2023

Язык программирования Python как инструмент для получения высоких баллов на ЕГЭ по информатике

МОРОЗОВ В. В.

ОГЛАВЛЕНИЕ

Язык программирования Python как инструмент для получения высоких баллов на ЕГЭ по информатике

Морозов В. В.

– Нехорошо все это получилось, – прошипел юный удав, – пожалуй, мне придется донести Великому Питону о том, что ты здесь наболтал.

Фазиль Искандер

ВВЕДЕНИЕ

На ЕГЭ по информатике 27 заданий, за каждое задание кроме двух последних можно получить по одному баллу, за каждое из последних двух задач можно получить по 2 балла, всего можно получить 29 баллов. На ЕГЭ по информатике допустимо применять все компьютерные инструменты: калькулятор, электронные таблицы, графические редакторы, текстовые редакторы, и, наконец, языки программирования.

Среди допустимых языков программирования (Basic, школьный алгоритмический язык, Pascal, C, Python) именно язык Python ярко выделяется лаконичностью: программа на Python гораздо короче, чем на любом другом языке, а значит потребует гораздо меньше времени. Кроме того, формулировки заданий ЕГЭ часто похожи на код программы Python. Складывается впечатление, что, не смотря на большое количество разрешенных языков программирования, участника ЕГЭ вынуждают изучать и использовать язык Python.

Круг задач, которые удается решить с помощью языка программирования, довольно широк. Вот типы задач, которые успешно решаются на Python: 2, 5, 6, 8, 12, 13, 14, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27. Это дает возможность получить 20 первичных баллов из 29, это 68,9%, то есть 68,9% всех баллов можно получить, с помощью Python! Часто эти алгоритмы однотипны и легки для запоминания, а значит можно использовать их как шаблоны, разучив их однажды. Давайте рассмотрим примеры таких задач и способы их решения на языке программирования Python.

ТИП 2

Тип 2. Сайт[1] «Компьютерный ЕГЭ». Задача № 5424 (Уровень: Сложный).

Логическая функция F задаётся выражением

Формула 1

$$(p_3 \to p_1) \to (p_4 \vee \neg p_2).$$

На рисунке приведён частично заполненный фрагмент таблицы истинности функции *F*, содержащий неповторяющиеся строки. Определите, какой столбец в таблице каждой переменной в выражении.

[1] https://kompege.ru/task

Таблица 1

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

В ответе напишите буквы x,y,z,w в том порядке, который соответствует переменным p_1, p_2, p_3 и p_4 в выражении. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Пример. Функция задана выражением $\neg p_2 \vee p_1$, зависящим от двух переменных, а фрагмент таблицы имеет следующий вид.

Таблица 2

x	y	$\neg x \vee y$
0	1	0

В этом случае p_1 соответствует переменная y, а p_2 – переменная x. В ответе следует написать yx.

Решение[2]. На Python пишем скрипт для построения таблицы истинности формулы 1:

Таблица 3

`print("  p1  p2  p3  p4")`	Печатаем заголовок таблицы истинности.
`for p1 in [False,True]:` `  for p2 in [False,True]:` `  for p3 in [False,True]:` `  for p4 in [False,True]:`	Для каждой из переменных формулы 1 строим вложенные циклы с параметром, в которых перебираются все возможные значения переменных.
`L=(p3<=p1)<=(p4 or not(p2))`	Вычисляем значение логической формулы 1 и записываем это значение в переменную *L*.
`if not(L):` `  print(p1,p2,p3,p4)`	Печатаем только те строки таблицы истинности, для которых значение формулы 1 ложно.

Вот таблица истинности, которую построила программа.

Таблица 4

p1	*p2*	*p3*	*p4*
False	*True*	*False*	*False*
True	*True*	*False*	*False*
True	*True*	*True*	*False*

Сравниваем эту таблицу с таблицей 1. Видим, что только один столбец (p4) принимает значение False. В таблице 1 это может быть только переменная *z*. Только один столбец (p2) принимает значение True. В таблице 1 это может быть только переменная *w*. При этом, становится ясно, как заполнить пустые ячейки таблицы 1:

[2] См Приложение 1.

Таблица 5

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

Сравниваем таблицы 4 и 5 далее: только в одном столбце (p3) значения False, False, True. В таблице 5 это переменная *x*. Только в одном столбце (p1) значения False, True, True. В таблице 5 это переменная *y*. Таким образом, получаем ответ: ywxz.

ТИП 5

Тип 5. Сайт[3] «Компьютерный ЕГЭ». № 5899 (Уровень: Сложный) (Автор задачи Д. Тараскин[4]) Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам:

1) Из цифр, образующих десятичную запись *N*, строятся все возможные двузначные числа (числа не могут начинаться с нуля).

2) Из получившихся двузначных чисел выбираются только те, которые являются простыми.

Каждую цифру трехзначного числа можно использовать ровно столько раз, сколько она встречается в этом числе. К примеру, возьмем число 123. Из него можно составить числа: 12, 13, 21, 31, 23, 32.

Для какого наибольшего N количество выбранных простых чисел будет максимальным?

Решение[5]. На Python пишем скрипт для решения этой задачи. Общая идея программы довольно проста: будем перебирать все трехзначные числа, для каждого трехзначного числа составим список двузначных чисел из его цифр, оставим в этом списке только неповторяющиеся числа и простые числа, будем запоминать трехзначные числа, для которых список простых чисел самый длинный.

Таблица 6

`def isprime(n):` `  L=True` `  k=int(n**0.5)`	Объявляем функцию *isprime*, которая проверяет, является ли натуральное число простым[6] или нет. Функция будет возвращать логическое значение True, если число простое и False в противном случае. Сначала целевая переменная L принимает значение True. Вычисляем $k = \sqrt{n}$, как известно, нужно проверять делимость n на $2, 3, 4, \dots, \sqrt{n}$.
`  for i in range(2,k+1):` `  if n%i==0:` `  L=False` `  break`	Проверяем делимость n на $2, 3, 4, \dots, \sqrt{n}$, если разделится хотя бы на одно из чисел, то значение переменной *L* меняется на

[3] https://kompege.ru/task
[4] https://vk.com/bio_kefir
[5] См. Приложение 2.
[6] Напомним, что натуральное число является простым (prime number), если оно имеет ровно два делителя: само число и единицу (то есть не делится ни на какие другие числа).

`    return L`	False, и цикл прерывается, поскольку дальше проверять делимость нет смысла. Возвращаем целевую переменную *L*.
`def getnumbers(n):` `    a=n//100` `    b=(n%100)//10` `    c=n%10` `    h=[a,b,c]`	Объявляем функцию *getnumbers*, которая для трехзначного числа n дает список двузначных чисел из цифр числа *n*. Определяем цифры трехзначного числа: *a* – первая цифра, количество сотен, *b* – вторая цифра, количество десятков, *c* – третья, последняя цифра. Строим список *h* из цифр данного числа.
`    hh=[]` `    for i in range(3):` `        for j in range(3):` `            if (i!=j)and(h[i]!=0):` `                hh+=[10*h[i]+h[j]]`	Строим список *hh* двузначных чисел из цифр списка *h*, следим, чтобы первая цифра была отлична от нуля, иначе число не будет двузначным.
`    for i in range(len(hh)):` `        for j in range(len(hh)):` `            if (i!=j) and` `               hh[i]==hh[j]):hh[j]=0`	Если в списке *hh* окажутся одинаковые двузначные числа, то одно из этих чисел заменяем на 0.
`hhh=[]` `    for i in range(len(hh)):` `        if (hh[i]!=0)and` `                (isprime(hh[i])):` `            hhh+=[hh[i]]` `    return hhh`	Строим список *hhh* из чисел списка *hh*, которые отличны от нуля и являются простыми, используем для этого функцию *isprime*, описанную выше. Возвращаем список *hhh* как результат функции *getnumbers*.
`nn=0` `maxp=0`	Переменная *nn* служит для хранения трехзначного числа, дающего наибольшего количества простых чисел. Переменная *maxp* служит для хранения наибольшего количества простых чисел.
`for n in range(100,1000):` `    tt=len(getnumbers(n))` `    if tt>=maxp:` `        maxp=tt` `        nn=n`	Перебираем всевозможные трехзначные числа, находим *tt* – количество простых двузначных чисел, составленных из цифр числа n, запоминаем наибольшее количество построенных простых чисел и наиболее результативное число *nn*.
`print(nn,maxp,sorted(getnumbers(nn)))`	Печатаем ответ: 731 6 [13, 17, 31, 37, 71, 73] Программа находит наилучшее трехзначное число 731, оно дает 6 простых чисел и список этих простых чисел.

Тип 5. Сайт7 «Компьютерный ЕГЭ». № 4870 (Уровень: Сложный). Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам.

1. Из цифр, образующих десятичную запись N, строятся наибольшее и наименьшее возможные двузначные числа (числа не могут начинаться с нуля).

2. На экран выводится разность полученных двузначных чисел.

Пример. Дано число N = 351. Наибольшее двузначное число из заданных цифр – 53, наименьшее – 13. На экран выводится разность 53 – 13 = 40.

Чему равно количество чисел N на отрезке [300; 400], в результате обработки которых на экране автомата появится число 20?

Решение[8].

Таблица 7

`def getminmax(n):` `  a=n//100` `  b=(n-a*100)//10` `  c=n%10` `  x=[a,b,c]` `  x.sort()`	Объявляем функцию *getminmax*, на вход функция получает натуральное трехзначное число *n* и возвращает разность максимального и минимального двузначного числа, построенных из цифр числа *n*. Определяем цифры трехзначного числа: *a* – первая цифра, количество сотен, *b* – вторая цифра, количество десятков, *c* – третья, последняя цифра. Строим список *x* из цифр данного числа. Сортируем список по возрастанию.
`  if x[0]==0:` `  minx=x[1]*10+x[0]` `  maxx=x[2]*10+x[1]` `  else:` `  minx=x[0]*10+x[1]` `  maxx=x[2]*10+x[1]` `  return maxx-minx`	Строим минимальное и максимальное двузначное число из цифр списка *x*. Если первый элемент списка x равен 0, то он не может быть первой цифрой минимального числа. Возвращаем разность максимального и минимального двузначного числа, построенных из цифр числа *n*.
`count=0` `for i in range(300,401):` `  if getminmax(i)==20: count+=1`	Начинаем основную программу. Инициируем счетчик count. В цикле просматриваем все числа от 300 до 400, и если максимальное и минимальное число отличается на 20, то увеличиваем счетчик на 1.
`print(count)`	Печатаем ответ. Программа находит ответ: 12.

[7] https://kompege.ru/task

[8] См. Приложение 3.

ТИП 6

Тип 6[9]. Сайт[10] «Компьютерный ЕГЭ». № 6780 (Уровень: Средний[11]).

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, **хвост поднят**. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует 6 команд: **Поднять хвост**, означающая переход к перемещению без рисования; **Опустить хвост**, означающая переход в режим рисования; **Вперёд n** (где n – целое число), вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова; **Назад n** (где n – целое число), вызывающая передвижение в противоположном голове направлении; **Направо m** (где m – целое число), вызывающая изменение направления движения на m градусов по часовой стрелке, **Налево m** (где m – целое число), вызывающая изменение направления движения на m градусов против часовой стрелки.

Запись **Повтори k [Команда1 Команда2 … КомандаS]** означает, что последовательность из S команд повторится k раз.

Черепахе был дан для исполнения следующий алгоритм:

Алгоритм 1

Повтори 2
[Вперёд 10 Направо 90 Вперёд 20 Направо 90]
Поднять хвост
Вперёд 4 Направо 90 Вперёд 3 Налево 90
Опустить хвост
Повтори 2 [Вперёд 70 Направо 90 Вперёд 80 Направо 90]

Определите минимальное расстояние от точки с координатами (0, 0) до следа, оставленного черепахой.

Таблица 8

Решение[12]. `from turtle import *`	Импортируем все команды из библиотеки *turtle*.
`speed(0)` `screensize(3000,3000)`	Выставляем максимальную скорость черепашки. Устанавливаем размер графического окна, где будет работать черепашка.
`lt(90)` `stamp()`	По условию задачи в начале работы черепашка смотрит наверх, а в среде Python по умолчанию черепашка смотрит налево, поэтому поворачиваем чере-

[9] Задачи об исполнителе Черепашка.
[10] https://kompege.ru/task
[11] https://vk.com/pro100ege68
[12] См. Приложение 4.

	пашку на $90°$ налево. Штампуем черепашку, чтобы обозначить начало координат.
`color("red")`	Для координатной сетки устанавливаем красный цвет.
`x1,x2,y1,y2,step=-10,30,-10,30,40`	Вертикальные линии сетки будут прочерчиваться для $x_1 \leq x \leq x_2$. Горизонтальные линии сетки будут прочерчиваться для $y_1 \leq y \leq y_2$. Количество пикселей на единицу чертежа запишем в переменную *step*. Значения всех этих переменных определяются опытным путем для каждой конкретной задачи этого типа.
`for x in range(x1,x2):` `  pu()` `  goto(x*step,y1*step)` `  pd()` `  goto(x*step,y2*step)` `  pu()`	Чертим вертикальные линии сетки. При необходимости поднимаем перо или опускаем перо.
`for y in range(y1,y2):` `  pu()` `  goto(x1*step,y*step)` `  pd()` `  goto(x2*step,y*step)` `  pu()`	Чертим горизонтальные линии сетки. При необходимости поднимаем перо или опускаем перо.
`goto(0,0)` `color("blue")` `width(2)`	Переносим черепашку в начало координат. Для траектории движения черепашки устанавливаем синий цвет и толщину линии 2.
`for i in range(2):` `  fd(10*step)` `  rt(90)` `  fd(20*step)` `  rt(90)` `pu()` `fd(4*step)` `rt(90)` `fd(3*step)` `lt(90)` `pd()` `for i in range(2):` `  fd(70*step)` `  rt(90)` `  fd(80*step)` `  rt(90)`	В соответствии с Алгоритмом 1 программируем движение черепашки.

Результат работы программы приведен на рисунке 1.

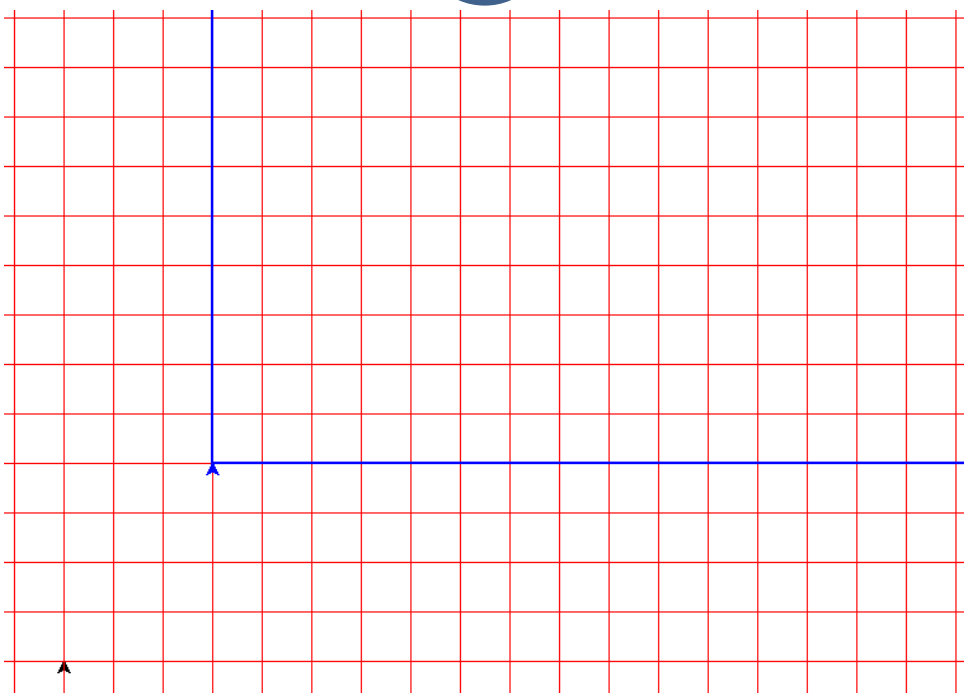

Рисунок 1

Видно, что расстояние от начала координат до траектории движения черепашки можно найти по теореме Пифагора (впрочем, можно использовать и египетский треугольник):

$$d = \sqrt{3^2 + 4^2} = 5$$

Ответ. 5.

Тип 6[13]. Сайт[14] «Компьютерный ЕГЭ». № 5747 (Уровень: Средний) Автор задачи Д. Тараскин[15].

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует шесть команд:

Поднять хвост, означающая переход к перемещению без рисования;

Опустить хвост, означающая переход в режим рисования;

Вперёд n (где n – целое число), вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова;

Назад n (где n – целое число), вызывающая передвижение Черепахи на n единиц в направлении противоположном тому, куда указывает её голова;

Направо m (где m – целое число), вызывающая изменение направления движения на m градусов по часовой стрелке, и

[13] Задачи об исполнителе Черепашка.
[14] https://kompege.ru/task
[15] https://vk.com/bio_kefir

Налево m (где m – целое число), вызывающая изменение направления движения на m градусов против часовой стрелки.

