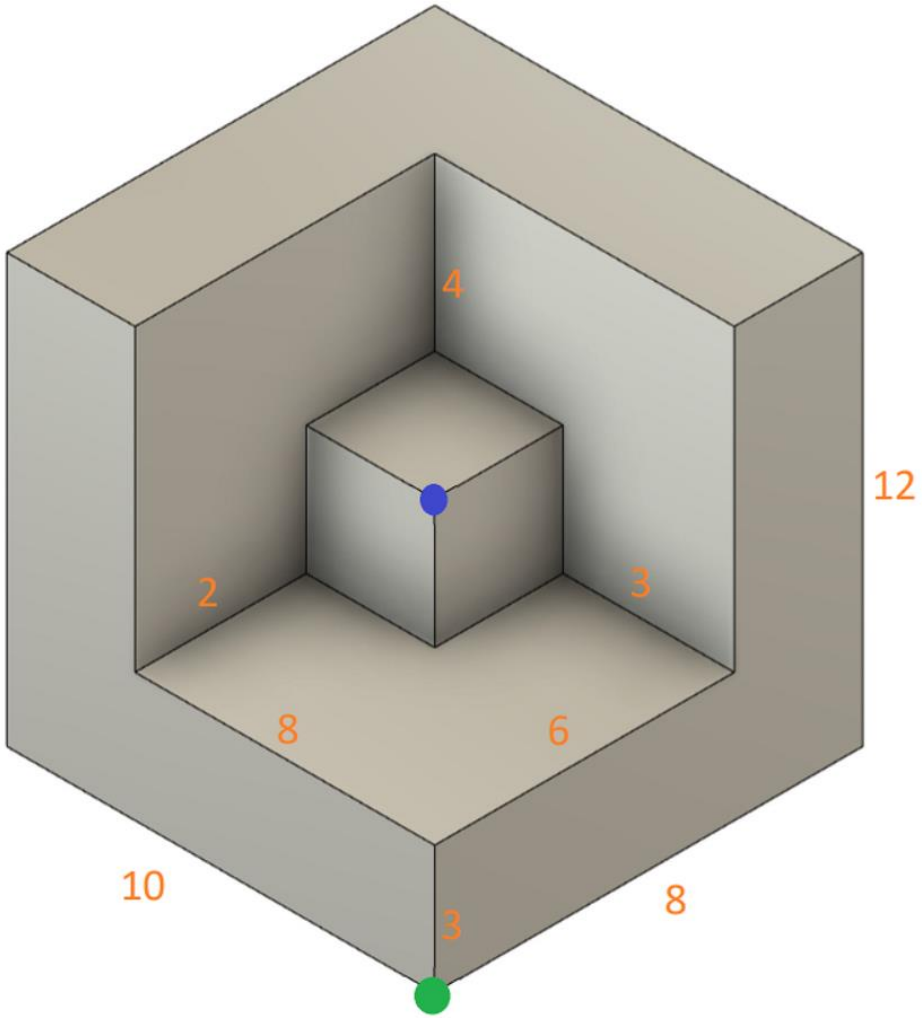
Здесь основная сложность в том, что в явном виде не дан алгоритм как работать со строкой. Но можно заметить, что у нас есть ограничивающее условие - пока в строке будет хотя бы 3 цифры. И также у нас есть действие, которое должен выполнить алгоритм - первые 3 найденные цифры заменить на их сумму.

Задание 12. Решение

```
#входная строка
s = 100*'1'+100*'2'+100*'3'+100*'4'+100*'5'+100*'6'+100*'7'+100*'8'+100*'9'
#счетчик количества замен
k = 0
#пока в строке будет хотя бы 3 цифры
while len(s) >= 3:
    #забираем первые 3 цифры и ищем их сумму
    summa = int(s[0]) + int(s[1]) + int(s[2])
    #во входной строке первые 3 элемента слева заменяем на summa
    s = str(summa) + s[3:]
    #увеличиваем счетчик, потому что замена произойдет в любом случае
    k += 1
print(k, s)
```

```
665 18
>>>
```

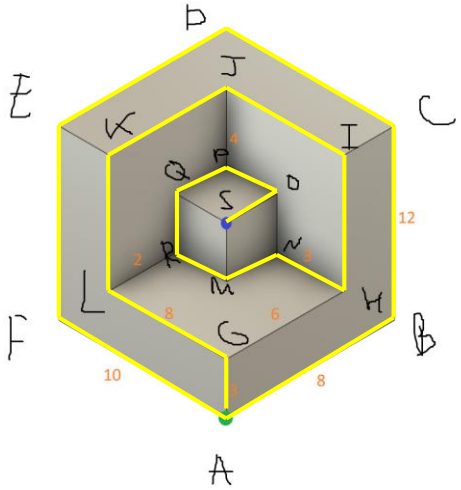
Задание 13. Условие

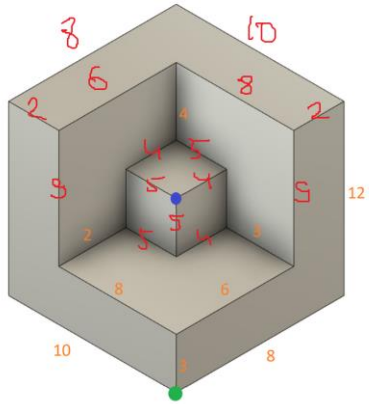


Найдите самый оптимальный по длине путь, проходящий по ребрам объемной фигуры, из зеленой точки в синюю, содержащий максимально возможное количество ребер. Все углы фигуры прямые, числа на ребрах обозначают их длину. Через одно и то же ребро нельзя проходить дважды, ходить можно только по видимым ребрам. В ответ запишите длину этого пути.

Задание 13. Решение

Первым делом восстановим все недостающие размеры ребер и обозначим вершины






```
afedcbaglkjihnmrqpos 133
abcdefaghijklrmnopqs 131
afedcbaghijklrmnopqs 131
abcdefaglkjihnmrqpos 133
```

Далее, можно заметить, что мы можем обойти все внешние ребра по кругу фигуры (неважно по часовой стрелке или против, с точки зрения количества ребер и длины эти пути одинаковые) и вернуться в точку A.

Из точки A в любом случае после этого придется идти в точку G. С этого момента нам уже важно в какую сторону мы идем. Мы снова можем сделать обход по кругу внутренних ребер, но в точку G возвращаться нельзя, т.к. из нее уже не получится сделать ход. Итого, мы можем остановиться или в вершине H, или в вершине L.

Далее нам придется идти во внутренний кубик и, чтобы собрать как можно больше ребер, придется обойти два нижних ребра, подняться вверх и обойти 3 верхних (как показано на рисунке желтым цветом для случая, когда идем из вершины H).