Запись **Повтори k [Команда1 Команда2 … КомандаS]** означает, что последовательность из S команд повторится k раз.

Черепахе был дан для исполнения следующий алгоритм:

Алгоритм 2

Повтори 6 [Повтори 3 [Вперед 7 Направо 120] Направо 60]

Сколько точек с целочисленными координатами находятся внутри получившейся фигуры? Точки на внешних и внутренних линиях учитывать не нужно.

Решение[16].

`from turtle import *`	Импортируем все команды из библиотеки *turtle*.
`speed(0)` `screensize(2000,2000)`	Выставляем максимальную скорость черепашки. Устанавливаем размер графического окна, где будет работать черепашка.
`lt(90)` `stamp()`	По условию задачи в начале работы черепашка смотрит наверх, а в среде Python по умолчанию черепашка смотрит налево, поэтому поворачиваем черепашку на $90°$ налево. Штампуем черепашку, чтобы обозначить начало координат.
`color("red")`	Для координатной сетки устанавливаем красный цвет.
`x1,x2,y1,y2,step=-10,30,-10,30,60`	Вертикальные линии сетки будут прочерчиваться для $x_1 \le x \le x_2$. Горизонтальные линии сетки будут прочерчиваться для $y_1 \le y \le y_2$. Количество пикселей на единицу чертежа запишем в переменную *step*. Значения всех этих переменных определяются опытным путем для каждой конкретной задачи этого типа.
`for x in range(x1,x2):` `  pu()` `  goto(x*step,y1*step)` `  pd()` `  goto(x*step,y2*step)` `  pu()`	Чертим вертикальные линии сетки. При необходимости поднимаем перо или опускаем перо.

[16] См. Приложение 5.

`for y in range(y1,y2):` `  pu()` `  goto(x1*step,y*step)` `  pd()` `  goto(x2*step,y*step)` `  pu()`	Чертим горизонтальные линии сетки. При необходимости поднимаем перо или опускаем перо.
`goto(0,0)` `color("blue")` `width(2)`	Переносим черепашку в начало координат. Для траектории движения черепашки устанавливаем синий цвет и толщину линии 2.
`for i in range(6):` `  for j in range(3):` `  fd(7*step)` `  rt(120)` `  rt(60)`	В соответствии с Алгоритмом 2 программируем движение черепашки.

Результат работы программы приведен на рисунке 2.

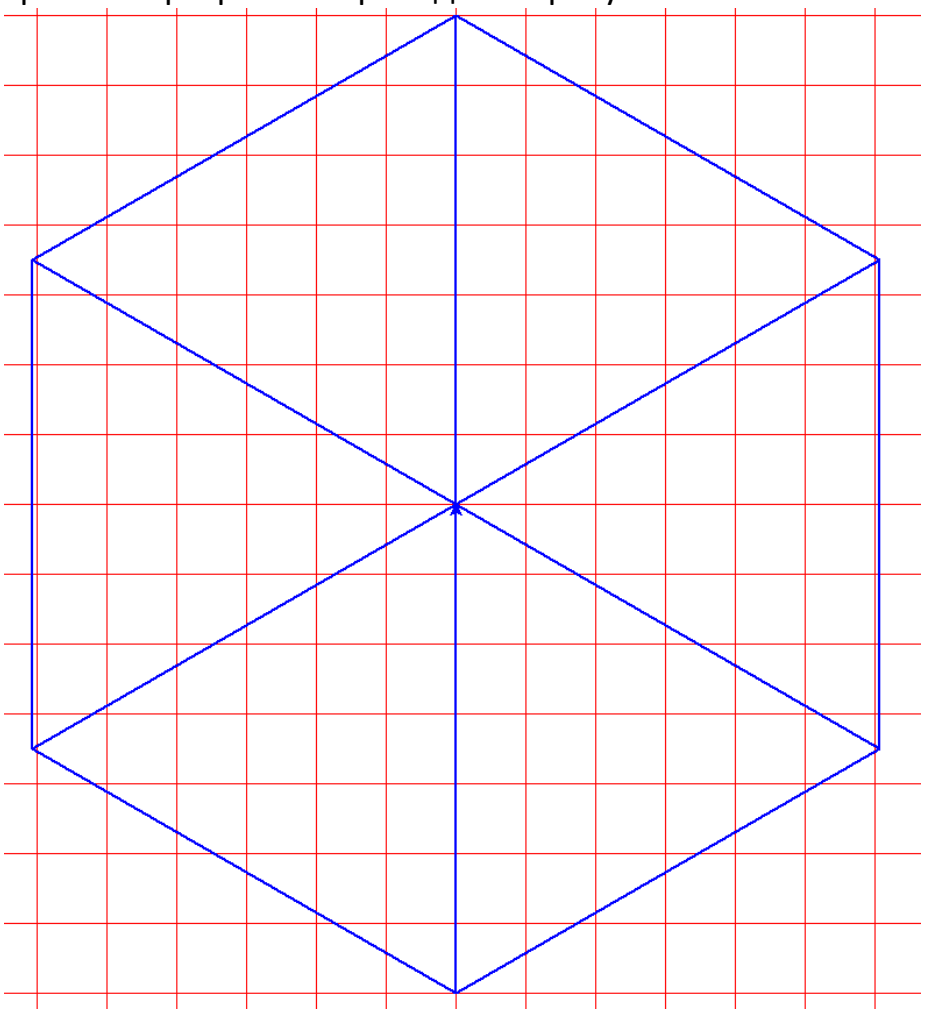

Рисунок 2

Количество точек, удовлетворяющих условию задачи приходится подсчитывать вручную, допустимо в качестве подручного средства для этого использовать графический редактор Paint.

Ответ. 120.

ТИП 8

Тип 8. Сайт[17] «Компьютерный ЕГЭ». № 6591 (Уровень: Средний). Пробник ИМЦ СПб.

Определите количество пятизначных чисел, записанных в семеричной системе счисления, в записи которых:

1. только одна цифра 6;
2. сумма четных цифр числа меньше суммы нечетных цифр числа.

Решение[18]. Задача решается простым перебором.

Таблица 9

count=0 for a in range(1,7): for b in range(7): for c in range(7): for d in range(7): for e in range(7):	Инициируем счетчик *count* пятизначных чисел. В цикле перебираем цифры пятизначного числа. Первая цифра *a* не может быть нулем, поэтому принимает значения от 1 до 6. Остальные цифры от b до e принимают значение от 0 до 6 (в системе счисления с основанием 7 все цифры меньше 6).
c6=0 if a==6: c6+=1 if b==6: c6+=1 if c==6: c6+=1 if d==6: c6+=1 if e==6: c6+=1 if c6==1:	По условию задачи цифра 6 в числе должна встретиться ровно 1 раз. Подсчитываем количество цифр 6 и сохраняем в переменную *c6*. Если цифра 6 встречается ровно один раз, то проверяем следующее условие.
s1=0 s2=0	В переменных *s1* и *s2* будем подсчитывать сумму нечетных цифр и количество четных цифр соответственно. Инициируем эти переменные.
if a%2==0: s2+=a else: s1+=a if b%2==0: s2+=b else: s1+=b if c%2==0: s2+=c else: s1+=c if d%2==0: s2+=d else: s1+=d if e%2==0: s2+=e else: s1+=e	Вычисляем сумму нечетных цифр и сумму четных цифр.

[17] https://kompege.ru/task

[18] См. Приложение 6.

`if s2<s1:` `  count+=1`	Если сумма четных цифр текущего числа меньше суммы нечетных чисел, то загибаем палец, увеличиваем счетчик *count* на 1.
`print(count)`	Печатаем ответ, количество найденных чисел. Программа выдала ответ: 1390.

Тип 8. Сайт[19] «Компьютерный ЕГЭ». № 1363 (Уровень: Сложный). Автор Джобс. Сколько существует четных пятеричных чисел длиной 6, начинающихся с цифры 3?

Решение[20].

Таблица 10

`count=0` `for a in range(5):` `  for b in range(5):` `  for c in range(5):` `  for d in range(5):` `  for e in range(5):`	Инициируем счетчик *count* чисел длины 6. Первая цифра зафиксирована, 3. остальные 5 цифр от a до e перебираем в цикле от 0 до 4 включительно, ведь цифры в пятеричной системе счисления не превосходят 5.
`s=1+a+b+c+d+e`	Остановимся подробно на этой хитрой строке кода. В этом месте участник ЕГЭ может сделать ошибку, подумав, что число четно, если и только если последняя цифра четна. Это не так. Это верно только в системе счисления с четным основанием. Но у нас система счисления пятеричная, в этот системе счисления признак делимости на 2 совсем иной. На самом деле, число в системе счисления с нечетным основанием четно тогда и только тогда, когда сумма цифр четна. В самом деле, рассмотрим число Формула 2 $$\overline{3abcde} = \\ = 3 * 5^5 + a * 5^4 + b * 5^3 + c * 5^2 + \\ + d * 5 + e$$ Рассмотрим сравнимость[21] каждого слагаемого по модулю 2. Формула 3 $$3 * 5^5 \equiv 1 \ (mod\ 2)$$ $$x * 5^n \equiv x \ (mod\ 2), x = a \dots e$$ Поэтому

[19] https://kompege.ru/task

[20] См. Приложение 7.

[21] О сравнимости по модулю можно узнать, посмотрев, скажем, видео от популяризатора математики Бориса Трушина: https://yandex.ru/video/preview/10549136424170246267.

	Формула 4 $\overline{3abcde} \equiv 1 + a + b + c + d + e \ (mod\ 2)$
`if s%2==0:` `  count+=1`	Далее все просто: если текущее число четно, то загибаем палец, счетчик *count* увеличиваем на 1.
`print(count)`	Печатаем ответ. Программа выдает ответ: 1562.

ТИП 12

Тип 12. Сайт[22] «Компьютерный ЕГЭ». № 7002 (Уровень: Средний).

Исполнитель Редактор получает на вход строку цифр и преобразовывает её.

Дана программа для Редактора:

Алгоритм 3

```
НАЧАЛО
ПОКА нашлось(>1) ИЛИ нашлось(>2) ИЛИ нашлось(>0)
  ЕСЛИ нашлось(>1)
    ТО заменить(>1,22>)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось(>2)
    ТО заменить(>2,2>)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось(>0)
    ТО заменить(>0,1>)
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

На вход приведённой выше программе поступает строка, начинающаяся с символа «>», а затем содержащая 40 цифр «0», n цифр «1» и 40 цифр «2».

Определите наименьшее значение n, при котором сумма числовых значений цифр строки, получившейся в результате выполнения программы, является трехзначным числом, состоящем из трех одинаковых цифр.

Решение[23].

Таблица 11

`def value(h):` `  n=len(h)` `  s=0` `  for i in range(n-1):` `  s+=int(h[i])` `  return(s)`	Определим функцию *value*, которая на вход получает строку цифр и с символом «>» в конце и возвращает сумму цифр, кроме, естественно, последнего символа.
`def iswww(h):` `  hh=str(h)`	Определим функцию *iswww*, которая, получив число h и возвращает логическое значение True, если число трехзначное и

[22] https://kompege.ru/task

[23] См. Приложение 8.

	состоит из одинаковых цифр и False в противном случае. Преобразуем число *h* в строку *hh*.
`    n=len(hh)` `    L=(n==3)`	Находим длину строки *hh* и убеждаемся, что у числа h ровно три цифры.
`    for i in range(n-1):` `        if hh[i]!=hh[i+1]:` `            L=False` `    return L`	Сравниваем соседние цифры строки *hh*, и если хотя бы раз цифры не совпадают, то значение переменной *L* устанавливаем False. В качестве результата возвращаем значение логической переменной L.
`for n in range(100):` `    h=">"` `    for i in range(40): h+="0"` `    for i in range(n): h+="1"` `    for i in range(40): h+="2"`	Перебираем различные значения n от 0 до 99. Значение параметра 100 можно менять в сторону увеличения, если программа не найдет нужное n. Заполняем строку в соответствии с заданием.
`    while (">1" in h)` `        or(">2" in h)or(">0" in h):` `        if (">1" in h):` `            h=h.replace(">1","22>")` `        if (">2" in h):` `            h=h.replace(">2","2>")` `        if (">0" in h):` `            h=h.replace(">0","1>")`	Алгоритм 3, данный в задаче на языке Python принимает вид (см. ячейку таблицы слева).
`    if iswww(value(h)):` `        print(n)` `        break`	Проверяем, является ли сумма цифр строки после работы алгоритма 3 трехзначным числом из одинаковых цифр, при этом используем описанные выше функции *iswww* и *value*. Если такое число найдено, то печатаем его и выходим из цикла. Это обеспечит нам нахождение минимального числа n. Программа выдала ответ: 81.

Тип 12. Сайт24 «Компьютерный ЕГЭ». № 4324 (Уровень: Сложный).

Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Приведённую ниже программу применили к входной строке, состоящей из единицы, за которой следуют 105 нулей подряд. В результате работы Редактора получилось некоторое число, записанное в двоичной системе счисления. Определите сумму цифр в шестнадцатеричной записи получившегося числа.

В ответе запишите сумму цифр шестнадцатеричной записи, записанную в десятичной системе счисления.

[24] https://kompege.ru/task

Алгоритм 4

НАЧАЛО
ПОКА нашлось(10)
  ЕСЛИ нашлось(100)
    ТО заменить(100, 1011)
    ИНАЧЕ заменить(10, 11)
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ

Решение[25].

`h="1"+"0"*105`	В соответствии с заданием строим исходную строку.
`while "10" in h:` `  if "100" in h:` `  h=h.replace("100","1011",1)` `  else:` `  h=h.replace("10","11",1)`	Алгоритм 4, данный в задаче на языке Python принимает вид (см. ячейку таблицы слева).
`print(h)`	Печатаем строку, которая получилась после работы алгоритма 4. Видим результат: строка из большого количества единиц.
`k=len(h)` `print(k)`	Видим, что в получившейся строке 158 единиц. Переведем это двоичное число в шестнадцатеричную систему счисления. 158=39*4+2, поэтому полученное двоичное число состоит из 39 групп «1111» в конце и одной группы «11» в начале. Поэтому в шестнадцатеричной системе счисления полученное число имеет вид: 3FFFF…F, одна цифра «3» в начале и 39 цифр F. Поэтому сумма цифр в шестнадцатеричной записи получившегося числа равна Формула 5 $3 + F * 39 = 3 + 15 * 39 = 588$
`print((k//4)*15+3)`	Печатаем ответ: 588.

[25] См. Приложение 9.

ТИП 13

Тип 13. Сайт[26] «Компьютерный ЕГЭ». Пробный 06.2022. Задача 4325.

На рисунке представлена схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З, И, К, Л, М. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город М, проходящих через город Ж, и при этом проходящих через 7 и более городов?

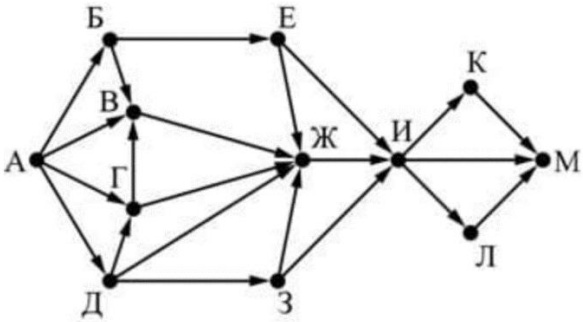

Рисунок 3

Решение[27]. Найти количество всех путей, проходящих через вершину Ж несложно. Но как найти количество длинных путей, проходящих через 7 и более вершин? Может быть сначала найти количество коротких путей, проходящих через 6 и менее городов и потом вычесть это количество из количества всех возможных путей? А Может быть методично выписывать все возможные пути, и выбирать из них нужные маршруты, удовлетворяющих данному в задаче условию. Неужели перебирать руками всевозможные пути? Очень уж это утомительно.

А может быть поручить эту нудную процедуру компьютеру? Очевидно, придется написать рекурсивную процедуру, от которой мы хотим, чтобы она выписывала все возможные пути от начальной вершины к конечной?

Если мы напишем такую волшебную процедуру *F()*, то тогда ее можно будет использовать всяких раз, когда нас попросят найти количество всех путей графа, удовлетворяющих определенному условию…

Чего же мы хотим от нашей процедуры *F()*? Она будет использовать некоторую глобальную переменную x, – список символьных строк – маршрутов.

Сначала в списке x только одна строка – x=[“А”].

После первого прохода процедуры F() список x должен быть таким (см. рисунок 3): x=[“АБ”, ‘АВ”, “АГ”, “АД”].

Затем к списку маршрутов x снова применяем процедуру *F(),* после этого список x должен быть таким: x=[“АБЕ”,“АБВ”, “АВЖ”, “АГВ”,“АГЖ”, “АДГ”, “АДЖ”, “АДЗ”]. Иными словами, для каждого элемента списка x (это срока) надо посмотреть на последний символ и дописать к этой строке новую вершину в соответствии с данным графом (рисунок 3). Таким образом получим новый список (назовем его *xx*) более длинных маршрутов.

Для каждого маршрута (строки) нового списка *xx* будем вызывать процедуру *F()* рекурсивно до тех пор, пока последний символ маршрута не окажется последней вершиной графа.

---

[26] https://kompege.ru/task

[27] См. Приложение 10.

Рассмотрим алгоритм подробнее.

Таблица 12

`def f():` `  global x,y`	Объявляем рекурсивную процедуру *f* без параметров. Для обмена данными будем использовать переменную *x* – список всевозможных маршрутов, с началом в вершине А. Объявляем глобальную переменную *y* – список маршрутов (строк), которые начинаются на «А» и заканчиваются на «М».
`  xx=[]` `  for a in x:` `  ls=a[-1]`	Новый список *xx* пока пустой. Для каждой строки списка *x* получаем последний символ и записываем символ в переменую *ls*[28].
`  if ls=="А":` `  xx+=[a+"Б",a+"В", a+"Г", a+"Д"]` `  if ls=="Б":` `  xx+=[a+"Е",a+"В"]` `  if ls=="В":` `  xx+=[a+"Ж"]` `  if ls=="Г":` `  xx+=[a+"В",a+"Ж"]` `  if ls=="Д":` `  xx+=[a+"Г",a+"Ж",a+"З"]` `  if ls=="Е":` `  xx+=[a+"Ж",a+"И"]` `  if ls=="Ж":` `  xx+=[a+"И"]` `  if ls=="З":` `  xx+=[a+"Ж",a+"И"]` `  if ls=="И":` `  xx+=[a+"К",a+"М",a+"Л"]` `  if ls=="К":` `  xx+=[a+"М"]` `  if ls=="Л":` `  xx+=[a+"М"]`	В соответствии с данным графом (см. Рисунок 3) составляем новый список *xx*. Для другого графа эта часть кода будет меняться: если на графе из города P идут стрелки к городам Q, R, S, то соответствующая строка примет вид: `if ls=="P":` `  xx+=[a+"Q",a+"R", a+"S"]` Этот участок кода надо внимательно проверять, соответствует ли он графу на рисунке 5, ошибку по невнимательности допустить легко.
`  x=xx` `  for a in xx:` `  ls=a[-1]` `  if ls=="М":y+=[a]` `  if ls!="М": f()`	Старый список очищаем, заносим туда новый список более длинных маршрутов. Дополняем список y маршрутов из списка *x*, которые оканчиваются на финальной вершине «М». Если маршрут заканчивается на «М», записываем его в список *y*, таким образом в глобальный переменной *y* будет накапливаться список всех маршрутов, начинающихся в «А» и заканчивающихся в «М». Если же маршрут не заканчивается на букву «М», то к списку *x* снова применяем рекурсивный алгоритм *F()*.

[28] Намёк на Last symbol.

`x=["A"]` `y=[]` `f()`	Начинаем тело программы. Объявляем, с какой вершины графа начинаются все маршруты. Список всех маршрутов от первого до последнего города пока пуст. Вызываем рекурсивную процедуру *F()*.
`z=[]` `for i in range(len(y)):` `  if ("Ж" in y[i])and(len(y[i])>=7):z+=[y[i]]` `print(z,len(z))`	К этому моменту список у всех маршрутов построен. Теперь построим список *z* маршрутов, удовлетворяющих заданному условию: в каждой строке маршрута должна быть буква «Ж» и длина маршрута должна быть равна 7 или более. Печатаем список z таких маршрутов и количество таких маршрутов. Вот маршруты, составленные программой: *['АБЕЖИКМ', 'АБЕЖИЛМ', 'АБВЖИКМ', 'АБВЖИЛМ', 'АГВЖИКМ', 'АГВЖИЛМ', 'АДГВЖИКМ', 'АДГВЖИЛМ', 'АДГВЖИМ', 'АДГЖИКМ', 'АДГЖИЛМ', 'АДЗЖИКМ', 'АДЗЖИЛМ'].* Количество таких маршрутов – 13.

Тип 13. Сайт[29] «Компьютерный ЕГЭ». Задача 4537. Автор задачи Алексей Богданов[30]). На рисунке представлена схема дорог. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город Я, содержащих нечетное количество дорог?

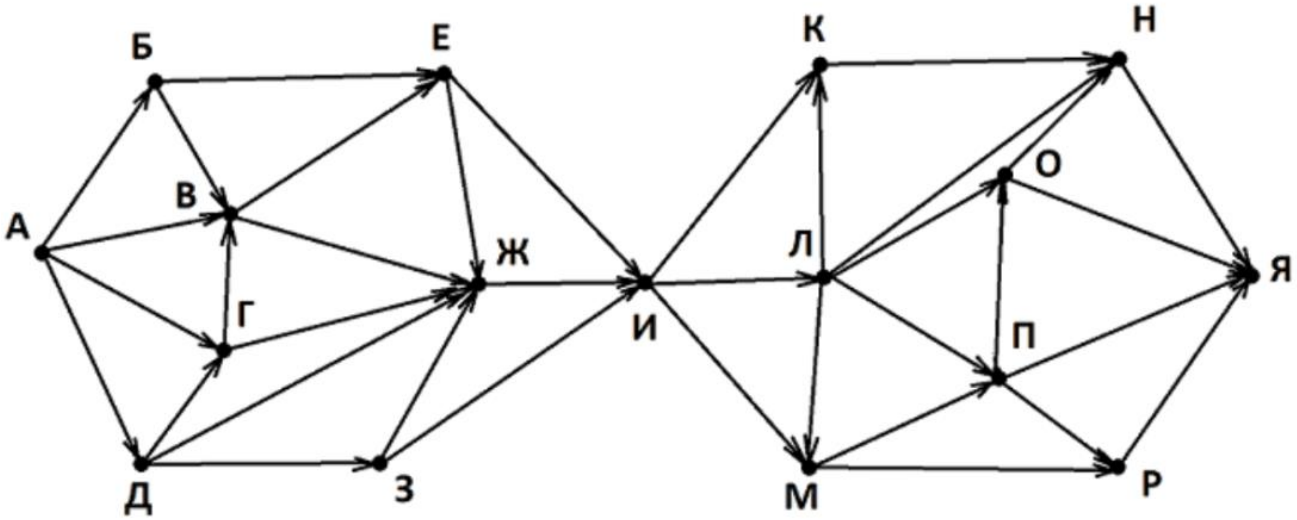

Рисунок 4

[29] https://kompege.ru/task
[30] https://vk.com/danov

Решение[31]. Изменим программу, приведенную таблице 12.

Таблица 13

`def f():` `  global x,y`	Объявляем рекурсивную процедуру *f* без параметров. Для обмена данными будем использовать переменную *x* – список всевозможных маршрутов, с началом в вершине А. Объявляем глобальную переменную *y* – список маршрутов (строк), которые начинаются на «А» и заканчиваются на «М».
`  xx=[]` `  for a in x:` `  ls=a[-1]`	Новый список *xx* пока пустой. Для каждой строки списка *x* получаем последний символ и записываем символ в переменую *ls*[32].
`  if ls=="А":` `  xx+=[a+"Б",a+"В",a+"Г",a+"Д"]` `  if ls=="Б":` `  xx+=[a+"Е",a+"В"]` `  if ls=="В":` `  xx+=[a+"Е",a+"Ж"]` `  if ls=="Г":` `  xx+=[a+"В",a+"Ж"]` `  if ls=="Д":` `  xx+=[a+"Г",a+"Ж",a+"З"]` `  if ls=="Е":` `  xx+=[a+"Ж",a+"И"]` `  if ls=="Ж":` `  xx+=[a+"И"]` `  if ls=="З":` `  xx+=[a+"Ж",a+"И"]` `  if ls=="И":` `  xx+=[a+"К",a+"Л",a+"М"]` `  if ls=="К":` `  xx+=[a+"Н"]` `  if ls=="Л":` `  xx+=[a+"К",a+"Н",a+"О",a+"П",` `a+"М"]` `  if ls=="М":` `  xx+=[a+"П",a+"Р"]` `  if ls=="Н":` `  xx+=[a+"Я"]` `  if ls=="О":` `  xx+=[a+"Н",a+"Я"]` `  if ls=="П":` `  xx+=[a+"О",a+"Я",a+"Р"]` `  if ls=="Р":` `  xx+=[a+"Я"]`	В соответствии с данным графом (см. Рисунок 4) составляем новый список *xx*. Для другого графа эта часть кода будет меняться: если на графе из города P идут стрелки к городам Q, R, S, то соответствующая строка примет вид: `if ls=="P":` `  xx+=[a+"Q",a+"R", a+"S"]` Этот участок кода надо внимательно проверять, соответствует ли он графу на рисунке 4, ошибку по невнимательности допустить легко. Этот участок кода можно заменить на код следующей строки настоящей таблицы.
`  graph=["АБВГД","БЕВ","ВЕЖ",` `"ГВЖ","ДГЖЗ","ЕЖИ","ЖИ","ЗЖИ",` `"ИКЛМ","КН","ЛКНОПМ","МПР",` `"НЯ","ОНЯ","ПОЯР","РЯ"]` `  for a in x:` `  ls=a[-1]` `  for h in graph:` `  if ls==h[0]:` `  for i in range(1,len(h)):`	Код предыдущей строки можно написать более лаконично: по графу (рисунок 4) строим список *graph*, каждый элемент которого – строка, первый символ которой – очередная вершина графа, а все следую-

[31] См. Приложение 11.

[32] Намёк на Last symbol.

`xx+=[a+h[i]]`	щие символы строки – вершины, в которые можно приехать. При этом порядок всех символов кроме первого – не важен.
`x=xx` `for a in xx:` `  ls=a[-1]` `  if ls=="Я":` `  y+=[a]` `  else: f()`	Старый список очищаем, заносим туда новый список более длинных маршрутов. Дополняем список y маршрутов из списка *x*, которые оканчиваются на финальной вершине «М». Если маршрут заканчивается на «М», записываем его в список y, таким образом в глобальный переменной *y* будет накапливаться список всех маршрутов, начинающихся в «А» и заканчивающихся в «М». Если же маршрут не заканчивается на букву «М», то к списку *x* снова применяем рекурсивный алгоритм *F()*.
`x=["A"]` `y=[]` `f()`	Начинаем тело программы. Объявляем, с какой вершины графа начинаются все маршруты. Список всех маршрутов от первого до последнего города пока пуст. Вызываем рекурсивную процедуру *F()*.
`z=[]` `for i in range(len(y)):` `  if (len(y[i])%2==0):z+=[y[i]]` `print(z,len(z))`	К этому моменту список y всех маршрутов построен. Теперь построим список *z* маршрутов, удовлетворяющих заданному условию: количество дорог нечетно, если количество городов в маршруте четно. Печатаем список *z* таких маршрутов и количество таких маршрутов. Количество маршрутов – 180. Маршруты, составленные программой, приведем не полностью, чтобы поберечь нежную психику читателя: *['АБЕЖИКНЯ', 'АБЕЖИЛПОНЯ', 'АБВЕЖИЛМПОНЯ', 'АГВЕЖИЛМПОНЯ', 'АДГВЕЖИЛПОНЯ', 'АДГВЕЖИЛМПОЯ', 'АДГВЕЖИЛМПРЯ', 'АДГВЕЖИМПОНЯ', 'АДГВЕИЛМПОНЯ', 'АДГВЖИЛМПОНЯ', 'АБЕЖИЛМПОЯ', 'АБЕЖИЛМПРЯ', …*

Ту же идею решения можно применить и в случае, если на графе рассматриваются циклические маршруты, когда некоторые города допускается посещать несколько раз. Рассмотрим пример такой задачи.

Тип 13. Сайт[33] «Компьютерный ЕГЭ». Автор А. Игнатюк[34]. Задача 5920. Уровень сложный. На рисунке представлена схема дорог, связывающих города А, Б, В, Г, Д. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Определите количество различных путей ненулевой длины, которые начинаются и заканчиваются в городе А, не содержат этот город в качестве промежуточного пункта и проходят через промежуточные города не более двух раз.

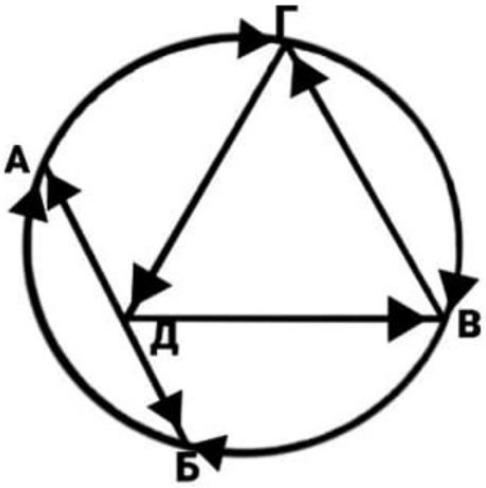

Рисунок 5

Решение[35]. Несколько поправим скрипт.

Таблица 14

`def isgood(h):`	Объявляем функцию *isgood*, которая принимает на вход строку *h* и проверяет, является ли строка хорошей: каждый символ встречается в строке не более чем 2 раза.
`    L=True` `    for i in range(len(h)-1):` `        s=0` `        for j in range(i+1,len(h)):` `            if h[i]==h[j]:s+=1` `        #print(s)` `        if s>1:L=False` `    return L`	Целевая логическая переменная *L* сначала принимает значение True. В цикле для каждого символа кроме последнего подсчитываем количество символов строки *h*, которые совпадают с данным символом. Если один из символов встречается более трех раз, функция возвращает False.
`def f():` `    global x,y`	Объявляем рекурсивную процедуру *f* без параметров. Для обмена данными будем использовать переменную *x* – список всевозможных маршрутов, с началом в вершине А. Объявляем глобальную переменную *y* – список маршрутов (строк), которые начинаются на «А» и заканчиваются на «А».

[33] https://kompege.ru/task
[34] https://vk.com/programmist_ka
[35] См. Приложение 12.

`xx=[]` `for a in x:` `  ls=a[-1]`	Новый список *xx* пока пустой. Для каждой строки списка *x* получаем последний символ и записываем символ в переменую *ls*[36].
`graph=["АГ","БА","ВГБ","ГДВ","ДВАБ"]` `for a in x:` `  ls=a[-1]` `  for h in graph:` `  if ls==h[0]:` `  for i in range(1,len(h)):` `  if isgood(a+h[i]):` `  xx+=[a+h[i]]`	По графу (рисунок 5) строим список *graph*, каждый элемент которого – строка, первый символ которой – очередная вершина графа, а все следующие символы строки – вершины, в которые можно приехать. При этом порядок всех символов кроме первого – не важен. Обратим внимание, что в список *xx* добавляются только «хорошие» маршруты, которые удовлетворяют функции *isgood*, то есть те маршруты, в которых каждый город встречается не более 2 раз.
`x=xx` `for a in xx:` `  ls=a[-1]` `  if (ls=="A"):` `  y+=[a]` `  else: f()`	Старый список очищаем, заносим туда новый список более длинных маршрутов. Дополняем список *y* маршрутов из списка *x*, которые оканчиваются на финальной вершине «А». Если маршрут заканчивается на «А», записываем его в список *y*, таким образом, в глобальный переменной *y* будет накапливаться список всех маршрутов, начинающихся в «А» и заканчивающихся в «А», и каждый город маршрута встречается не более 2 раз. Если же маршрут не заканчивается на букву «А», то к списку *x* снова применяем рекурсивный алгоритм *F()*.
`x,y=["A"],[]` `f()`	Начинаем тело программы. Объявляем, с какой вершины графа начинаются все маршруты. Список всех маршрутов от первого до последнего города пока пуст. Вызываем рекурсивную процедуру *F()*.
`print(y,len(y))`	К этому моменту список *y* всех маршрутов построен. Печатаем список *y* таких маршрутов и количество таких маршрутов. Маршруты, составленные программой: *['АГДВГДВБА', 'АГДВГДБА', 'АГДВГВБА', 'АГВГДВБА', 'АГДВГДА', 'АГВГДБА',*

[36] Намёк на Last symbol.

	'АГВГВБА', 'АГДВБА', 'АГВГДА', 'АГДБА', 'АГВБА', 'АГДА'] *Программой найдено 12 маршрутов.* *Ответ. 12.*

Тип 13. Сайт[37] «Компьютерный ЕГЭ». Задача 379. Уровень сложный. На рисунке представлена схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З, И, К, Л, М. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой.

Сколько существует маршрутов длиной 8 из пункта А в пункт М? Длиной пути считать количество дорог, составляющих этот путь.

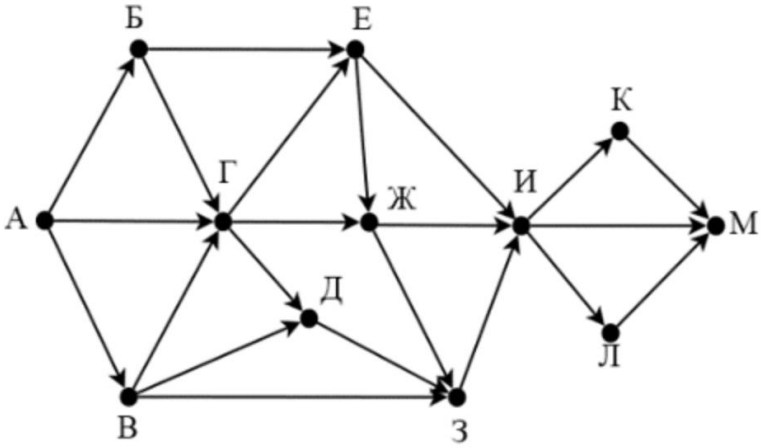

Рисунок 6

Решение[38]. Несколько поправим скрипт.

Таблица 15

`def f():` `  global x,y`	Объявляем рекурсивную процедуру *f* без параметров. Для обмена данными будем использовать переменную *x* – список всевозможных маршрутов, с началом в вершине А. Объявляем глобальную переменную *y* – список маршрутов (строк), которые начинаются на «А» и заканчиваются на «А».
`  xx=[]` `  for a in x:` `  ls=a[-1]`	Новый список *xx* пока пустой. Для каждой строки списка *x* получаем последний символ и записываем символ в переменую *ls*[39].
`graph=["АБВГ","БЕГ","ВГДЗ","ГЕЖД","ДЗ",` `"ЕЖИ","ЖИЗ","ЗИ","ИКЛМ","КМ","ЛМ"]` `  for a in x:` `  ls=a[-1]` `  for h in graph:` `  if ls==h[0]:` `  for i in range(1,len(h)):` `  xx+=[a+h[i]]`	По графу (рисунок 6) строим список *graph*, каждый элемент которого – строка, первый символ которой – очередная вершина графа, а все следующие символы

[37] https://kompege.ru/task

[38] См. Приложение 13.

[39] Намёк на Last symbol.

	строки – вершины, в которые можно приехать. При этом порядок всех символов кроме первого – не важен.
`x=xx` `for a in xx:` `  ls=a[-1]` `  if (ls=="M"):` `  y+=[a]` `  else: f()`	Старый список очищаем, заносим туда новый список более длинных маршрутов. Дополняем список *y* маршрутов из списка *x*, которые оканчиваются на финальной вершине «А». Если маршрут заканчивается на «А», записываем его в список *y*, таким образом, в глобальный переменной *y* будет накапливаться список всех маршрутов, начинающихся в «А» и заканчивающихся в «М». Если же маршрут не заканчивается на букву «М», то к списку *x* снова применяем рекурсивный алгоритм *F()*.
`x,y,z=["A"],[],[]` `f()`	Начинаем тело программы. Объявляем, с какой вершины графа начинаются все маршруты. Список всех маршрутов от первого до последнего города пока пуст. Список *z* предназначен для хранения маршрутов длины 8 (8 дорог в маршруте, 9 городов в маршруте). Вызываем рекурсивную процедуру *F()*.
`for h in y:` `  if len(h)==9:z+=[h]` `print(z,len(z))`	К этому моменту список *y* всех маршрутов построен. Строим список z маршрутов нужной длины. Маршруты, составленные программой: ['АБГЕЖЗИКМ', 'АБГЕЖЗИЛМ', 'АВГЕЖЗИКМ', 'АВГЕЖЗИЛМ'] *Программой найдено 4 маршрута.* *Ответ. 4.*

ТИП 14

Тип 14. Сайт[40] «Компьютерный ЕГЭ». Задача 7702. Уровень сложный. Автор Грачев[41] Н. Дано арифметическое выражение $5xyA_{18} + 18x7_y$. Определите, сколько различных значений может принимать выражение при всех возможных x и y.

Решение. Поскольку x и y – цифры первого слагаемого, а оно в восемнадцатеричной системе счисления, то $x < 18, y < 18$. Поскольку x – цифра второго слагаемого, и

[40] https://kompege.ru/task
[41] https://vk.com/yakapustaa

оно записано в y-чной системе счисления, то $x < y < 18$. И, наконец, поскольку у второго слагаемого есть цифра 8, то основание системы счисления y обязательно больше 8: $y \geq 9$. Итак,

Формула 6

$$x < 18, \quad 9 \leq y < 18, \quad x < y < 18$$

Составим скрипт[42] на Python для решения этой задачи.

Таблица 16

`v=[]` `for y in range(9,18):` `  for x in range(y):` `  v+=[(5*18**3+x*18**2+y*18+10)` `  +(y**3+8*y**2+x*y+7)]`	Составляем список *y* всех чисел, удовлетворяющих условию 6.
`n=len(v)` `v.sort()`	Определяем *n* – количество чисел списка *y* и сортируем его по возрастанию.
`count=0` `for i in range(n-1):` `  if v[i]==v[i+1]:` `  count+=1`	Определяем количество чисел, встречающихся 2 или более раз, записываем его в переменную *count*.
`print(n-count)`	Находим и печатаем количество уникальных чисел списка y. Ответ. 116

Тип 14. Сайт[43] «Компьютерный ЕГЭ». Задача 3759. Уровень сложный. Значение выражения $53^{123} + 65^{2222} - 172^{12}$ записали в системе счисления с основанием 7. Определите количество комбинаций цифр 6# в этой записи, где # – любая цифра от 1 до 5.