Итого, максимальное количество ребер, которые можно обойти = 19, а чтобы их собрать по сути есть всего 2 маршрута (вообще 4, но по внешним границам неважно в каком порядке идти, поэтому этим можно пренебречь). Осталось только посчитать их длины и выбрать минимальную.

Задание 14. Условие

В уравнении буквами a, b, c обозначены неизвестные цифры

$1a2b3c_{21} * x^2 + b67c99_{14} * x + aaccbb_{17} = 0$

Найдите максимально возможную разницу между корнями этого уравнения. В качестве ответа укажите первые 6 цифр после запятой получившегося числа

Перед нами квадратное уравнение. Спрашивают про разницу между корнями. В квадратном уравнении может быть максимум 2 корня при условии, что дискриминант > 0. Значит, перед тем как вычислять разницу, придется считать дискриминант, проверять его, вычислять корни и только потом находить между ними разницу (по модулю)

Диапазоны значений для параметров a, b, c

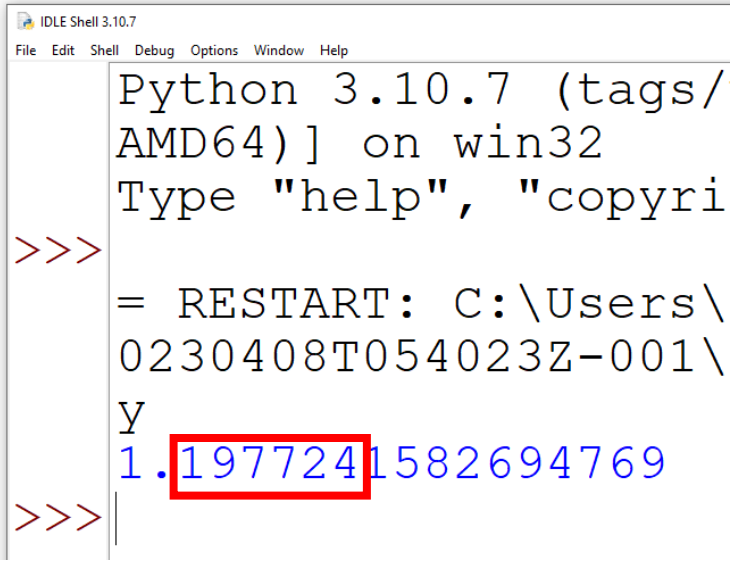
a = [1..16]

b = [1..13]

c = [0.. 13]

Задание 14. Решение

```
from math import sqrt
arr = []
for a in range(1, 17):
    for b in range(1, 14):
        for c in range(14):
            #вычисляем коэффициенты
            a1 = 21**5 + a*21**4 + 2*21**3 + b*21**2 + 3*21 + c
            b1 = b*14**5 + 6*14**4 + 7*14**3 + c*14**2 + 9*14 + 9
            c1 = a*17**5 + a*17**4 + c*17**3 + c*17**2 + b*17 + b
            #рассчитываем дискриминант
            d = b1**2 - 4 * a1 * c1
            #проверяем дикриминант
            if d > 0:
                #вычисляем оба корня
                x1 = (-b1-sqrt(d)) / (2*a1)
                x2 = (-b1+sqrt(d)) / (2*a1)
                #считаем разницу по модулю
                arr.append(abs(x1-x2))
#выводим ответ
print(max(arr))
```



Задание 15. Условие

(А. Игнатюк) Введем утверждение НОД(n, m), которое значит: «наибольший общий делитель чисел n и m не менее тройки» и утверждение СУММЦИФР(n, m), которое значит: «сумма цифр в числе n больше суммы цифр в числе m»
Число A называется подходящим, если формула

$$(НОД(x, 19) \equiv СУММЦИФР(x, 36)) \rightarrow (A < 1000 - x)$$

тождественно истинна (т.е. принимает значение 1) при любом натуральном значении переменной x. Найдите наибольшее натуральное возможное подходящие число A.

Задание 15. Решение

```
Задание 15
1
2
3
4
5
6
7
8
9
10
11
```

```
from math import *
#функция нахождения суммы цифр числа
def d(k):
    c=0
    for i in str(k):
        c+=int(i)
    return c
#функция для вычисления значения логического выражения
def f(x,a):
    return ((gcd(x,19)>=3) == (d(x)>d(36))) <= (a<1000 - x)
#перебор вариантов
print('Задание 15')
for a in range(1,1000):
    if all(f(x,a)==1 for x in range(1,1000)):
        print(a)
```

Задание 16. Условие

Задание 16 .

(Д. Тараскин) Функция F(x) задается рекуррентным соотношением:

F(2) = 0

F(x) = F(x // 2) + 1, при четных x > 2

F(x) = F(x - 1) + 1, при нечетных x > 2

Для скольких X значение функции F(x) = 35

Конечно, это задание невозможно решить полным перебором. Максимальное число, при котором F(x) = 35 будет 2**36 = 68719476736.

Но давайте посмотрим, что происходит внутри самой функции. Если число четное, то мы его делим пополам. Если число нечетное, то мы из него отнимаем 1, получаем четное и следующим ходом делим пополам. Можно заметить, что отнять два раза подряд 1 мы не можем, а вот на деление это ограничение не распространяется. Значит, вместо того, чтобы перебирать числа и проверять их мы будем генерировать различные последовательности команд, выполняя которые мы гарантированно получим число 35 в ответе.

Задание 16. Решение

```
def f(x, n):
    if len(x) == n:
        return 1
    else:
        if x[-1] == '1':
            return f(x+'1',n) + f(x+'2',n)
        else:
            return f(x+'1',n)
print(f('1',35)+f('2',35))
```

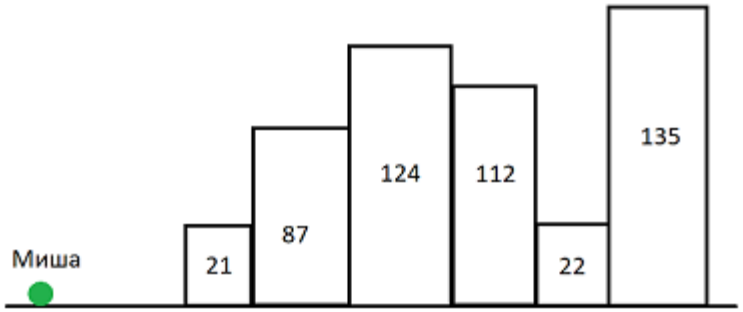
24157817

Командой 1 я обозначил деление. Командой 2 – вычитание. Фактически, я генерирую строки заданной длины, в которых символ 2 не встречается дважды подряд (если последний символ 1, то после нее можно ставить любой, если последний символ 2, то после нее ставить только 1). Начинать можно как с символа 2, так и с символа 1, поэтому функцию придется запускать дважды.