Решение[44].

Таблица 17

`def f7(n):` `  h=""` `  while n>0:` `  d=n%7` `  h=str(d)+h` `  n=n//7` `  return h`	Определим функцию *f7*, которая на вход получает натуральное число n и возвращает строку – число, переведенной в семеричную систему счисления.
`n=53**123+65**2222-172**12` `hh=f7(n)`	Вычисляем данное число (заметим, что с такой длинной арифметикой Python справляется на раз, в другим средах будут проблемы). Тут же переводим число в семеричную систему счисления с помощью описанной выше функции *f7* и записываем результат в строку *hh*.
`count=0`	Инициируем счетчик *Count*.

[42] См. Приложение 14.
[43] https://kompege.ru/task
[44] См. Приложение 15.

`for i in range(len(hh)-1):` `  if(hh[i]=="6")and(hh[i+1]in "12345"):` `  count+=1`	Просматриваем строку *hh* от 0 до предпоследнего символа, е если встречается цифра 6, а сразу после нее идет одна из цифр от 1 до 6, то загибаем палец, увеличиваем счетчик *count* на 1.
`print(count)`	Печатаем ответ. Ответ. 478.

ТИП 16

Тип 16. Сайт[45] «Компьютерный ЕГЭ». Задача 6269. Уровень сложный. Автор А. Богданов[46]. Алгоритм вычисления значения функции F(n), где n – целое неотрицательное число, задан следующими соотношениями:

F(n)=n при n<10;

F(n)=F(n//10)+F(n%10), если 10≤n<1000;

F(n)=F(n//1000)−F(n%1000), если n≥1000.

Определите количество значений n, не превышающих 10^6, для которых F(n)=0?

Решение[47].

Таблица 18

`def f(n):` `  if n<10: return(n)` `  if n<1000:` `  return f(n//10)+f(n%10)` `  return f(n//1000)-f(n%1000)`	Определяем рекурсивную функцию f в точности, как она определена в задании.
`count=0` `for n in range(1000000):` `  count += (f(n)==0)`	Инициируем счетчик count. В цикле проверяем все числа, не превышающие 1000000, для которых значение функции равно 0, если находим очередное такое число, то загибаем палец – увеличиваем счетчик count на 1.
`print(count)`	Печатаем ответ. Ответ. 55252.

Возможно и иное решение, более быстрое, если мы обратим внимание на то, что функция возвращает сумму цифр, если число не более чем трехзначное. А если число более чем трехзначное, то функция возвращает разность суммы первых трех цифр и последних трех цифр. Заметим, что функция возвращает 0 если и только если сумма первых трех цифр равна сумме последних трех цифр, иными словами, в задаче просят найти количество всевозможных шестизначных счастливых билетика, и тогда решаем задачу не с помощью рекурсивной функции, но даже без функции.

[45] https://kompege.ru/task

[46] https://vk.com/inf_intensive

[47] См. Приложение 16.

Таблица 19

`# счастливые билетики` `count=0` `for a in range(10):` `  for b in range(10):` `  for c in range(10):` `  for d in range(10):` `  for e in range(10):` `  for f in range(10):`	Инициируем счетчик *count*. Перебираем всевозможные шестизначные билетики от «000000» до «999999».
`  if a+b+c==d+e+f:` `  count+=1`	Если сумма первых трех цифр равна сумме последних трех цифр, то загибаем палец – увеличиваем счетчик count на 1.
`print(count)`	Печатаем ответ: 55252.

ТИП 17

Тип 17. Сайт[48] «Компьютерный ЕГЭ». Задача 7619. Уровень Базовый. Досрочная волна 2023. В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 10 000 включительно. Определите количество пар последовательности, в которых только одно число является двухзначным, а сумма элементов пары кратна максимальному двухзначному элементу последовательности. В ответе запишите количество найденных пар, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Файлы к заданию[49]:17.txt

Решение[50].

Таблица 20

`f=open("17_7619.txt","r")` `h=f.readlines()` `f.close()` `p=[]` `for x in h:` `  p+=[int(x)]`	Открываем для чтения файл *17_7619.txt*, записываем строки из файла в список строк *h*. Создаем пустой список *p*, и записываем в него числа, которые получились от преобразования строк списка *h*. В результате получили список *p* чисел, прочитанных из данного файла.
`maxww=0` `for x in p:` `  if (x<100)and(x>9)and(x>maxww):` `  maxww=x`	В переменной *maxww* ищем наибольшее двузначное число в списке *p*.
`count=0` `mmm=0`	Инициируем счетчик *count* и переменную *mmm*. В переменной *count* будет записано количество найденных пар, удо-

[48] https://kompege.ru/task

[49] https://kompege.ru/files/bLdMiRLC1.txt

[50] См. Приложение 17.

	влетворяющих условию задачи, в переменной *mmm* будет храниться максимальная из сумм элементов таких пар.
`for i in range(len(p)-1):` `  s=0` `  if (p[i]>9)and(p[i]<100):` `  s+=1` `  if (p[i+1]>9)and(p[i+1]<100):` `  s+=1` `  summ=p[i]+p[i+1]` `  if (s==1)and(summ%maxww==0):` `  count+=1` `  if summ>mmm:` `  mmm=summ`	В цикле перебираем всевозможные пары соседних элементов. В переменной *s* находим количество двузначных чисел в этой паре, в переменной *sum* находим сумму элементов этой пары. Если в текущей паре только один элемент является двузначным числом, и сумма элементов этой пары делится нацело на наибольшее двузначное число *maxww,* найденное ранее, то загибаем палец – увеличиваем счетчик count на 1, а если сумма элементов текущей пары больше mmm, то переменную mmm поправляем: заносим туда ту сумму, которая оказалась больше.
`print(count,mmm)`	Печатаем ответ: сначала количество найденных пар, а затем наибольшую сумму элементов таких пар. Программа выдала ответ: 1, 2970.

Тип 17. Сайт[51] «Компьютерный ЕГЭ». Задача 7848. Уровень Сложный. Автор задачи А. Богданов[52]. В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от 10 до 100000 включительно. Определите количество пар последовательности, в которых только одно число состоит из строго возрастающих цифр (например, 247, где 2<4<7), а произведение элементов пары кратно сумме цифр минимального числа из строго убывающих цифр (например, 321, где 3>2>1). В ответе запишите сначала количество найденных пар, затем минимальную из сумм элементов таких пар. Под парой элементов подразумеваются пары соседних элементов. Файлы к заданию:17.txt

Решение[53].

Таблица 21

`def good(n):` `  ld=[]` `  while n>0:` `  d=n%10` `  ld=[d]+ld` `  n=n//10` `  L=True` `  for i in range(len(ld)-1):` `  if ld[i]>=ld[i+1]:` `  L=False`	Определим функцию *good*. На вход функция получает натуральное число *n*, строит список *ld* цифр этого числа, функция возвращает значение *True*, если цифры числа *n* идут строго по возрастанию и *False* в противном случае.

[51] https://kompege.ru/task
[52] https://vk.com/inf_intensive
[53] См. Приложение 18.

`` break return L ```	
`` def good2(n): ld=[] while n>0: d=n%10 ld=[d]+ld n=n//10 L=True for i in range(len(ld)-1): if ld[i]<=ld[i+1]: L=False break return L ```	Определим функцию *good2*. На вход функция получает натуральное число *n*, строит список *ld* цифр этого числа, функция возвращает значение *True*, если цифры числа *n* идут строго по убыванию и *False* в противном случае.
`` def sumd(n): ld=[] while n>0: d=n%10 ld=[d]+ld n=n//10 return sum(ld) ```	Определим функцию *sumd*. На вход функция получает натуральное число *n, строит список цифр ld*[54] и возвращает сумму цифр числа *n* – сумму элементов списка *ld*.
`` f=open("17_7848.txt","r") h=f.readlines() f.close() ```	Читаем данные из файла 17_7848.txt в список строк h. Закрываем файл.
`` p=[] for x in h: p+=[int(x)] ```	Из списка строк *h* получаем список целых чисел *p*.
`` m=10000000 for i in range(len(p)): a=p[i] if good2(a): if a<m:m=a k=sumd(m) ```	В писке p находим минимальное число *m*, в котором цифры идут по убыванию (используем описанную ранее функцию *isgood2*). Находим сумму цифр этого минимального числа *m*, сохраняем ее в переменной *k*.
`` count=0 mm=1000000000 for i in range(len(p)-1): a,b=p[i],p[i+1] s=0 if good(a):s+=1 if good(b):s+=1 if (s==1)and(a*b%k==0): count+=1 s=a+b if s<mm:mm=s ```	Инициируем переменные count (счетчик для подсчета количества пар с нужными свойствами) и mm (для подсчета минимальной суммы элементов таких пар). Перебираем все пары из соседних чисел, у которых ровно один элемент пары состоит из возрастающих цифр, и произведение цифр делится на сумму цифр минимального числа из строго убывающих цифр.
`` print(count,mm) ```	Печатаем ответ: количества пар с нужными свойствами и минимальной суммы элементов таких пар. Ответ. 30, 4138.

[54] List of Digits.

ТИП 19-21

Тип 19-21. Сайт[55] «Компьютерный ЕГЭ». Задача 7850. Уровень Базовый. Автор задачи А. Богданов[56]. Эти три задачи об одной и той же игре, с одинаковыми правилами, только вопросы в задачах разные. Поэтому для трех задач будем писать одну программу на Python .

Тип 19. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может увеличить количество камней в куче на 1, 3 камня или удвоить количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 73. Игрок, первым получивший кучу из не менее 73 камней, считается победителем.

В начальный момент в куче было S камней; $1 \le S \le 72$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника

Укажите такое **минимальное** значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом.

Тип 20. Для игры, описанной в задании 19, найдите **два наименьших** значения S, при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

- Петя не может выиграть за один ход;
- Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Найденные значения запишите в ответ в порядке возрастания.

Тип 21. Для игры, описанной в задании 19, найдите **два наименьших** значения S, при котором одновременно выполняются два условия:

- у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
- у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом;

Найденные значения запишите в ответе в порядке возрастания

Решение[57].

Таблица 22

`def f(a,m):` `  if a>=73: return m%2==0`	В соответствии с правилами игры описываем рекурсивную функцию *f* с параметрами a – количество камней в куче, m – номер хода[58]. Если количество камней в куче достигло 73 или больше, то игра закончилась, функция возвращает True если

[55] https://kompege.ru/task
[56] https://vk.com/inf_intensive
[57] См. Приложение 19.
[58] Намек на слово Move – ход.

	номер хода четный и False в противном случае[59].
`if m==0: return False`	До этой строки исполнитель дойдет, если количество камней в куче не превосходит 72. Если при этом номер хода равен 0, значит больше никто ходить не будет, возвращаем False.
`h=[f(a+1,m-1),f(a+3,m-1),f(a*2,m-1)]` `return any(h) if (m-1)%2==0 else all(h)`	Составляем список всевозможных ходов в соответствии с правилами игры[57].
`print(min(s for s in range(1,72) if f(s,2)and (not(f(s,1)))))`	Формируем ответ 19 задачи. Программа выдала 36. Ответ: 36.
`print([s for s in range(1,73) if f(s,3)and (not(f(s,1)))])`	Формируем ответ 20 задачи. Программа построила список [18, 33, 35]. Наименьшие значения этого списка 18 и 33. Ответ. 18, 33.
`print([s for s in range(1,73) if f(s,4)])`	Формируем ответ 21 задачи. Программа построила список [32, 34, 36]. Наименьшие значения этого списка 32 и 34. Ответ. 32, 34.

Тип 19, 20, 21. Сайт[60] «Компьютерный ЕГЭ». Задача 6272. Уровень Средний. Автор задачи А. Богданов[61].

Тип 19. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить **в меньшую** кучу один или три камня. Изменять количество камней в большей куче не разрешается. Игра завершается, когда количество камней в кучах становится равным. Победителем считается игрок, сделавший последний ход, то есть первым сравнявшим количество камней в двух кучах. Игроки играют рационально, т.е. без ошибок. В начальный момент в первой куче было 13 камней, а во второй – S камней, $1 \le S \le 23$?

Укажите такое **минимальное** значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом.

Тип 20.

[59] Красным цветом отмечена часть кода, которая будет меняться при решении другой задачи.
[60] https://kompege.ru/task
[61] https://vk.com/inf_intensive

Для игры, описанной в задании 19, найдите **два наименьших** значения S, при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

- Петя не может выиграть за один ход;
- Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Найденные значения запишите в ответ в порядке возрастания.

Тип 21.

Для игры, описанной в задании 19, найдите **два значения** S, при котором одновременно выполняются три условия:

- у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
- у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом;
- Ваня может выиграть первым ходом после одного из ходов Пети.

Решение[62].

Таблица 23

`def f(a,b,m,p):` `  if a==b: return m%2==0`	В соответствии с правилами игры описываем рекурсивную функцию *f* с параметрами a и b – количество камней в кучах, m – номер хода[63], p – параметр: если p=0, то игрок может совершать глупые ходы и p=1 если игрок играет разумно. Если количество камней в кучах стало равным, то игра закончилась, функция возвращает True если номер хода четный и False в противном случае[64].
`  if m==0: return False`	До этой строки исполнитель дойдет, если количество камней в куче равное. Если при этом номер хода равен 0, значит больше никто ходить не будет, возвращаем False.

[62] См. Приложение 19. Как и прежде, красным цветом обозначена часть кода, которая будет меняться от задачи к задаче.

[63] Намек на слово Move – ход.

[64] Красным цветом отмечена часть кода, которая будет меняться при решении другой задачи.

`if a<b:` `  h=[f(a+1,b,m-1,p),f(a+3,b,m-1,p)]` `else:` `  h=[f(a,b+1,m-1,p),f(a,b+3,m-1,p)]` `if p==1:` `  return any(h) if (m-1)%2==0 else all(h)` `else:` `  return any(h)`	Составляем список всевозможных ходов в соответствии с правилами игры. Учитываем, разумно или нет ходит текущий игрок.
`print(min(s for s in range(1,32) if f(13,s,2,1)and not(f(13,s,1,1))))`	Формируем ответ 19 задачи. Программа выдала 9. Ответ: 9.
`print([s for s in range(1,32) if f(13,s,3,1)and not(f(13,s,1,1))])`	Формируем ответ 20 задачи. Программа построила список [6, 8, 18, 20]. Наименьшие значения этого списка 6 и 8. Ответ. 6, 8.
`print([s for s in range(1,32) if (f(13,s,4,1)or(f(13,s,2,1)))and (not(f(13,s,2,1)))and(f(13,s,2,0))])`	Формируем ответ 21 задачи. Программа построила список [7, 19]. Ответ. 7, 19.

ТИП 22

Тип 22. Статград 15.12.2022 г.

В компьютерной системе необходимо выполнить некоторое количество вычислительных процессов, которые могут выполняться параллельно или последовательно. Для запуска некоторых процессов необходимы данные, которые получаются как результаты выполнения одного или двух других процессов – поставщиков данных. Независимые процессы (не имеющие поставщиков данных) можно запускать в любой момент времени. Если процесс B (зависимый процесс) получает данные от процесса A (поставщика данных), то процесс B может начать выполнение сразу же после завершения процесса A. Любые процессы, готовые к выполнению, можно запускать параллельно, при этом количество одновременно выполняемых процессов может быть любым, длительность процесса не зависит от других параллельно выполняемых процессов.

В таблице представлены идентификатор (ID) каждого процесса, его длительность и ID поставщиков данных для зависимых процессов.

Определите, какое наибольшее количество процессов может быть завершено за первые 170 мс с момента запуска первого процесса.

К задаче прилагался рабочий лист Excel, содержащий информацию о 98 процессах (рисунок 3 показывает лишь фрагмент рабочего листа).

	A	B	C
1	ID процесса	Время выполнения процесса (мс)	ID поставщиков данных
2	10026	18	0
3	10123	4	0
4	10208	41	10026;10123
5	10218	92	0
6	10251	10	0
7	10285	70	10026;10218
8	10374	41	0
9	10423	44	0
10	10495	2	10218
11	10594	13	0
12	10629	59	0
13	10707	55	10629
14	10746	35	10707
15	10824	25	10123
16	10856	33	10707
17	10954	40	10374;10856
18	10980	91	10824
19	10994	23	10594

Рисунок 7

Прежде всего, сохраним электронную таблицу в текстовом формате с разделителями табуляции (рисунок 8).

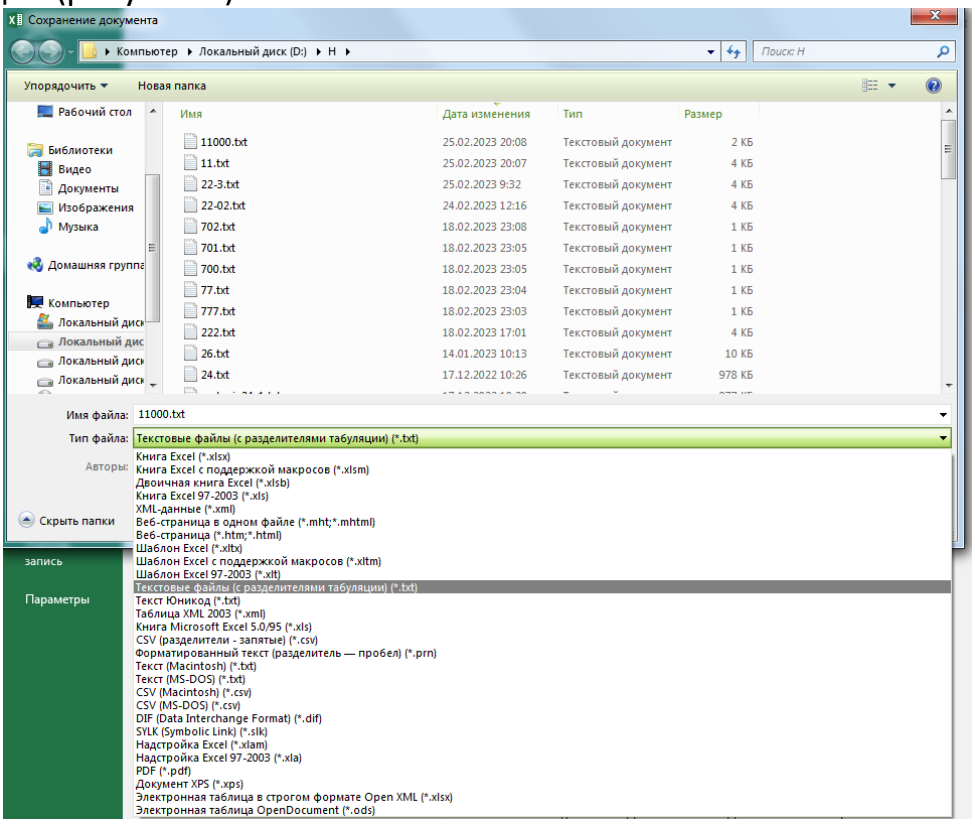

Рисунок 8

И тогда у нас будет текстовый фал, который можно открыть средствами языка программирования Python (рисунок 9).

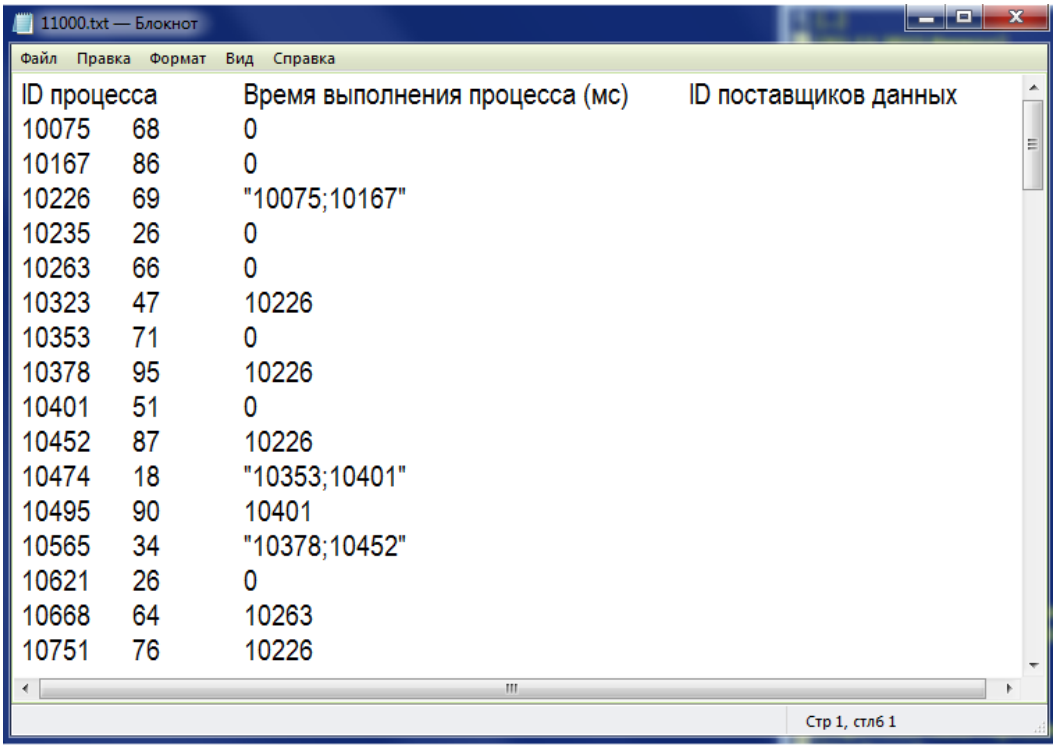

Рисунок 9

Первую строку с заголовками полей можно удалить и так сохранить файл. А можно и не удалять, прочитав первую строку, но проигнорировав ее.

Теперь рассмотрим реализацию алгоритма на языке Python[65].

Таблица 24

`def mintt(pp,tt,ff):` `  res=100` `  for i in range(0,len(tt)):` `if(not(ff[i]))and(pp[i]==[])and(tt[i]<res):` `  res=tt[i]` `  return res`	Функция *mintt* находит наименьший элемент в массиве tt, при этом мы игнорируем процессы, которые закончились или которые не начаты.
`x=[]` `p=[]` `t=[]` `pp=[]` `tt=[]`	Создаем пустые списки *x, p, t, pp, tt.* *Назначения списков:* *x* – список строк, полученных из файла *p* – список для хранения идентификаторов процессов (столбец A в таблице Excel, рис. 4). *t* – список продолжительностей процессов (столбец B в таблице Excel, рис. 4). *pp* – список, аналог столбца *C* таблицы *excel*. *tt* – список продолжительностей процессов (столбец D в таблице Excel, рис. 4).
`f=open('222.txt','r',encoding='utf-16')` `x=f.readlines()` `f.close()`	Заполняем список *x* данными из файла, используя кодировку *utf-16*. Закрываем файл.

[65] См. Приложение 20.

`for i in range(0,len(x)):` `# с 1 начинать, если игнорировать первую строку` `    s = x[i]` `    l = len(s)`	В цикле отрабатываем каждую строку *s* из списка *x*.
`    integ = []` `    k = 0` `    while k < l:` `        s_int = ''` `        a = s[k]` `        while '0' <= a <= '9':` `            s_int += a` `            k+=1` `            if k < l:` `                a = s[k]` `            else:` `                break` `        k+= 1` `        if s_int != '':` `            integ.append(int(s_int))` `    q=int(integ[0])` `    p+=[q]`	Из строки s получаем список чисел *integ*. Нулевой элемент этого списка – идентификатор процесса, записываем его в переменную *q*. Строим список *p* идентификаторов процессов.
`    t+=[int(integ[1])]`	Строим список *t* продолжительностей процессов (столбец B в таблице *Excel*, рис. 4).
`    z=[]` `    for j in range(2,len(integ)):` `        if integ[j]!=0:` `            z+=[integ[j]]`	Строим список *z* идентификаторов поставщиков процессов. Этот список изначально может быть пустым (тогда текущий процесс можно запускать), может состоять из одного или нескольких идентификаторов.
`    pp+=[z]`	Составляем список *pp* – аналог столбца C таблицы *excel*.
`for i in range(0,len(pp)):` `    if (pp[i]==[]):` `        tt+=[t[i]]` `    else:` `        tt+=[1000000]  #списки составлены`	Составляем список *tt* – список продолжительностей процессов (столбец D в таблице Excel, рис. 4). Списки составлены.
`ff=[]` `for i in range(0,len(p)):` `    ff+=[0]` `time=0` `flag=1`	Осуществляем инициализацию списка *ff* –списка флагов, признаков того, что *i*-й процесс (с идентификатором *p[i]*) завершен. Обнуляем счетчик времени. Поднимаем флаг *flag* – признак того, что счетчик времени *time* не достиг 170.
`while flag==1:` `    m=mintt(pp,tt,ff)`	Начинаем в цикле обрабатывать данные. Цикл продолжается, пока счетчик времени *time* не достигнет 170.

	С помощью описанной выше функции *mintt* находит наименьший элемент в массиве *tt*, при этом мы игнорируем процессы, которые закончились или которые не начаты.
`for i in range(0,len(ff)):` `  if (ff[i]==0)and(pp[i]==[]):` `  tt[i]-=m`	У всех активных процессов уменьшаем время на найденное значение *m*.
`for i in range(0,len(ff)):` `  if tt[i]==0:` `  #Завершаем процесс p[i]` `  ff[i]=1`	После этого обязательно найдется хотя бы один процесс (может быть и больше, если прежде минимум m достигался несколько раз), у которого время выполнения стало равно нулю, то есть процесс к этому моменту завершился. Тогда у этого процесса поднимаем флаг *ff[i]* – признак того, что процесс завершился.
`for j in range(0,len(ff)):` `  if (p[i] in pp[j]):` `  pp[j].remove(p[i])` `  if (pp[j]==[]):` `  tt[j]=t[j]` `time+=m`	Поэтому теперь в каждом элементе списка *pp* удаляем идентификатор процесса, который завершен. Если после этого список поставщиков процессов оказывается пустым, то мы можем запускать текущий процесс на выполнение. Прошло m миллисекунд, счетчик времени *time* увеличиваем на *m*.
`if (time>=170):` `  flag=0`	После этого проверяем состояние счетчика времени time, и если время 170 миллисекунд достигнуто, то опускаем флаг flag, тем самым завершая цикл. Остается только…
`s=0` `for i in range(0,len(ff)):` `  s+=ff[i]` `print(s)`	… посчитать, сколько процессов завершено и вывести ответ на печать. Задача решена. Для данного файла исходных данных программа выдает верный ответ 72. Ответ. 72.

Теперь, когда процесс автоматизирован, нам не страшно, если количество процессов и их продолжительность значительно увеличатся. Все равно это не нам считать – будут трудиться написанные нами алгоритмы полностью автоматически.

Здесь автору придется пристыженно признаться, что решение этой задачи на Python приведено здесь, скорее, для красного словца, лишь для увеличения количества

задач, которые на ЕГЭ по информатике можно сделать на Python. На самом деле гораздо быстрее эту задачу решить в электронных таблицах Excel. Приведем здесь и это решение.

Шаг 1. Идентификаторы процессов в столбце С надо разбить на несколько колонок. Выделяем столбец С, на вкладке Данные жмем кнопку Текст по столбцам. Запускается Мастер распределения текста по столбцам (Рисунок 10). Жмем кнопку Далее…

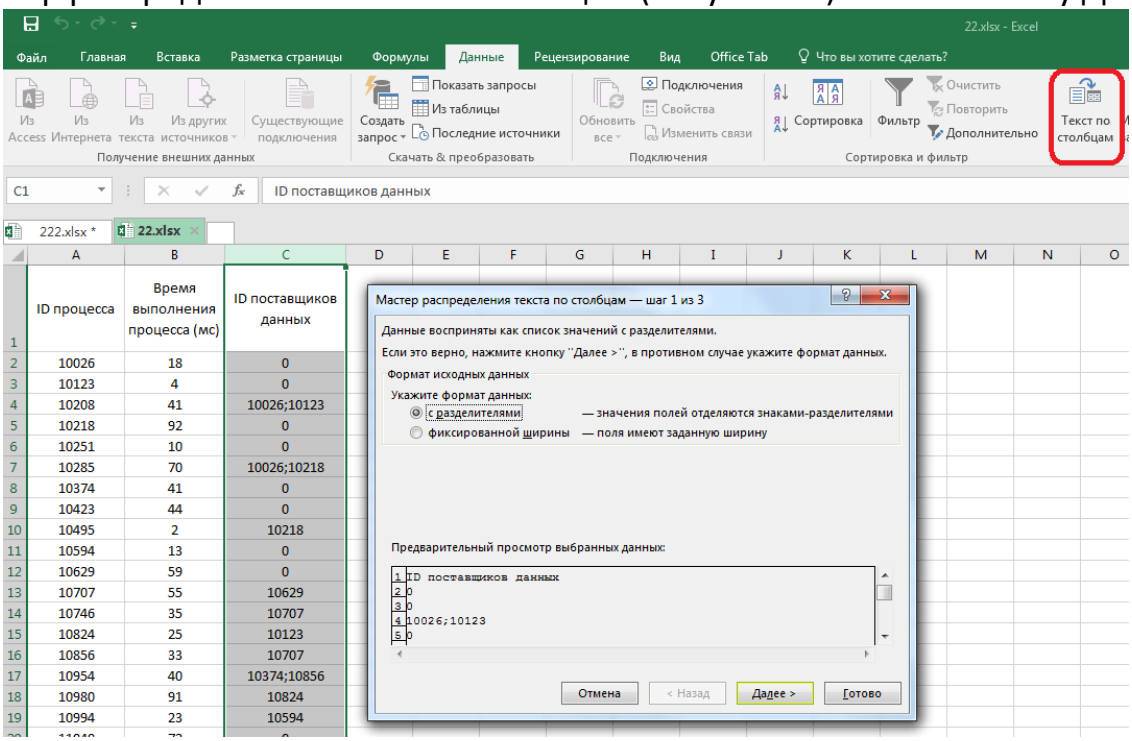

Рисунок 10

Добавляем в Мастере разделитель точка с запятой (Рисунок 11). Жмем кнопку Далее.

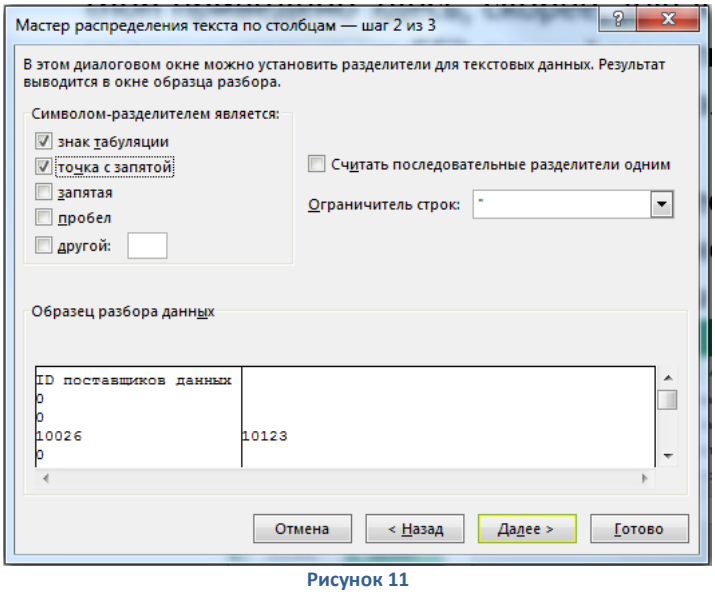

Рисунок 11

В Мастере заполняем поле Поместить в (Рисунок 12). Пишем в этом поле «=$C:$D», поскольку в столбце С не более двух идентификаторов процессов. Но если идентификаторов процессов в столбце С больше, то заполняем это поле соответственно. Жмем кнопку Готово.

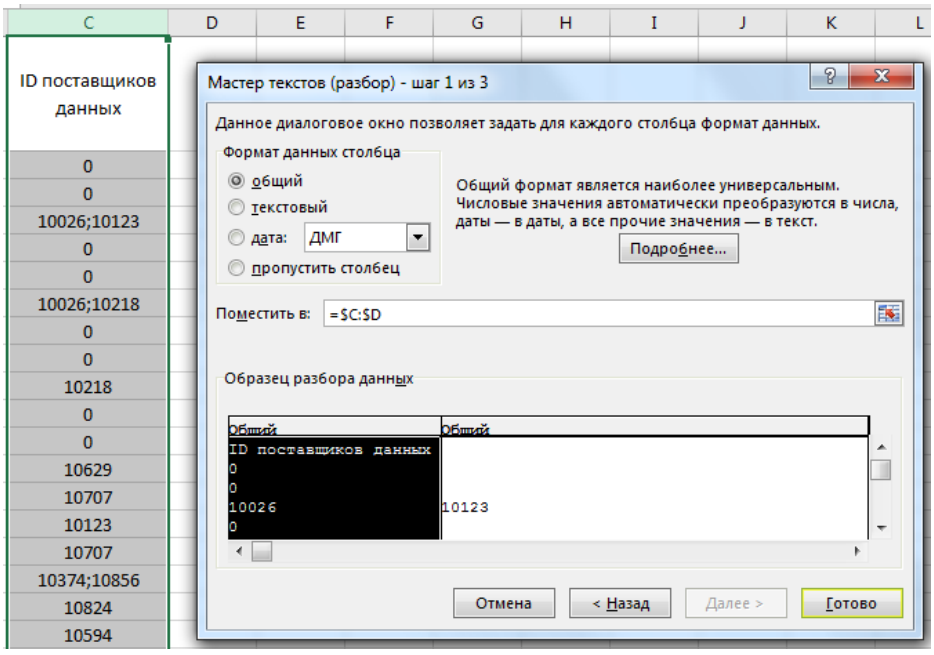

Рисунок 12

В результате столбец С разбился на 2 столбца (или более, в других задачах). Для удобства раскрашиваем эти 2 столбца в желтый цвет, столько же столбцов следующих голубой цвет, и 1 следующий столбец в зеленый цвет ().

	A	B	C	D	E	F	G
1	ID процесса	Время выполнения процесса (мс)	ID поставщиков данных				
2	10026	18	0				
3	10123	4	0				
4	10208	41	10026	10123			
5	10218	92	0				
6	10251	10	0				
7	10285	70	10026	10218			
8	10374	41	0				
9	10423	44	0				
10	10495	2	10218				
11	10594	13	0				
12	10629	59	0				
13	10707	55	10629				
14	10746	35	10707				
15	10824	25	10123				

Рисунок 13

Добавляем одну строку для нулевого процесса, в столбец А пишем идентификатор 0, оставляя остальные ячейки строки пустыми (Рисунок 14).

	A	B	C	D	E	F	G
1	ID процесса	Время выполнения процесса (мс)	ID поставщиков данных				
2	0						
3	10026	18	0				
4	10123	4	0				
5	10208	41	10026	10123			
6	10218	92	0				
7	10251	10	0				
8	10285	70	10026	10218			
9	10374	41	0				
10	10423	44	0				
11	10495	2	10218				
12	10594	13	0				
13	10629	59	0				
14	10707	55	10629				
15	10746	35	10707				

Рисунок 14

Заполняем ячейки таблицы следующим образом (Рисунок 15).

	A	B	C	D	E	F	G	H
1	ID процесса	Время выполнения	ID поставщиков дан					
2	0							
3	10026	18	0		=ВПР(C3;$A:$G;7;0)	=ВПР(D3;$A:$G;7;0)	=МАКС(E3:F3)+B3	=СЧЁТЕСЛИ(G3:G100
4	10123	4	0		=ВПР(C4;$A:$G;7;0)	=ВПР(D4;$A:$G;7;0)	=МАКС(E4:F4)+B4	
5	10208	41	10026	10123	=ВПР(C5;$A:$G;7;0)	=ВПР(D5;$A:$G;7;0)	=МАКС(E5:F5)+B5	
6	10218	92	0		=ВПР(C6;$A:$G;7;0)	=ВПР(D6;$A:$G;7;0)	=МАКС(E6:F6)+B6	
7	10251	10	0		=ВПР(C7;$A:$G;7;0)	=ВПР(D7;$A:$G;7;0)	=МАКС(E7:F7)+B7	
8	10285	70	10026	10218	=ВПР(C8;$A:$G;7;0)	=ВПР(D8;$A:$G;7;0)	=МАКС(E8:F8)+B8	
9	10374	41	0		=ВПР(C9;$A:$G;7;0)	=ВПР(D9;$A:$G;7;0)	=МАКС(E9:F9)+B9	
10	10423	44	0		=ВПР(C10;$A:$G;7;0)	=ВПР(D10;$A:$G;7;0)	=МАКС(E10:F10)+B1(	
11	10495	2	10218		=ВПР(C11;$A:$G;7;0)	=ВПР(D11;$A:$G;7;0)	=МАКС(E11:F11)+B1	
12	10594	13	0		=ВПР(C12;$A:$G;7;0)	=ВПР(D12;$A:$G;7;0)	=МАКС(E12:F12)+B1	
13	10629	59	0		=ВПР(C13;$A:$G;7;0)	=ВПР(D13;$A:$G;7;0)	=МАКС(E13:F13)+B1	
14	10707	55	10629		=ВПР(C14;$A:$G;7;0)	=ВПР(D14;$A:$G;7;0)	=МАКС(E14:F14)+B1	
15	10746	35	10707		=ВПР(C15;$A:$G;7;0)	=ВПР(D15;$A:$G;7;0)	=МАКС(E15:F15)+B1	

Рисунок 15

В ячейке H3 получаем результат (Рисунок 16). При определенных навыках решение задачи занимает всего несколько минут.

	A	B	C	D	E	F	G	H
1	ID процес	Время вы	ID поставщиков данных					
2	0							
3	10026	18	0		0	0	18	72
4	10123	4	0		0	0	4	
5	10208	41	10026	10123	18	4	59	
6	10218	92	0		0	0	92	
7	10251	10	0		0	0	10	
8	10285	70	10026	10218	18	92	162	
9	10374	41	0		0	0	41	
10	10423	44	0		0	0	44	
11	10495	2	10218		92	0	94	
12	10594	13	0		0	0	13	
13	10629	59	0		0	0	59	
14	10707	55	10629		59	0	114	
15	10746	35	10707		114	0	149	
16	10824	25	10123		4	0	29	

Рисунок 16

Ответ. 72.

ТИП 23

Тип 23. Сайт[66] «Компьютерный ЕГЭ». Задача 5552. Уровень Средний.

Исполнитель Лёня преобразует число на экране. У исполнителя есть две команды, которые обозначены латинскими буквами:

А. Прибавь 2

В. Прибавь максимальную цифру текущего числа

Программа для исполнителя – это последовательность команд, которые увеличивают число.

Сколько существует программ, для которых при исходном числе 32 результатом является число 76, и при этом траектория вычислений содержит число 55?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы ВА при исходном числе 81 траектория будет состоять из чисел 89, 91.

Решение[67].

Таблица 25

`def maxdigit(n):` `  max=0;` `  while n>0:` `  d=n%10` `  n=n//10` `  if d>max:max=d` `  return max`	Функция *maxdigit* на вход получает натуральное число и возвращает максимальную цифру в десятичной записи числа.
`def f(s,fin):` `  global count`	Объявляется рекурсивная процедура *f*, на вход она получает переменные *s* – исходное число и *fin* – конечное число[68]. Процедура *f* возвращает в глобальную переменную *count* количество программ, которые преобразуют исходное число *s* в результирующее число *fin*.
`  if s==fin:` `  count+=1`	Если s=fin (добрались до конечного числа), то глобальную переменную count увеличиваем на 1, если s>fin, то мы ничего не делаем, ибо промахнулись мимо.
`  else:` `  if s<fin:` `  f(s+2,fin)` `  f(s+maxdigit(s),fin)`	Если *s<fin*, то пытаемся делать ходы: увеличиваем исходное число на 2 или на максимальную цифру (используем описанную выше функцию *maxdigit*). Обратим внимание на деталь: даже если максимальная цифра числа равна 2, все равно учитываем обе команды, и когда прибавляем 2, и когда

[66] https://kompege.ru/task

[67] См. Приложение 21.

[68] Намек ан французское fin – конец.

	прибавляем максимальную цифру 2, ведь команды имеют разные имена – A и B, и программы получатся разные, их нужно все подсчитать.
count=0 f(32,55) c1=count count=0 f(55,76) c2=count print(c1*c2)	Учитывая сведения из комбинаторики, чтобы найти количество программ, для которых при исходном числе 32 результатом является число 76, и при этом траектория вычислений содержит число 55, надо умножить количество программ приводит 32 к 55 на количество программ, которые приводят 55 к 76. Программа выдает результат 476. Ответ. 476.

Тип 23. Сайт[69] «Компьютерный ЕГЭ». Задача 7852. Уровень Базовый. Автор А. Богданов[70].

Исполнитель Симпли преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1.Прибавить 1

2. Прибавить 2

3. Умножить на 3

Программа для исполнителя Симпли – это последовательность команд. Сколько существует программ, для которых при исходном числе 8 результатом является число 32 и при этом траектория вычислений содержит число 16 и не содержит простые числа?

Решение[71]. Построим решение несколько иначе.

Таблица 26

def isprime(n): k=int(n**0.5) s=0 for i in range(2,k+1): if n%i==0: s+=1 break return s==0	Определим функцию *isprime*, на вход функция получает натуральное число n, а возвращает True, если число n простое и False в противном случае.
def f(a,b): if a>=b: return a==b if isprime(a): return 0	Объявляется рекурсивная функция *f*, на вход она получает переменные *a* – исходное число и *b* – конечное число. Функция *f* возвращает количество программ, которые преобразуют исходное число *a* в результирующее число *b*. При этом если число a простое, то такой путь игнорируется.

[69] https://kompege.ru/task

[70] https://vk.com/inf_intensive

[71] См. Приложение 22.

`h=[f(a+1,b),f(a+2,b),f(a*3,b)]` `  return sum(h)`	Составляем список *h* всех возможных попыток подействовать на исходное число a алгоритмами 1, 2, 3. Возвращаем сумму списка *h*.
`print(f(8,16)*f(16,32))`	Учитывая сведения из комбинаторики, чтобы найти количество программ, для которых при исходном числе 8 результатом является число 32, и при этом траектория вычислений содержит число 16, надо умножить количество программ приводит 8 к 16 на количество программ, которые приводят 16 к 32. Программа выдает результат 40. Ответ. 40.

Тип 23. Сайт[72] «Компьютерный ЕГЭ». Задача 5937. Уровень Сложный. Автор Дмитрий Статный[73].

Исполнитель Калькулятор преобразует число, записанное на экране. У исполнителя есть три команды, которым присвоены номера:

A. Прибавить 2.

B. Прибавить 3.

C. Умножить на 2 и прибавить 1.

Первая команда увеличивает на 2, вторая – увеличивает на 3, третья – умножает на 2 и увеличивает на 1. Сколько существует таких программ, которые исходное число 1 преобразуют в число 55, и при этом траектория содержит не более 15-х чётных чисел.

Решение[74].

Таблица 27

`def f(a,b,k2):` `  if a%2==0:` `  k2+=1` `  if k2>15:return 0` `  if a>=b: return a==b`	Объявляется рекурсивная функция *f*, на вход она получает переменные *a* – исходное число и *b* – конечное число. Переменная k2 служит для подсчета количества четных чисел, содержащихся в траектории Функция *f* возвращает количество программ, которые преобразуют исходное число *a* в результирующее число *b*. При этом если количество четных чисел в траектории окажется более 15, то такой путь игнорируется (возвращается 0).
`  h=[f(a+2,b,k2),f(a+3,b,k2),f(a*2+1,b,k2)]` `  return sum(h)`	Составляем список *h* всех возможных попыток подействовать на исходное число a алгоритмами A, B, C. Возвращаем сумму списка *h*.

[72] https://kompege.ru/task

[73] https://vk.com/azinho

[74] См. Приложение 22.

`print(f(1,55,0))`	Печатаем ответ: количество траекторий с начальным числом 1, конечным числом 55 и начальным количеством четных чисел 0. Программа выдает ответ 4197234. Ответ. 4197234.

Тип 23. Сайт[75] «Компьютерный ЕГЭ». Задача 6274. Уровень Сложный. Автор А. Богданов[76].

У исполнителя Кузнечик есть 4 команды:

1. Прибавить 1

2. Прибавить 3

3. Вычесть 1

4. Вычесть 3

Сколько существует программ, для которых при исходном числе 42 результатом будет являться число 42, при этом траектория вычисления содержит только числа от 40 до 49, притом не более 1 раза, т.е. без повторов.

Решение[77].

Таблица 28

`def f(s):` `  a=s[-1]`	Опишем рекурсивную функцию *f*, которая на вход получает список, в котором сначала содержится только число 42. В переменную a записываем последний элемент списка s.
`  if len(s)>1 and(a==42):` `  return 1`	Если в списке более 1 элемента и последний элемент списка 42 то эту траекторию учитываем, возвращаем 1.
`  if a<40 or a>49 or a in s[:-1]:` `  return 0`	Если траектория вышла за пределы отрезка [40; 49], то игнорируем ее, возвращаем 0.
`  return sum(f(s+[a-h]) for h in [-3,-1,1,3])`	Возвращаем количество найденных программ.
`print(f([42]))`	Программа печатает результат 85. Ответ. 85.

ТИП 24

Тип 24. Сайт[78] «Компьютерный ЕГЭ». Задача 7723. Уровень Базовый. Автор Н. Грачев[79].

Текстовый файл состоит из символов D, R и цифр 1, 8.

Определите максимальное количество идущих подряд троек символов вида **двузначное число + буква** в прилагаемом файле.

Для выполнения этого задания следует написать программу.

---

[75] https://kompege.ru/task
[76] https://vk.com/inf_intensive
[77] См. Приложение 23.
[78] https://kompege.ru/task
[79] https://vk.com/yakapustaa

Файлы к заданию[80]:24.txt.

Начало файла можно увидеть на Рисунке 17.

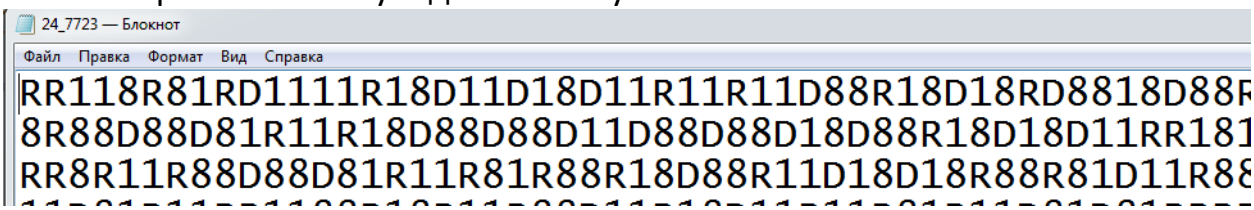

Рисунок 17

Решение[81].

Таблица 29

`def isgood(q):` `  if len(q)<3:return False` `  return (q[0] in "18")` `  and(q[1] in "18")` `  and(q[2] in "RD")`	Опишем функцию *isgood*. На вход функция получает строку q, а возвращает True, если строка имеет вид двузначное число + буква и False в противном случае.
`f=open("24_7723.txt","r")` `h=f.readline()`	Открываем файл для чтения. Читаем содержимой файла в одну длинную строку h.
`n=len(h)` `i=0` `max=0` `t=h[i:i+3]`	В переменной n длина строки h. В переменной t строка из первых трех символов.
`while i<n-3:` `  while not(isgood(t)):` `  i+=1` `  t=h[i:i+3]`	Ищем первую хорошую тройку, используем описанную выше функцию *isgood*.
`  count=0` `  while isgood(t):` `  count+=1` `  i+=3` `  t=h[i:i+3]`	Считаем все хорошие тройки, идущие подряд.
`  if count>max:max=count`	Ищем наибольшее количество идущих подряд троек вида двузначное число + буква.
`print(max)`	Программа выдает результат 67. Ответ. 67.

ТИП 25

Тип 25. Сайт[82] «Компьютерный ЕГЭ». Задача 7825. Уровень Базовый. Автор М. Ишимов[83].

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 123*4?5 соответствуют числа 123405 и 12300405.

[80] https://kompege.ru/files/06ExX0vcK.txt
[81] См. Приложение 24.
[82] https://kompege.ru/task
[83] https://t.me/infkege

Среди натуральных чисел, не превышающих 108, найдите все числа, соответствующие маске 1?0?6*39, делящиеся на 131 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 131.

Количество строк в таблице для ответа избыточно.

Решение[84].

Таблица 30

<table>
<tr><td><pre>
for a in range(10):
    for b in range(10):
        #1?0?639

x=1000000+a*100000+b*1000+639
        if x%131==0:
            print(x,x//131)
</pre></td><td>Перебираем всевозможные числа вида 1?0?639. Если число делится на 131, то печатаем это число и результат деления этого числа на 131 нацело.</td></tr>
<tr><td colspan="2"><pre>
        for c in range(10):

x=10000000+a*1000000+b*10000+6000+c*100+39
</pre></td><td>Перебираем всевозможные числа вида 1?0?6?39.</td></tr>
<tr><td><pre>
            if x%131==0:
                print(x,x//131)
</pre></td><td colspan="2">Если число делится на 131, то печатаем это число и результат деления этого числа на 131 нацело.
Программа выдает результат:
1004639 7669
10056739 76769
10096039 77069
11026139 84169
13056639 99669
14026039 107069
16056539 122569
19056439 145469</td></tr>
</table>

Тип 25. Сайт[85] «Компьютерный ЕГЭ». Задача 3744. Уровень Средний. Автор Джобс.

Найдите все натуральные числа, цифры в которых идут в строго возрастающем порядке, кратные 103. В качестве ответа запишите все найденные числа в порядке возрастания, справа от числа укажите число, умножение которого на 103 дает найденное.

Решение[86].

Таблица 31

<table>
<tr><td><pre>
k=1+123456789//103
for i in range(1,k):
</pre></td><td>Самое большое число, в котором цифры идут по возрастанию это 123456789. Поэтому искомые числа ищем на отрезке [1; k], где
$$k = \left[\frac{123456789}{103}\right] + 1$$
Формула 7</td></tr>
</table>

[84] См. Приложение 25.

[85] https://kompege.ru/task

[86] См. Приложение 26.

`h=str(103*i)` `L=True` `for j in range(len(h)-1):` `  if int(h[j])>=int(h[j+1]):` `  L=False` `  break`	Умножаем проверяемое число i на 103 и проверяем, возрастают ли цифры у этого числа.
`if L: print(h,i)`	Если у числа цифры возрастают, то печатаем это число и проверяемое число. Программа выдала результат: 1236 12 2369 23 2678 26 16789 163

ТИП 26

Тип 26. Сайт[87] «Компьютерный ЕГЭ». Задача 8432. Уровень Средний.

На парковке имеется 70 мест для легковых автомобилей и 30 мест для микроавтобусов. Приезжающий на парковку автомобиль занимает любое свободное место соответствующего типа. При этом если свободных мест для легковых автомобилей нет, то легковой автомобиль занимает свободное место, предназначенное для микроавтобуса, но микроавтобус не может занять место, предназначенное для легкового автомобиля. Если подходящего места нет, автомобиль уезжает.

Входные данные

Первая строка входного файла содержит целое число N – общее количество автомобилей, в течение суток приехавших на парковку. Каждая из следующих N строк описывает один автомобиль и содержит 2 целых числа и букву. Первое число означает время в минутах с начала суток, когда автомобиль прибыл на парковку, второе – необходимую длительность стоянки в минутах. Буква означает тип автомобиля: A – легковой, B – микроавтобус.

Гарантируется, что никакие два автомобиля не приезжают одновременно. Если время прибытия автомобиля совпадает со временем окончания стоянки другого автомобиля, вновь прибывший автомобиль может занять освободившееся место, если оно подходит ему по типу.

В ответе запишите два целых числа: сначала количество микроавтобусов, которые смогут припарковаться, затем – общее количество автомобилей (как легковых, так и микроавтобусов), которые уедут из-за отсутствия мест.

Файлы к заданию[88]: 26.txt (См. Рисунок)

[87] https://kompege.ru/task
[88] https://kompege.ru/files/hpn423kfk.txt

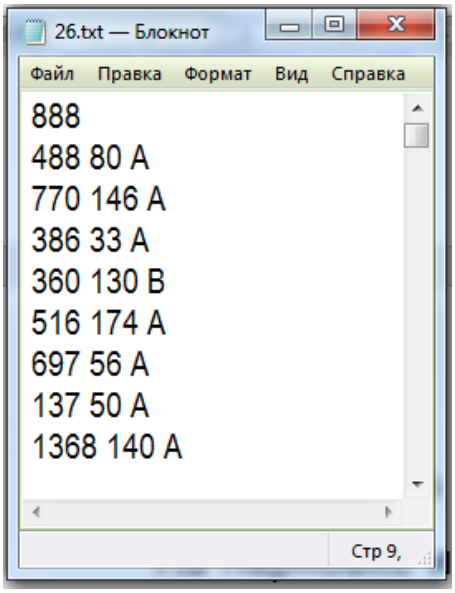

Рисунок 18

Решение[89].

Таблица 32

`def getnums(s):` `  length=len(s)` `  ints=[]` `  i=0` `  while i < length:` `  s_int = ""` `  # строка для нового числа` `  while i<length and s[i] in "0123456789":` `  s_int+=s[i]` `  i+=1` `  i+=1` `  if s_int!="":` `  ints+=[int(s_int)]` `  return ints`	В строке s находятся числа, разделенные пробелом. Функция *getnums* на вход получает эту строку *s* и возвращает список *ints* натуральных чисел.
`f=open("26.txt","r")` `h=f.readlines()` `f.close()` `n=int(h[0])` `c=[]` `cc=[]`	Открываем файл для чтения. Читаем содержимой файла в одну длинную строку *h*. В переменной n количество автомобилей. Инициируем списки c и cc, пока они пустые. Список c – список строк h. Список *cc* – список писков соответствующих чисел.
`for i in range(1,len(h)):` `  c+=[h[i]]` `  cc+=[getnums(h[i])+[h[i][-2]]]` `car=sorted(cc)`	Car – отсортированный по времени прибытия список автомобилей.
`pa=[-1]*70` `pb=[-1]*30` `pripark=0` `out=0`	pa – список стоянок типа A, в списке хранится время, когда стоянка освобождается. –1 означает что парковка изначально свободна. Аналогично, pb – список стоянок типа B. Инициируем счетчики припаркованных микроавтобусов и счетчик автомобилей, которые развернулись и уехали без парковки (обиженные).

[89] См. Приложение 27.

`for i in range(len(car)):` `  tip=car[i][2] #Тип автомобиля`	Приехал автомобиль car[i]. Определяем тип автомобиля.
`  if tip=="B":` `  #` `  k=0` `  while (pb[k]>car[i][0])and(k<29):` `  k+=1` `  if pb[k]<=car[i][0]:` `  # Нашлось свободное место` `  pb[k]=car[i][0]+car[i][1]` `  pripark+=1`	Если приехал микроавтобус, то ищем пустое место. Если место типа B есть, ставим туда микроавтобус, количество припаркованных микроавтобусов увеличиваем на 1
`  else:` `  out+=1`	В противном случае количество обиженных автомобилей увеличиваем на 1.
`  if tip=="A":` `  k=0` `  while (pa[k]>car[i][0])and(k<69): k+=1` `  if pa[k]<=car[i][0]:` `  # Нашлось свободное место` `  pa[k]=car[i][0]+car[i][1]`	Если приехал легковой автомобиль, то ищем сначала для него свободное место на парковке типа A.
`  else:` `  k=0` `  while (pb[k]>car[i][0])and(k<29):` `  k+=1` `  if pb[k]<=car[i][0]:` `  # Нашлось свободное место` `  pb[k]=car[i][0]+car[i][1]`	Если парковка типа A в данный момент занята, то пытаемся припарковаться на стоянке типа B.
`  else:` `  out+=1`	Если места не нашлось, то увеличиваем количество обиженных на 1.
`print(pripark,out)`	Печатаем количество припаркованных микроавтобусов и количество автомобилей, которые уедут из-за отсутствия свободных мест. Программа выдала результат: 168 6. Ответ. 168, 6.

ТИП 27

Тип 27. Сайт[90] «Компьютерный ЕГЭ». Задача 7881. Уровень Средний.

На вход программы поступает последовательность из N натуральных чисел. Рассматриваются все пары различных элементов последовательности (элементы пары не обязательно должны стоять в последовательности рядом, порядок в паре неважен). Необходимо определить количество пар, для которых разность кратна 100, ровно один из элементов пары делится на 37, а номера элементов в последовательности отличаются **не более**, чем на K.

[90] https://kompege.ru/task

Входные данные

Даны два входных файла (A и B), каждый из которых в первой строке содержит число N - количество чисел, во второй строке K – максимальная разница между номерами элементов ($1 \le N \le 10\,000\,000$, $N > K$). В каждой из следующих N строк записаны элементы последовательности (все числа неотрицательные, не превышающие 2 000 000).

В ответе укажите два числа: сначала значение искомой величины для файла A, затем - для файла B.

Типовой пример организации данных во входном файле

10
5
61
274
219
74
26
263
74
274
74
289

Пример выходных данных для приведённого выше примера входных данных:

5

В ответе укажите два числа: сначала значение искомой суммы для файла A, затем для файла B.

Файлы к заданию: 27A.txt[91], 27B.txt[92] (Содержимое файлов типа A и B частично видно на рисунке).

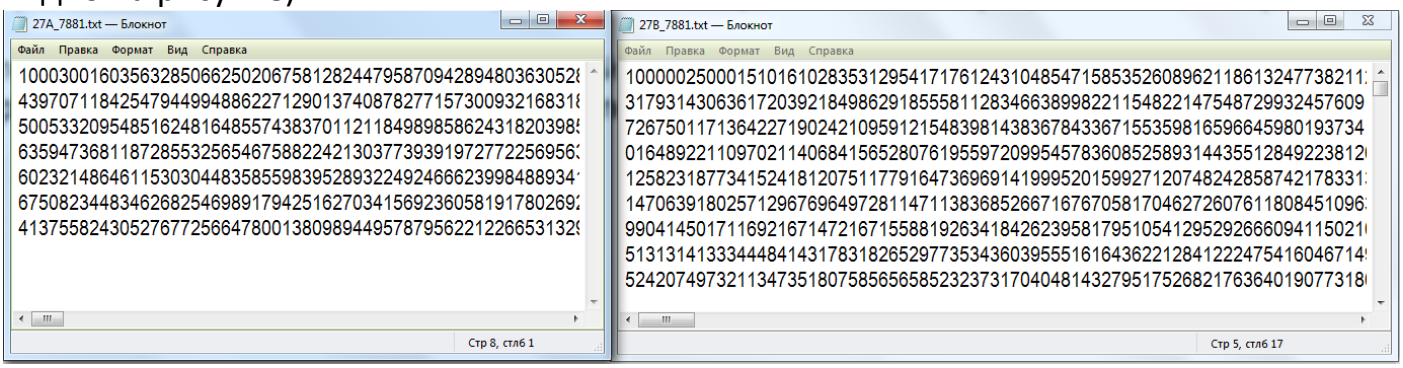

Рисунок 19

Решение[93].

Таблица 33

`f=open("27A_7881.txt","r")` `h=f.readlines()` `f.close`	Открываем файл 27A_7881.txt или 27B_7881.txt. Записываем строки файла в список *h*.

[91] https://kompege.ru/files/gMQIaMJCL.txt
[92] https://kompege.ru/files/KP6u_pO-T.txt
[93] См. Приложение 28.

`hh=[]` `for i in range(len(h)):` `    hh+=[int(h[i])]`	Формируем список *hh* – список чисел, натуральных чисел, прочитанных из файла.
`n,k=hh[0],hh[1]` `x=[]` `count=0`	Получаем *n* – количество чисел и *k* – максимальная разность между номерами элементов. Инициируем счетчик count для подсчета количества нужных пар.
`for i in range(2,len(hh)):x+=[hh[i]]`	Составляем список *x* всех остальных чисел.
`for i in range(len(x)):` `    for j in range(i+1,len(x)):` `        if 0<abs(i-j)<=k:` `            if (x[i]-x[j])%100==0:` `                s=0` `                if x[i]%37==0:s+=1` `                if x[j]%37==0:s+=1` `                if s==1:` `                    count+=1`	Перебираем всевозможные пары элементов, расстояние между которыми не превосходит данное число *k*, такие что разность элементов пары кратно 100. Учитываем только пары, для которых делится на 37 единственный элемент.
`print(count)`	Печатаем ответ. Программа для фала 27A_7881.txt выдает результат 73, а для файла 27B_7881.txt выдает результат 1146648. Ответ. 73, 1146648.

Решение задачи 5424.