Задание 17. Условие

Миша стоит в начале длинной улицы, на которой построены небоскребы. Все небоскребы имеют разную высоту и записаны в текстовый файл. Какие-то небоскребы выше остальных и их можно увидеть с позиции Миши, а какие-то ниже и их не видно за высокими зданиями.

Для наглядности приведем пример:



Миша со своей позиции увидит здания высотой 21, 87, 124, 135. Дома 112 и 22 он не увидит из-за дома 124.
В качестве ответа запишите какое количество зданий увидит Миша со своей позиции и какой самый высокий дом Миша **не увидит**?
Для тестового примера ответ был бы 4 112.

В этой задаче проверялось умеете ли вы находить максимальный элемент в массиве без использования встроенной функции.
Статистика показывает, что с заданием справились немногие.

Задание 17. Решение

Алгоритм нахождения максимального элемента в массиве:

1) За максимум принимаем самый первый элемент (потому что больше мы пока ничего о массиве не знаем)
2) Перебираем следующие элементы и проверяем:
   2а) Если новый элемент больше максимума, то он теперь он станет максимальным
   2б) Иначе ничего не делаем
3) То число, которое останется в переменной для максимума и будет наибольшим

```
arr = [int(x) for x in open('17-build.txt')]
#массив домов, которые Миша увидит
vis = [arr[0]]
for i in range(1, len(arr)):
    if arr[i] > vis[-1]:
        vis.append(arr[i])
print(len(vis))
#массив домов, которые Миша НЕ увидит
non_vis = []
for i in range(len(arr)):
    if arr[i] not in vis:
        non_vis.append(arr[i])
print(max(non_vis))
```

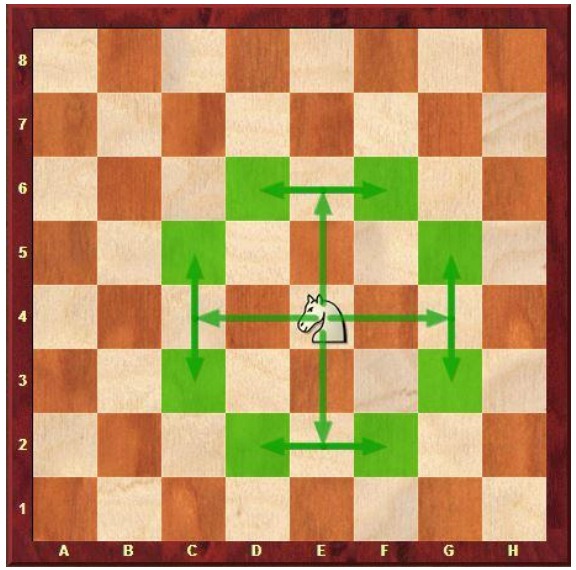
В этой задаче дома, которые Миша увидит, это будут по сути все значения, которые принимала переменная максимума, в процессе обхода массива. Их можно запомнить. А чтобы ответить на второй вопрос нужно из массива удалить те элементы, которые нам встретились на предыдущем шаге.

```
9
9987
```

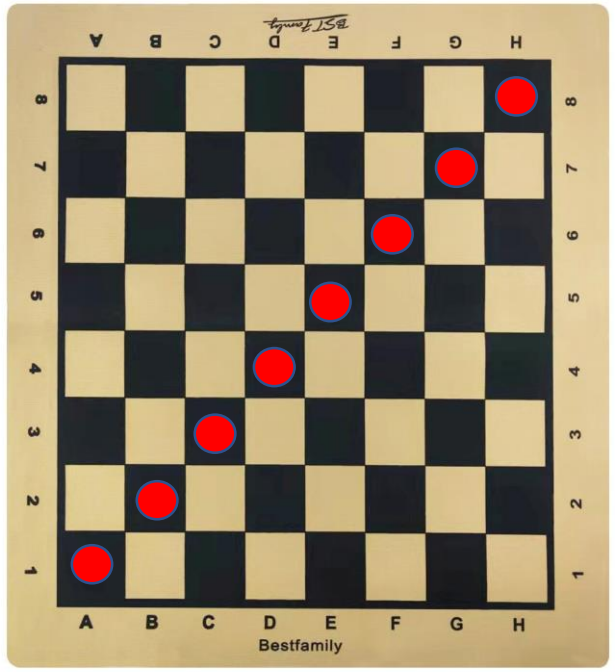
Задание 18. Условие

№ 6945 (Уровень: Базовый)

(Д.Тараскин) Два коня расположены на одной шахматной доске в противоположных углах главной диагонали (главная диагональ состоит из клеток с координатами 1-1, 2-2, 3-3 и т.д.). Размеры доски 30 х 30 клеток. Как известно, конь ходит буквой Г (передвигается на 2 клетки по горизонтали и на 1 по вертикали или наоборот). Кони делают ходы одновременно.
Через какое минимальное количество ходов кони смогут встретиться в одной клетке шахматного поля?



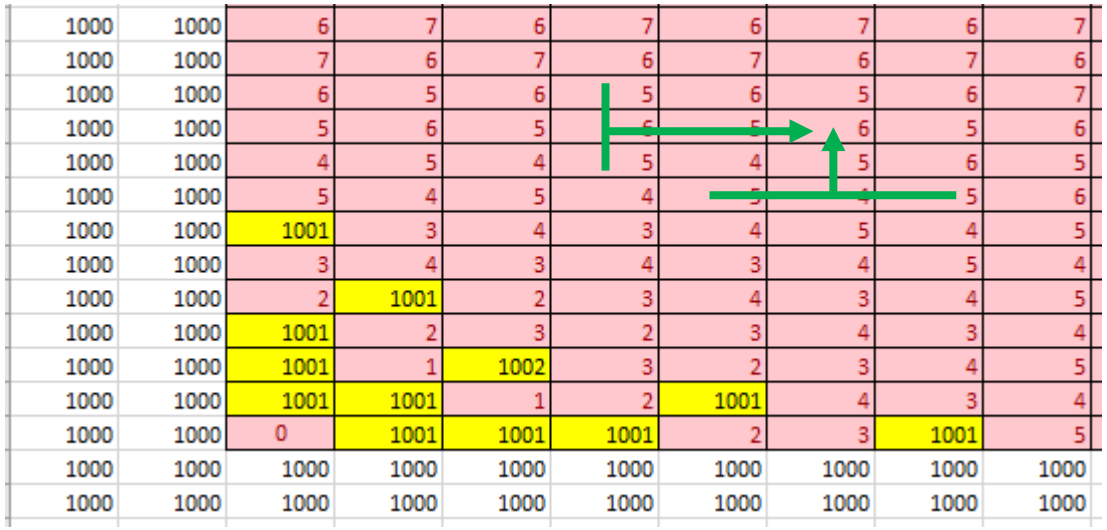
Напомню как ходит конь



Красными точками обозначена главная диагональ

Получается, что первый конь стоит в клетке 1-1, а второй в клетке 30-30.

Задание 18. Решение

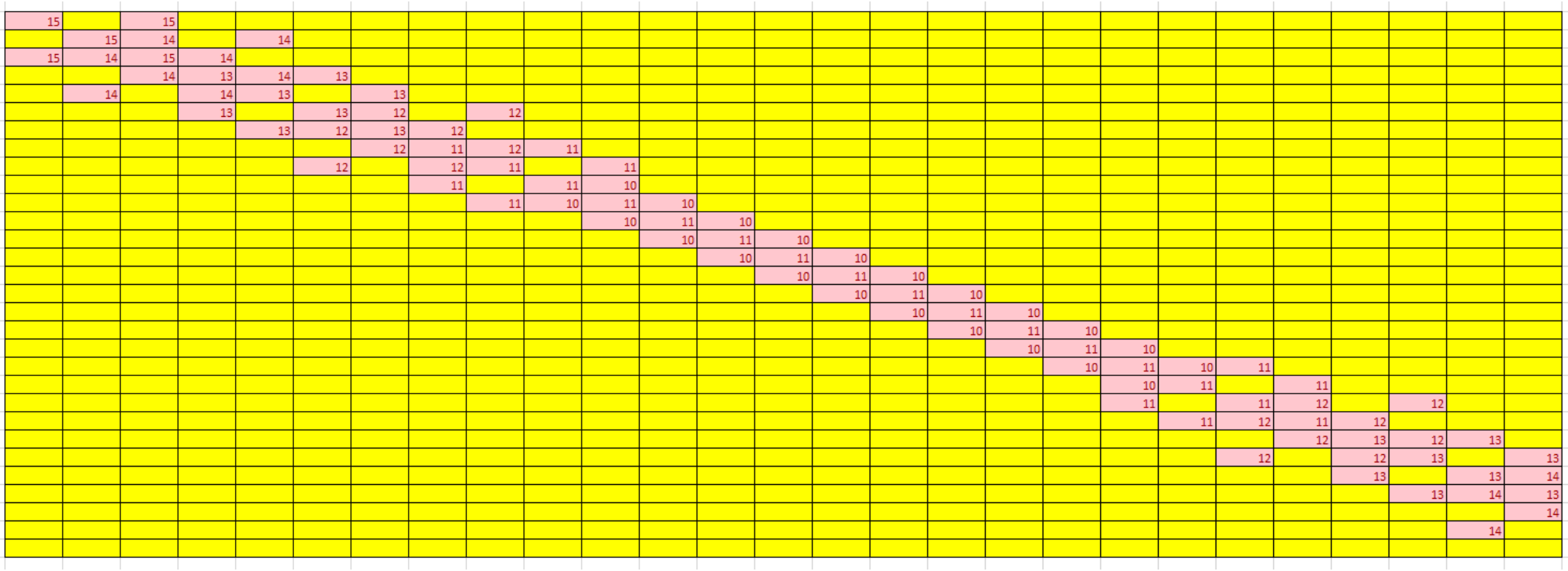


1000	1000	6	7	6	7	6	7	6	7
1000	1000	7	6	7	6	7	6	7	6
1000	1000	6	5	6	5	6	5	6	7
1000	1000	5	6	5	6	5	6	5	6
1000	1000	4	5	4	5	4	5	6	5
1000	1000	5	4	5	4	5	4	5	6
1000	1000	1001	3	4	3	4	5	4	5
1000	1000	3	4	3	4	3	4	5	4
1000	1000	2	1001	2	3	4	3	4	5
1000	1000	1001	2	3	2	3	4	3	4
1000	1000	1001	1	1002	3	2	3	4	5
1000	1000	1001	1001	1	2	1001	4	3	4
1000	1000	0	1001	1001	1001	2	3	1001	5
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Приведу фрагмент электронной таблицы для коня номер 1. Сама доска выделена границами, а по ее бокам в два ряда расставлено число 1000. Это нужно для того, чтобы они не влияли на функцию МИН (в этой задаче количество ходов будет явно меньше 1000).

В клетке, где находится конь, указана цифра 0. Из нее мы начинаем. Далее, для каждой следующей клетки мы пропишем формулу
МИН(k1, k2, k3, k4) + 1,
где k1, k2, k3, k4 – количество ходов в предыдущих клетках.
Я рассматриваю только половину комбинаций из 8 возможных, потому что кони так или иначе должны двигаться навстречу друг другу, чтобы встретиться как можно скорее. На фрагменте я показал возможные траектории движения коня в произвольную клетку. Иными словами, на каждом ходу конь должен улучшать свою позицию (продвигаться вперед), хотя бы по одному направлению (по вертикали или по горизонтали, можно и по обоим). Аналогичную таблицу составляем для второго коня.

Задание 18. Решение



Далее, чтобы сказать ответ, нам нужно составить третью результирующую таблицу, в которой мы будем сравнивать значения из одних и тех же клеток в обеих таблицах и, если значения в них одинаковые, будем их выводить.

Формула имеет вид
ЕСЛИ(значение1 = значение2; значение1; “”)

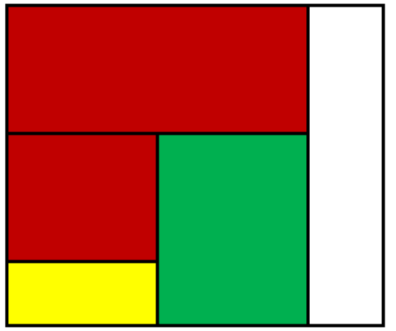
Ответом будет минимальное значение, которое мы увидим на экране. В данном случае это **10**

Задания 19-21. Условия

19) Два игрока – Петя и Ваня играют в следующую игру. Перед игроками расчерчено поле 15x15. В левом нижнем углу поля находится прямоугольник с размерами 5 по оси X и S по оси Y. Первый ход делает Петя. За свой ход игроки могут достроить прямоугольник, увеличив его соответствующую сторону на 2 или на 4 клетки. Игроки договорились, что Пете «соответствует» ось Y, а Ване – ось X.
В примере имеется прямоугольник 2x1 (жёлтого цвета). Петя ходит первым и достраивает прямоугольник, увеличивая его на 2 по оси Y (красный прямоугольник, суммарные размеры – 2x3). После этого Ваня увеличивает его по оси X на 2 (зелёный прямоугольник, суммарные размеры – 4x3). Вторым ходом, Петя делает любое увеличение – и выигрывает (красный прямоугольник, суммарные размеры – 4x5, площадь – 20).
Победителем считается тот, кто первым достигнет противоположной стороны. Известно, что Петя победил своим вторым ходом и, после его хода, незанятыми оказались 120 клеток. Напишите наименьшее S, при котором это возможно.