```
print("   p1   p2   p3   p4")
for p1 in [False,True]:
    for p2 in [False,True]:
        for p3 in [False,True]:
            for p4 in [False,True]:
                L=(p3<=p1)<=(p4 or not(p2))
                if not(L):
                    print(p1,p2,p3,p4)
```

ПРИЛОЖЕНИЕ 2

Решение задачи 5899.

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

def getnumbers(n):
    a=n//100
    b=(n%100)//10
    c=n%10
    h=[a,b,c]
    hh=[]
    for i in range(3):
        for j in range(3):
            if (i!=j)and(h[i]!=0):
                hh+=[10*h[i]+h[j]]
    for i in range(len(hh)):
        for j in range(len(hh)):
            if (i!=j) and (hh[i]==hh[j]):hh[j]=0
    hhh=[]
    for i in range(len(hh)):
        if (hh[i]!=0)and(isprime(hh[i])):
            hhh+=[hh[i]]
    return hhh

nn=0
maxp=0
for n in range(100,1000):
    tt=len(getnumbers(n))
    if tt>=maxp:
        maxp=tt
        nn=n
print(nn,maxp,sorted(getnumbers(nn)))
```

Решение задачи 4870.

```
def getminmax(n):
    a=n//100
    b=(n-a*100)//10
    c=n%10
    x=[a,b,c]
    x.sort()
    if x[0]==0:
        minx=x[1]*10+x[0]
        maxx=x[2]*10+x[1]
    else:
        minx=x[0]*10+x[1]
        maxx=x[2]*10+x[1]
    return maxx-minx

count=0
for i in range(300,401):
    if getminmax(i)==20: count+=1
print(count)
```