20) Для игры, описанной в задании 19, напишите минимальное и максимальное количество занятых, после первого хода Вани, клеток, при условии победы Пети своим вторым ходом.

21) Для игры, описанной в задании 19, напишите максимальное из таких S, при которых Ваня побеждает своим третьим ходом, причём занятая площадь максимально возможна.



Задание 19. Решение

Площадь поля = 15*15 = 225
Занято, после П2 - 105
Начальная площадь - 5s

1.Т.к. необходимо наименьшее S, представим, что игроки делают ходы, максимально увеличивающие площадь, т.е:

П1 = 5(s + 4)
В1 = 9(s + 4)
П2 = 9(s + 8)

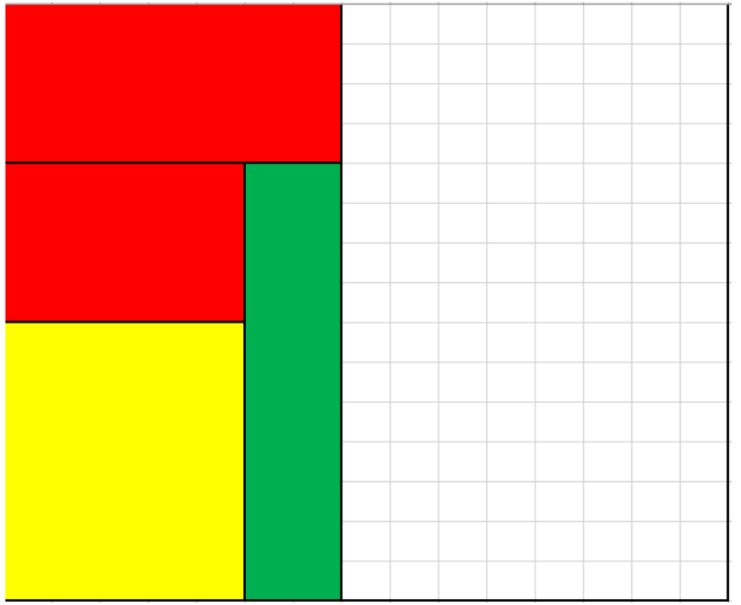
Как известно, игра окончилась вторым ходом Пети - и мы можем приравнять данное уравнение к 105:

9(s + 8) = 105
9s + 72 = 105
9s = 33

s нецелое, т.е. этот вариант ходов нам не подходит.

2.Предположим, что Ваня своим ходом увеличил площадь минимально:

П1 = 5(s + 4)
В1 = 7(s + 4)
П2 = 7(s + 8) = 105
7s + 56 = 105
7s = 49
s = **7**



Задание 20. Решение

Мы уже узнали, что минимальное s равно 7, следовательно, именно при s=7 достигается минимум закрашенных, после В1, клеток (для достижения минимума игроки обязаны делать ходы, минимально увеличивающие площадь):

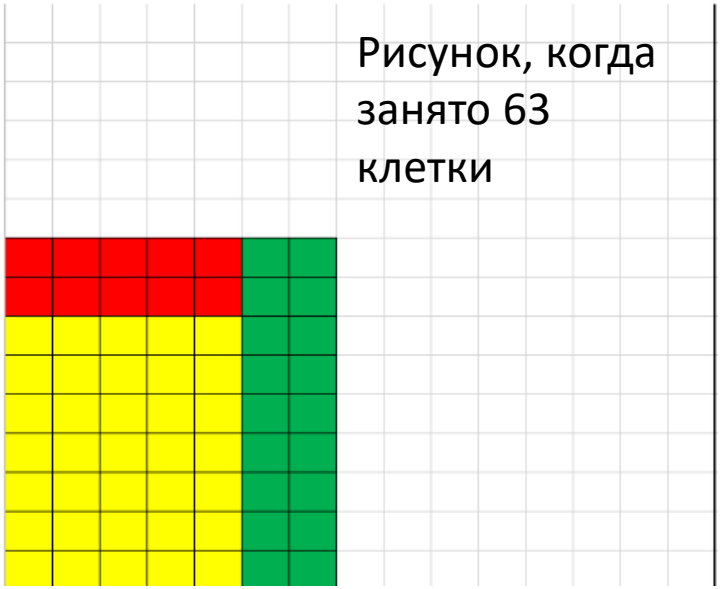
П1 = 5(7 + 2) = 45

В1 = 7(7 + 2) = **63**

Максимальное же s равно 12, т.к. это - максимальное число, при котором Петя не выигрывает своим первым ходом, т.е. ход доходит до Вани (при этом, Петя делает минимальный ход, чтобы не победить, а Ваня - максимальный, чтобы площадь достигла максимума):

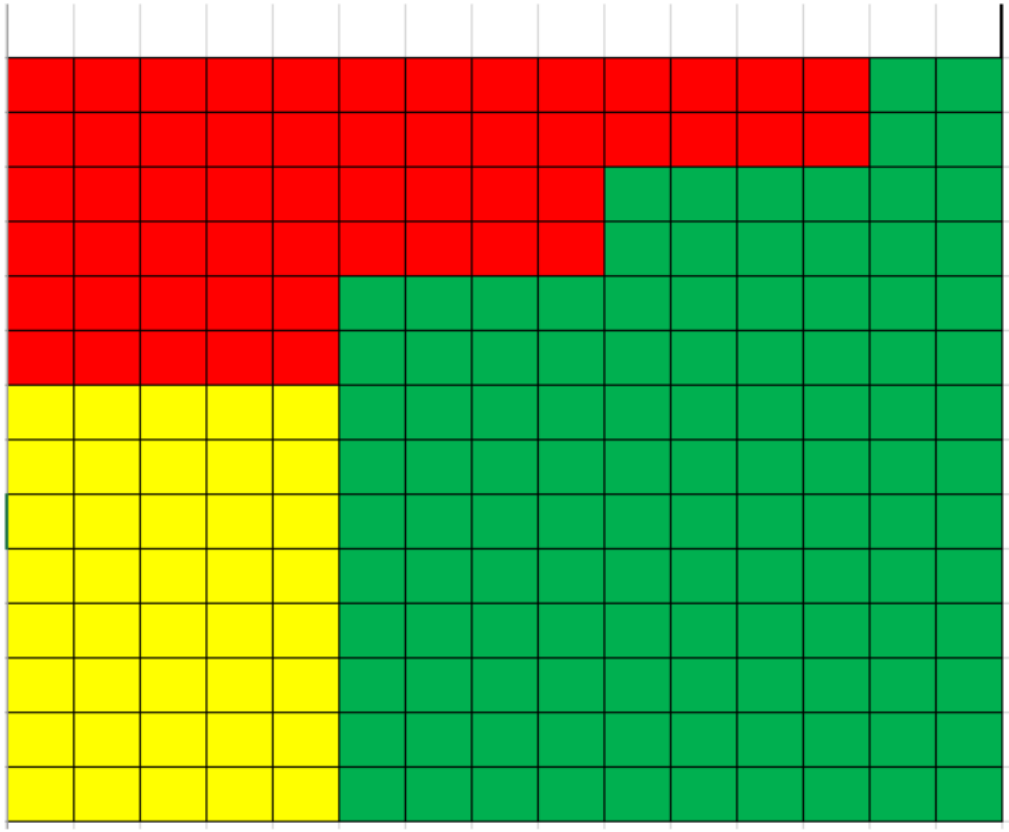
П1 = 5(12 + 2) = 70

В1 = 9(12 + 2) = **126**



Задание 21. Решение

Игра должна дойти до третьего хода Вани, т.е у Пети есть 2 варианта ходов:
+2+2+2
+4+2+2
+4+4+2
+4+4+4
Нас интересует вариант +2+2+2, поскольку, в этом случае, s максимально. При таком варианте, Петя не побеждает третьим ходом, при s <= 8, т.е. максимальное значение **s = 8**.
Для интереса посчитаем закрашенную, после В3, площадь:
П1 = 5(8 + 2) = 50
В1 = 9(8 + 2) = 90
П2 = 9(10 + 2) = 108
В2 = 13(10 + 2) = 156
П3 = 13(12 + 2) = 182
В3 = 15(12 + 2) = 210



Задание 22. Условие

В файле содержится информация о совокупности N вычислительных процессов, которые могут выполняться параллельно или последовательно. Будем говорить, что процесс B зависит от процесса A, если для выполнения процесса B необходимы результаты выполнения процесса A. В этом случае процессы могут выполняться только последовательно.

Информация о процессах представлена в файле в виде таблицы. В первом столбце таблицы указан идентификатор процесса (ID), во втором столбце таблицы — время его выполнения в миллисекундах, в третьем столбце перечислены с разделителем «;» ID процессов, от которых зависит данный процесс. Если процесс является независимым, то в таблице указано значение 0.

Одновременно с началом выполнения процессов и далее каждые 100 мс после завершения последнего сеанса логирования сроком на 30 мс запускается программа, которая фиксирует ID процессов, выполняющихся в эти периоды (в 0-ую миллисекунду, 130-ую, 260-ую и тд). Если время начала/конца процесса совпало с временем начала/окончания сеанса логирования, то этот процесс тоже учитывается. Найдите количество процессов, которые успеет зафиксировать программа.

Задание 22. Тестовый пример

ID процесса	Время выполнения	От чего зависит	Время начала выполнения (от момента запуска)	Время завершения выполнения (от момента запуска)
1	5	0	0	5
2	7	1	5	12
3	2	1; 2	12	14
4	8	0	0	8
5	5	0	0	5

Допустим, что у нас такая таблица и процесс-логгер запускается каждые 2 мс после завершения предыдущего сеанса сроком на 4 мс
Итого, процесс будет работать:
0 – 4 мс
6 – 10 мс
12 – 16 мс
С 0 по 4 мс он «поймает» процесс № 1, 4, 5.
С 6 по 10 мс – 2, 4
С 12 по 16 – 2, 3.
Итого он «увидит» 7 процессов (некоторые процессы могут попасть в несколько сеансов)

Задание 22. Алгоритм

1) Найти время выполнения всей совокупности процессов

2) Для каждого процесса запоминать время его начала (время ожидания, пока завершатся зависимые процессы) и время окончания (время ожидания + время выполнения)

3) Когда нашли максимальное время выполнения, то нужно интервал от момента запуска до момента завершения, «разбить» на сеансы (0 – 30 мс, 130 – 160, 260 – 290, 390 – 420 и тд).

4) Далее проверяем каждый процесс на попадание в какой-нибудь из сеансов. Если его время начала попадает в этот промежуток, или время окончания, то такой процесс будет зафиксирован.

5) Все процессы, которые прошли условие в п4 нужно посчитать и записать в ответ

Задание 23. Условие

Учитель информатики Иван Васильевич составляет для школьников 24-ю задачу и генерирует случайную строку из 10000 символов, среди которых могут встречаться только A, B и C. Найдите вероятность того, что в данной строке будет 10 или более подряд идущих одинаковых символов. В качестве ответа укажите первые 6 цифр после запятой в получившемся числе.

Примечание: вероятность нужно рассчитывать по классической формуле M / N, где M - число благоприятных исходов, N - общее количество исходов

Далее в задаче будем использовать обозначения:
k – количество символов, из которых можно составлять строку (в данном случае 3)
n – минимальное количество подряд идущих одинаковых символов (в данном случае 10)
length – длина строки (в данном случае 10000 символов)

Задание 23. Алгоритм

Решать данную задачу будем с помощью динамического программирования.

Заведём список a длиной $n + 1$.
Оставим элемент $a[0]$ без рассмотрения.
Пусть $a[i]$ при $1 \leq i < n$ - это количество m-буквенных строк $(1 \leq m \leq length)$, на конце которых ровно i одинаковых букв подряд, но при этом нет n или более одинаковых букв подряд внутри строки.
А $a[n]$ - количество строк, в которых есть n или более одинаковых букв подряд (назовём такие сроки "хорошими").
Заметим, что сумма всех чисел в списке a должна равняться k^m - количество всех m-буквенных слов, составленных из k различных символов.

Распишем формулу расчёта количества таких слов при переходе от строк длины m к строкам длины $m + 1$.