Решение задачи 6780.

```
from turtle import *
speed(0)
screensize(3000,3000)
lt(90)
stamp()
color("red")
x1,x2,y1,y2,step=-10,30,-10,30,40
for x in range(x1,x2):
    pu()
    goto(x*step,y1*step)
    pd()
    goto(x*step,y2*step)
    pu()
for y in range(y1,y2):
    pu()
    goto(x1*step,y*step)
    pd()
    goto(x2*step,y*step)
    pu()
goto(0,0)
color("blue")
width(2)
for i in range(2):
    fd(10*step)
    rt(90)
    fd(20*step)
    rt(90)
pu()
fd(4*step)
rt(90)
fd(3*step)
lt(90)
pd()
for i in range(2):
    fd(70*step)
    rt(90)
    fd(80*step)
    rt(90)
```

Решение задачи 5747.

```
from turtle import *
speed(0)
screensize(2000,2000)
lt(90)
stamp()
color("red")
x1,x2,y1,y2,step=-10,30,-10,30,60
for x in range(x1,x2):
    pu()
    goto(x*step,y1*step)
    pd()
    goto(x*step,y2*step)
    pu()
for y in range(y1,y2):
    pu()
    goto(x1*step,y*step)
    pd()
    goto(x2*step,y*step)
    pu()
goto(0,0)
pd()
color("blue")
for i in range(6):
    for j in range(3):
        fd(7*step)
        rt(120)
    rt(60)
```