1. $a[1] = (k - 1) \cdot \sum_{i=1}^{n-1} a[i] = (k - 1)\cdot$ (количество всех m-буквенных строк, к которым приписали букву, отличную от самой последней, за исключением "хороших" строк)
2. $a[i] = a[i - 1]$ при $2 \leq i < n$, поскольку для получения строки с i одинаковыми символами на конце, необходимо приписать такую же букву к строке, у которой $(i - 1)$ одинаковых символов на конце.
3. $a[n] = k * a[n] + a[n - 1]$, поскольку к "хорошим" словам, в которых уже попалось n одинаковых букв подряд, можно приписывать любую из k доступных символов, плюс в категорию "хороших" попали $a[n - 1]$ слов, к концу которых приписали одну такую же букву, как и $(n - 1)$ предыдущая.

Наша динамика стартует с однобуквенных $(m = 1)$ строк: $a[1] = k, a[i] = 0$ при $i \geq 2$.
Запускаем цикл for в диапазоне от $m = 2$ до $m = length$, на каждом шаге полностью пересчитываем числа в списке a.

Искомая вероятность $p = \frac{a[n]}{sum(a)}$.

Задание 24. Условие

(А. Игнатюк) В текстовом файле содержатся латинские буквы из набора A, B, C, O, E, F. Необходимо найти максимальную подстроку, которая будет начинаться с одной буквы A, содержать наибольшее количество букв B и удовлетворять маске A*B*C. При этом меньшая по длине подходящая подстрока не может содержаться в большей по длине подходящей подстроке вида A*B*C.

Задание 24. Решение

1) Прежде всего нужно разделить цепочку букв по элементу A, так как это гарантирует, что в полученных строках не будет другой цепочек, вида A*B*C

2) Затем делается проход по получившимся набором символов в результате разделения. Каждый набор имеет смысл рассматривать лишь тогда, когда в нем находятся буквы B и C

3) Создадим некоторый массив, где будет временно храниться подходящая строка максимальной длины в данной группе, и переменную, которая будет считать количество букв C

4) Таким образом, когда дойдет очередь до последней буквы C в подстроке, то в массив добавление букв будет остановлено, так как иначе C не будет стоять на конце, что не удовлетворяет условие маски

5) Подстроку, содержащуюся в массиве, вновь проверяет на количество букв B (так как все буквы B могут стоять за последним символом C, что не подходит под условие) и складываем количество букв B и длину подстроке в некоторой массив

6) После проделанных действий необходимо отсортировать конечный массив вывести последний элемент, который как раз и будет показывать наибольшее количество букв B и длину подстроки

Задание 25. Условие

Пара чисел называется хорошей, если:

1) Каждое из них равно сумме делителей другого (1 в качестве делителя учитывается, а само число нет)
2) Оба числа НЕ ПОДХОДЯТ под маску 1*5*

В качестве ответа выпишите первые 4 хорошие пары из диапазона 1_000_000 до 10_000_000, упорядочив их в порядке возрастания по меньшему числу в паре. Каждую пару необходимо также упорядочить по возрастанию элементов.

Здесь основной сложностью было реализовать «правильную» идею для подбора чисел в п1. Очевидно, что полным перебором (квадратичной сложностью) это сделать невозможно. И даже в случае если алгоритм написан верно и является оптимальным, то последние ответы будут подбираться пару минут. Это т.н. «проверка на уверенность»))

Задание 25. Алгоритм

1) Перебираем диапазон 1_000_000 до 10_000_000

2) Находим сумму делителей каждого числа

3) Число, полученное в п2 потенциально является парой первому числу (другие числа проверять бесполезно). Но в этом надо убедиться

4) Находим для числа из п2 сумму делителей (если это число, конечно, тоже находится в нужном диапазоне)

5) Число, полученное в п2 точно является суммой делителей первого числа. Если мы для него найдем сумму делителей (п4) и оно совпадет с начальным числом из п1, то получается, что мы нашли хорошую пару.

6) Если условие с маской выполнено, то сортируем и выводим в ответ

Задание 25. Решение

```
from fnmatch import *
from time import time
start = time()
rez = []
#находим сумму делителей
def f(x):
    d = set()
    d.add(1)
    for i in range(2, int(x**0.5) + 1):
        if x % i == 0:
            d.add(i)
            d.add(x//i)
    return sum(d)
rez = []
#перебираем диапазон
for i in range(1_000_000, 10_000_000):
    #находим сумму делителей первого и второго числа
    x = f(i)
    y = f(x)
    #проверка на хорошую пару
    if i == y and i > 1_000_000 and x > 1_000_000:
        #проверка на маску
        if fnmatch(str(i),'1*5*') == False and fnmatch(str(x),'1*5*') == False:
            pair = sorted((i, x))
            if pair not in rez:
                rez.append(pair)
                print(*pair)
            if len(rez) == 4:
                break
print('Время работы программы в минутах', (time() - start) / 60)
```

```
1077890 1099390
1468324 1749212
1669910 2062570
2082464 2090656
Время работы программы в минутах 1.1062580704689027
```