Решение задачи 6591

```
count=0
for a in range(1,7):
    for b in range(7):
        for c in range(7):
            for d in range(7):
                for e in range(7):
                    c6=0
                    if a==6: c6+=1
                    if b==6: c6+=1
                    if c==6: c6+=1
                    if d==6: c6+=1
                    if e==6: c6+=1
                    if c6==1:
                        s1=0
                        s2=0
                        if a%2==0: s2+=a
                        else: s1+=a
                        if b%2==0: s2+=b
                        else: s1+=b
                        if c%2==0: s2+=c
                        else: s1+=c
                        if d%2==0: s2+=d
                        else: s1+=d
                        if e%2==0: s2+=e
                        else: s1+=e
                        if s2<s1:
                            count+=1
print(count)
```

Решение задачи 1363.

```
count=0
for a in range(5):
    for b in range(5):
        for c in range(5):
            for d in range(5):
                for e in range(5):
                    s=1+a+b+c+d+e
                    if s%2==0: count+=1
print(count)
```

Решение задачи 7002.

```
def value(h):
    n=len(h)
    s=0
    for i in range(n-1): s+=int(h[i])
    return(s)
def iswww(h):
    hh=str(h)
    n=len(hh)
    L=(n==3)
    for i in range(n-1):
        if hh[i]!=hh[i+1]: L=False
    return L
for n in range(100):
    h=">"
    for i in range(40): h+="0"
    for i in range(n): h+="1"
    for i in range(40): h+="2"
    while (">1" in h)or(">2" in h)or(">0" in h):
        if (">1" in h):
            h=h.replace(">1","22>")
        if (">2" in h):
            h=h.replace(">2","2>")
        if (">0" in h):
            h=h.replace(">0","1>")
    if iswww(value(h)):
        print(n)
        break
```

Решение задачи 4324.

```
h="1"+"0"*105
while "10" in h:
    if "100" in h:
        h=h.replace("100","1011",1)
    else:
        h=h.replace("10","11",1)
print(h)
k=len(h)
print((k//4)*15+3)
```

Решение задачи 4325

```
def f():
    global x,y
    xx=[]
    for a in x:
        ls=a[-1]
        if ls=="А": xx+=[a+"Б",a+"В",a+"Г",a+"Д"]
        if ls=="Б": xx+=[a+"Е",a+"В"]
        if ls=="В": xx+=[a+"Ж"]
        if ls=="Г": xx+=[a+"В",a+"Ж"]
        if ls=="Д": xx+=[a+"Г",a+"Ж",a+"З"]
        if ls=="Е": xx+=[a+"Ж",a+"И"]
        if ls=="Ж": xx+=[a+"И"]
        if ls=="З": xx+=[a+"Ж",a+"И"]
        if ls=="И": xx+=[a+"К",a+"М",a+"Л"]
        if ls=="К": xx+=[a+"М"]
        if ls=="Л": xx+=[a+"М"]
    x=xx
    for a in xx:
        ls=a[-1]
        if ls=="М":
            y+=[a]
        else: f()

x=["А"]
y=[]
f()
z=[]
for i in range(len(y)):
    if ("Ж" in y[i])and(len(y[i])>=7):z+=[y[i]]
print(z,len(z))
```

Решение задачи 4537

```
def f():
    global x,y
    xx=[]

graph=["АБВГД","БЕВ","ВЕЖ","ГВЖ","ДГЖЗ","ЕЖИ","ЖИ","ЗЖИ","ИКЛМ
","КН","ЛКНОПМ","МПР","НЯ","ОНЯ","ПОЯР","РЯ"]
    for a in x:
        ls=a[-1]
        for h in graph:
            if ls==h[0]:
                for i in range(1,len(h)): xx+=[a+h[i]]
    x=xx
    for a in xx:
        ls=a[-1]
        if ls=="Я":
            y+=[a]
        else: f()

x=["А"]
y=[]
f()
z=[]
for i in range(len(y)):
    if (len(y[i])%2==0):z+=[y[i]]
print(z,len(z))
```

Решение задачи 5920.

```
def isgood(h):
    L=True
    for i in range(len(h)-1):
        s=0
        for j in range(i+1,len(h)):
            if h[i]==h[j]:s+=1
        #print(s)
        if s>1:L=False
    return L

def f():
    global x,y
    xx=[]
    graph=["АГ","БА","ВГБ","ГДВ","ДВАБ"]
    for a in x:
        ls=a[-1]
        for h in graph:
            if ls==h[0]:
                for i in range(1,len(h)):
                    if isgood(a+h[i]):
                        xx+=[a+h[i]]
    x=xx
    for a in xx:
        ls=a[-1]
        if (ls=="A"):
            y+=[a]
        else: f()

x,y=["A"],[]
f()
print(y,len(y))
```

Решение задачи 379

```
def f():
    global x,y
    xx=[]
    graph=["АБВГ","БЕГ","ВГДЗ","ГЕЖД","ДЗ","ЕЖИ","ЖИЗ","ЗИ",
"ИКЛМ","КМ","ЛМ"]
    for a in x:
        ls=a[-1]
        for h in graph:
            if ls==h[0]:
                for i in range(1,len(h)): xx+=[a+h[i]]
    x=xx
    for a in xx:
        ls=a[-1]
        if (ls=="М"):
            y+=[a]
        else: f()

x,y,z=["А"],[],[]
f()
for h in y:
    if len(h)==9:z+=[h]
print(z,len(z))
```

Решение задачи 7702.

```
# x<18, y<18 x<y<18, y>=9
v=[]
for y in range(9,18):
    for x in range(y):
        v+=[(5*18**3+x*18**2+y*18+10)+(y**3+8*y**2+x*y+7)]
n=len(v)
v.sort()
count=0
for i in range(n-1):
    if v[i]==v[i+1]:
        count+=1
print(n-count)
```

Решение задачи 3759.

```
def f7(n):
    h=""
    while n>0:
        d=n%7
        h=str(d)+h
        n=n//7
    return h
n=53**123+65**2222-172**12
hh=f7(n)
count=0
for i in range(len(hh)-1):
    if (hh[i]=="6")and(hh[i+1] in "12345"):
        count+=1
print(count)
```

Решение задачи 6269.

```
def f(n):
    if n<10: return(n)
    if n<1000: return f(n//10)+f(n%10)
    return f(n//1000)-f(n%1000)
count=0
for n in range(1000000):
    count += (f(n)==0)
print(count)
```

Альтернативное решение задачи 6269.

```
# счастливые билетики
count=0
for a in range(10):
    for b in range(10):
        for c in range(10):
            for d in range(10):
                for e in range(10):
                    for f in range(10):
                        if a+b+c==d+e+f:
                            count+=1
print(count)
```

Решение задачи 7619.

```
f=open("17_7619.txt","r")
h=f.readlines()
f.close()
p=[]
for x in h:
    p+=[int(x)]
maxww=0
for x in p:
    if (x<100)and(x>9)and(x>maxww):maxww=x
count=0
mmm=0
for i in range(len(p)-1):
    s=0
    if (p[i]>9)and(p[i]<100):s+=1
    if (p[i+1]>9)and(p[i+1]<100):s+=1
    summ=p[i]+p[i+1]
    if (s==1)and(summ%maxww==0):
        count+=1
        if summ>mmm: mmm=summ
print(count,mmm)
```

Решение задачи 7848.

```
def good(n):
    ld=[]
    while n>0:
        d=n%10
        ld=[d]+ld
        n=n//10
    L=True
    for i in range(len(ld)-1):
        if ld[i]>=ld[i+1]:
            L=False
            break
    return L

def good2(n):
    ld=[]
    while n>0:
        d=n%10
        ld=[d]+ld
        n=n//10
    L=True
    for i in range(len(ld)-1):
        if ld[i]<=ld[i+1]:
            L=False
            break
    return L

def sumd(n):
    ld=[]
    while n>0:
        d=n%10
        ld=[d]+ld
        n=n//10
    return sum(ld)

f=open("17_7848.txt","r")
h=f.readlines()
f.close()
p=[]
for x in h:
    p+=[int(x)]
m=10000000
for i in range(len(p)):
    a=p[i]
```

```
    if good2(a):
        if a<m:m=a
k=sumd(m)
count=0
mm=1000000000
for i in range(len(p)-1):
    a,b=p[i],p[i+1]
    s=0
    if good(a):s+=1
    if good(b):s+=1
    if (s==1)and(a*b%k==0):
        count+=1
        s=a+b
        if s<mm:mm=s
print(count,mm)
```

Решение задачи 2304.

```
def f(a,m):
    if a>=73: return m%2==0
    if m==0: return False
    h=[f(a+1,m-1),f(a+3,m-1),f(a*2,m-1)]
    return any(h) if (m-1)%2==0 else all(h)

print(min(s for s in range(1,72) if f(s,2)and (not(f(s,1)))))
print([s for s in range(1,73) if f(s,3)and (not(f(s,1)))])
print([s for s in range(1,73) if f(s,4)])
```

Решение задачи 6272.

```
def f(a,b,m,p):
    if a==b: return m%2==0
    if m==0: return False
    if a<b:
        h=[f(a+1,b,m-1,p),f(a+3,b,m-1,p)]
    else:
        h=[f(a,b+1,m-1,p),f(a,b+3,m-1,p)]
    if p==1:
        return any(h) if (m-1)%2==0 else all(h)
    else:
        return any(h)

print(min(s for s in range(1,32) if f(13,s,2,1)and
not(f(13,s,1,1))))
print([s for s in range(1,32) if f(13,s,3,1)and
not(f(13,s,1,1))])
print([s for s in range(1,32) if
(f(13,s,4,1)or(f(13,s,2,1)))and(not(f(13,s,2,1)))and(f(13,s,2,
0))])
```

Решение задачи 22.

```
def mintt(pp,tt,ff):
    res=100
    for i in range(0,len(tt)):
        if(not(ff[i]))and(pp[i]==[])and(tt[i]<res):
            res=tt[i]
    return res
x=[]
f=open('222.txt','r',encoding='utf-16')
x=f.readlines()
f.close()
p=[]
t=[]
pp=[]
tt=[]
for i in range(0,len(x)):
# с 1 начинать, если игнорировать первую строку
    s = x[i]
    l = len(s)
    integ = []
    k = 0
    while k < l:
        s_int = ''
        a = s[k]
        while '0' <= a <= '9':
            s_int += a
            k+=1
            if k < l:
                a = s[k]
            else:
                break
        k+= 1
        if s_int != '':
            integ.append(int(s_int))
    q=int(integ[0])
    p+=[q]
    t+=[int(integ[1])]
    z=[]
    for j in range(2,len(integ)):
        if integ[j]!=0:
            z+=[integ[j]]
    pp+=[z]
for i in range(0,len(pp)):
    if (pp[i]==[]):
        tt+=[t[i]]
```

```
    else:
        tt+=[1000000]
#списки составлены
ff=[]
for i in range(0,len(p)):
    ff+=[0]
time=0
flag=1
while flag==1:
    m=mintt(pp,tt,ff)
    for  i in range(0,len(ff)):
        if (ff[i]==0)and(pp[i]==[]):
            tt[i]-=m
    for  i in range(0,len(ff)):
        if tt[i]==0:
            #Завершаем процесс p[i]
            ff[i]=1
            for j in range(0,len(ff)):
                if (p[i] in pp[j]):
                    pp[j].remove(p[i])
                    if (pp[j]==[]):
                        tt[j]=t[j]
    time+=m
    if (time>=170):
        flag=0
s=0
for i in range(0,len(ff)):
    s+=ff[i]
print(s)
```

Решение задачи 5552.

```
def maxdigit(n):
    max=0;
    while n>0:
        d=n%10
        n=n//10
        if d>max:max=d
    return max

def f(s,fin):
    global count
    if s==fin:
        count+=1
    else:
        if s<fin:
            f(s+2,fin)
            f(s+maxdigit(s),fin)

count=0
f(32,55)
c1=count
count=0
f(55,76)
c2=count
print(c1*c2)
```

Решение задачи 7852.

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

def f(a,b):
    if a>=b: return a==b
    if isprime(a): return 0
    h=[f(a+1,b),f(a+2,b),f(a*3,b)]
    return sum(h)

print(f(8,16)*f(16,32))
```

Решение задачи 5937.

```
def f(a,b,k2):
    if a%2==0:
        k2+=1
        if k2>15:return 0
    if a>=b: return a==b
    h=[f(a+2,b,k2),f(a+3,b,k2),f(a*2+1,b,k2)]
    return sum(h)

print(f(1,55,0))
```

Решение задачи 6274.

```
def f(s):
    a=s[-1]
    if len(s)>1 and(a==42):return 1
    if a<40 or a>49 or a in s[:-1]: return 0
    return sum(f(s+[a-h]) for h in [-3,-1,1,3])

print(f([42]))
```

Решение задачи 7723.

```
def isgood(q):
    if len(q)<3:return False
    return (q[0] in "18")and(q[1] in "18")and(q[2] in "RD")

f=open("24_7723.txt","r")
h=f.readline()
n=len(h)
i=0
max=0
# i<=n-3
t=h[i:i+3]
while i<n-3:
    #Ищем первую хорошую тройку
    while not(isgood(t)):
        i+=1
        t=h[i:i+3]
    count=0
    #Считаем все хорошие тройки идущие подряд
    while isgood(t):
        count+=1
        i+=3
        t=h[i:i+3]
    if count>max:max=count
print(max)
```

Решение задачи 7825.

```
for a in range(10):
    for b in range(10):
        #1?0?639
        x=1000000+a*100000+b*1000+639
        if x%131==0:
            print(x,x//131)
        for c in range(10):
            x=10000000+a*1000000+b*10000+6000+c*100+39
            if x%131==0:
                print(x,x//131)
```

Решение задачи 3774.

```
k=1+123456789//103
for i in range(1,k):
    h=str(103*i)
    L=True
    for j in range(len(h)-1):
        if int(h[j])>=int(h[j+1]):
            L=False
            break
    if L: print(h,i)
```

Решение задачи

```
def getnums(s):
    length=len(s)
    ints=[]
    i=0
    while i < length:
        s_int = ""  # строка для нового числа
        while i<length and s[i] in "0123456789":
            s_int+=s[i]
            i+=1
        i+=1
        if s_int!="":
            ints+=[int(s_int)]
    return ints

f=open("26.txt","r")
h=f.readlines()
f.close()
n=int(h[0])
c=[]
cc=[]
for i in range(1,len(h)):
    c+=[h[i]]
    cc+=[getnums(h[i])+[h[i][-2]]]
car=sorted(cc)
#Отсортированный по времени прибытия список автомобилей
pa=[-1]*70
pb=[-1]*30
pripark=0
out=0
for i in range(len(car)): #приехал автомобиль car[i]
    tip=car[i][2] #Тип автомобиля
    if tip=="B": # Приехал микроавтобус. ищем пустое место
        k=0
        while (pb[k]>car[i][0])and(k<29): k+=1
        if pb[k]<=car[i][0]: # Нашлось свободное место
            pb[k]=car[i][0]+car[i][1]
            pripark+=1
        else:
            out+=1
    if tip=="A": # приехал легковой
        k=0
        while (pa[k]>car[i][0])and(k<69): k+=1
        if pa[k]<=car[i][0]: # Нашлось свободное место
```

```
            pa[k]=car[i][0]+car[i][1]
        else:
            # Парковка А занята. Пытаемся припарковаться на В
            k=0
            while (pb[k]>car[i][0])and(k<29): k+=1
            if pb[k]<=car[i][0]: # Нашлось свободное место
                pb[k]=car[i][0]+car[i][1]
            else:
                out+=1
print(pripark,out)
```

Решение задачи 7881 (файл типа A).

```
f=open("27A_7881.txt","r")
h=f.readlines()
f.close
hh=[]
for i in range(len(h)):
    hh+=[int(h[i])]
n,k=hh[0],hh[1]
x=[]
count=0
for i in range(2,len(hh)):x+=[hh[i]]
for i in range(len(x)):
    for j in range(i+1,len(x)):
        if 0<abs(i-j)<=k:
            if (x[i]-x[j])%100==0:
                s=0
                if x[i]%37==0:s+=1
                if x[j]%37==0:s+=1
                if s==1:
                    count+=1
print(count)
```

Решение задачи 7881 (файл типа B).

```
f=open("27B_7881.txt","r")
h=f.readlines()
f.close
hh=[]
for i in range(len(h)):
    hh+=[int(h[i])]
n,k=hh[0],hh[1]
x=[]
count=0
for i in range(2,len(hh)):x+=[hh[i]]
for i in range(len(x)):
    for j in range(i+1,min(len(x),i+k+1)):
        if (x[i]-x[j])%100==0:
            s=0
            if x[i]%37==0:s+=1
            if x[j]%37==0:s+=1
            if s==1:
                count+=1
print(count)
```

СПИСОК АЛГОРИТМОВ

СПИСОК ИЛЛЮСТРАЦИЙ

СПИСОК ТАБЛИЦ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