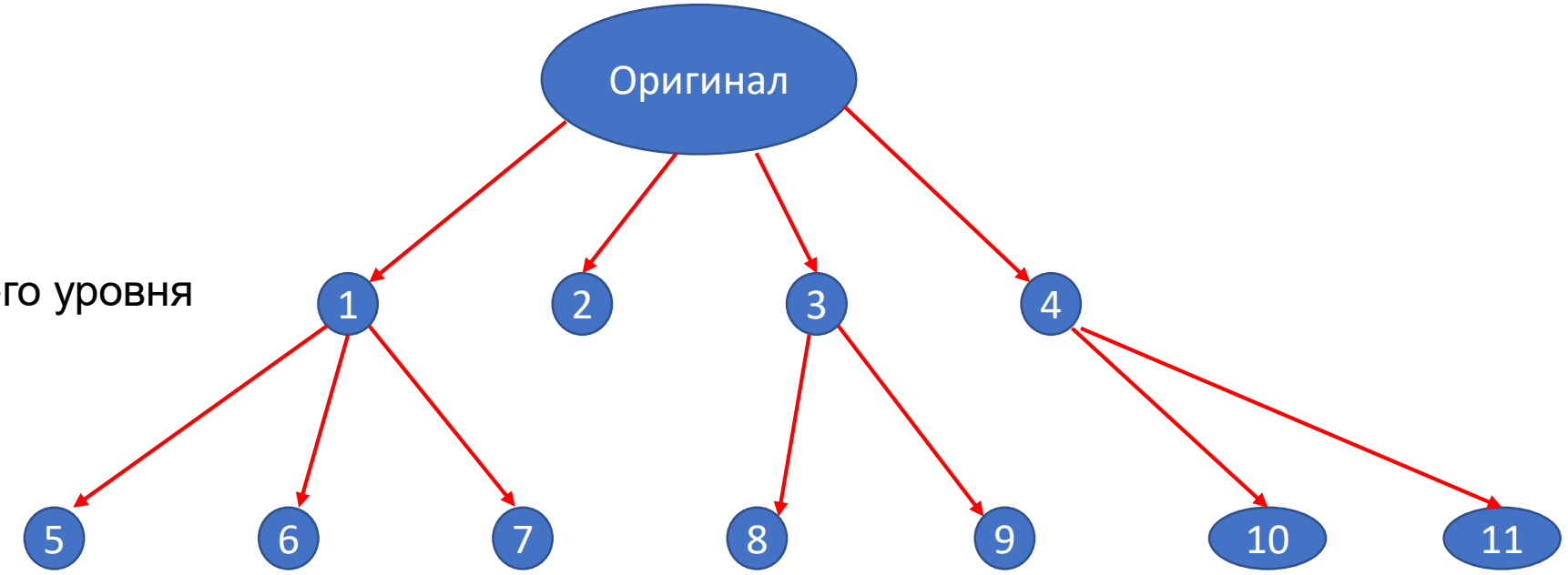
Задание 26. Условие

Имеется оригинальная запись с произвольным количеством просмотров.
Ее могут себе репостить пользователи, у них тоже будет какое то количество просмотров.
Те, кто зарепостил себе оригинальную запись, называются авторами 1-го уровня.
Если оригинальная запись набрала 100 просмотров, а единственный, кто ее репостнул, набрал 500 просмотров, то общее количество просмотров оригинальной записи будет 600.
Если кто-то репостнет себе запись автора 1-го уровня (т.е. автор 2-го уровня), то его просмотры прибавятся вверх по цепочке (и к просмотрам автора 1-го уровня, и к просмотрам оригинальной записи). Каждый автор может делать только один репост, но одного автора могут репостнуть несколько человек.

Количество уровней в этой задаче больше, чем 2. Нужно найти, максимальное число просмотров среди авторов 1-го уровня и среди авторов 2-го уровня.

Задание 26. Алгоритм

В этой задаче модель распространения поста очень похожа на дерево. Корнем является оригинальная запись, а его узлами – авторы. Давайте нарисуем схему и в этом убедимся.



Авторы 1-го уровня

Авторы 2-го уровня

Во входном файле по сути описана вся структура дерева – кто репостнул, это узлы, кого репостнул – узлы, с которыми соединен текущий. Очень удобно будет хранить это в виде словаря, где ключом будет текущий узел, а значениями пара чисел – от кого этот узел произошел и сколько просмотров набрал

Задание 26. Алгоритм

1) Запишем файл в словарь, где в качестве ключей будут выступать ID авторов, а в качестве значений пара чисел – кого автор репостнул и сколько просмотров он набрал.
2) Далее мы знаем, кого репостнул автор. Назовем этого пользователя адресат (у оригинала адресатом будет пустая строка).
3) По ключу с ID адресата мы можем обратиться к элементу словаря (все записи идут в хронологическом порядке, это в любом случае можно будет сделать) и к его просмотрам прибавить просмотры автора из П2.
4) Сменяем адресата и идем по цепочке вверх до тех пор, пока не дойдем до оригинальной записи.
5) Авторы, кто репостнул себе оригинальную запись, это будут авторы 1-го уровня. Авторы, кто репостнул себе авторов 1-го уровня будут авторами 2-го уровня. Далее посчитать ответ не составит труда.

Покажем как работают п2-4 на простом примере

Original – 100
Миша – Original, 50
Петя – Original, 70
Слава – Миша, 20

Original – 100 + 50
Миша – Original, 50
Петя – Original, 70
Слава – Миша, 20

Original – 100 + 50
Миша – Original, 50 + 20
Петя – Original, 70
Слава – Миша, 20

Original – 100 + 50 + 20
Миша – Original, 50 + 20
Петя – Original, 70
Слава – Миша, 20

Задание 26. Решение

```
slov = {}
for line in open('26-main.txt').readlines():
    stat = line.split()
    #создаем пару чисел, которая будет являться значением для ключа с ID автора
    if stat[0] == 'Original':
        slov[stat[0]] = ['', int(stat[1])]
    else:
        slov[stat[0]] = [stat[1], int(stat[2])]
        views = int(stat[2])
        #запоминаем кого репостнул текущий автор
        adr = stat[1]
        #пока адресат не станет пустой строкой (т.е. пока не дойдем до оригинала)
        while adr != '':
            #прибавляем просмотры автора предыдущего уровня к автору следующего уровня
            slov[adr][1] += views
            #сменяем адресата
            adr = slov[adr][0]
#словарь авторов 1-го уровня
a1 = {}
#словарь авторов 2-го уровня
a2 = {}
for key in slov:
    if slov[key][0] == 'Original':
        a1[key] = slov[key][1]
for key in slov:
    if slov[key][0] in a1:
        a2[key] = slov[key][1]
print(len(a1), len(a2))
print(max(a1.values()), max(a2.values()))
```

```
5651135 1081478
```

Задание 27. Условие

В текстовом файле дан отчет из отдела логистики, который занимается погрузкой грузов на борт корабля. В отчете в первой строке указаны вместительность судна M, количество грузов N и вес каждого из них. По имеющимся данным необходимо узнать, сколькими способами можно полностью загрузить корабль.
Пример входных данных:
8 16
7
3
5
11
14
9
4
8
Решим задачу для данного примера:
Рассмотрим сколькими способами можно полностью загрузить корабль, вместительность которого равна M. 3+8+5; 11+5; 7+9; 3+4+9; 4+5+7. Итого полностью загрузить корабль можно 5-ю способами. Ответ для примера: 5

Задание 27. Алгоритм

1) Записываем все числа из файла в массив и находим максимально доступный груз.

2) Создаем словарь, где ключами выступают все числа, не превышающие значения суммарного объема. По умолчанию в качестве значений во всех ключах 0.

3) Начинаем перебирать все доступные грузы. Каждый груз в этом цикле будем называть текущим.

4) Вложенным циклом начинаем перебирать все возможные значения грузов, которые мы можем к нему добавить (логично, что это числа в диапазоне от 1 до максимального значения груза)

5) Если вес текущего груза + добавленный не превышает суммарный объём, то в словаре к ключу <вес текущего груза + добавленный> прибавляем значение, которое соответствуют ключу с добавленным грузом.

6) После завершения вложенного цикла в словаре с ключом, соответствующему текущему грузу, добавляем 1, потому что этот груз есть в массиве и его можно получить без дополнительных операций.

7) После завершения перебора ответ будет находиться в словаре с ключом = суммарному объему