Тема. Кодирование данных, комбинаторика, системы счисления

Решение задачи 8 из ЕГЭ

1. В русском языке 33 буквы: 10 гласных букв (а, у, о, ы, и, э, я, ю, ё, е), 21 согласная буква (б, в, г, д, ж, з, й, к, л, м, н, п, р, с, т, ф, х, ц, ч, ш, щ) и два знака (ь, ъ).
2. Алфавит английского языка по написанию совпадает с латинским алфавитом и состоит из 26 букв.
3. принципы работы с числами, записанными в позиционных системах счисления
4. если слово состоит из L букв, причем есть n_1 вариантов выбора первой буквы, n_2 вариантов выбора второй буквы и т.д., то число возможных слов вычисляется как произведение

$$N = n_1 \cdot n_2 \cdot \ \ldots \cdot n_L$$

1. если слово состоит из L букв, причем каждая буква может быть выбрана n способами, то число возможных слов вычисляется как $N = n^L$
2. если в программе L вложенных циклов и внешний цикл выполняется n_1 раз, следующий (вложенный) n_2 раз и т.д., то команды самого внутреннего цикла будут выполняться N раз, где

$$N = n_1 \cdot n_2 \cdot \ \ldots \cdot n_L.$$

1. Если $n_1 = n_2 = \ \ldots = n_L = n$, то $N = n^L$.
2. при увеличении n или L значение N сильно возрастает, что приводит к существенному увеличению времени выполнения программы.

пример использования функции **map()** для преобразования строк чисел в обычные числа:

```
b = map(int, ['1', '2', '3', '5', '7'])
```

Первым аргументом указана ссылка на функцию, которая будет последовательно применяться к каждому элементу списка, а вторым аргументом – список из строк с числами. Вообще, вторым аргументом можно записывать любой итерируемый объект. **На выходе функция map() возвращает итератор**. **То есть, переменная b – это итератор, который можно перебрать для преобразования строк в числа.**

Метод join ()

Объединение списка строк с пробелом в качестве разделителя:

```
words = ["Привет", "мир"]
print(" ".join(words)) # Привет мир
```

Объединение списка строк с помощью запятой:

```
fruits = ["яблоко", "банан", "апельсин"]
print(",".join(fruits)) # яблоко,банан,апельсин
```

Метод split () — это **встроенный в Python строковый метод, который позволяет разбить строку на список подстрок на основе указанного разделителя**. Разделитель — это символ или строка, разделяющая отдельные подстроки в исходной строке. По умолчанию метод split () в Python использует пробел в качестве разделителя, что означает, что он будет разбивать строку на подстроки всякий раз, когда встречается пробел, символ табуляции или символ новой строки.

Примеры использования метода split() в Python

Зададим строку my_string, как это показанного ниже. После этого вызовем метод split() для my_string .

```
my_string = "I code for 2 hours everyday"
my_string.split()
# ['I', 'code', 'for', '2', 'hours', 'everyday']
```

при решении с помощью программы на языке Python удобно использовать функции из модуля **itertools**:

combinations – комбинации, например,

```
from itertools import combinations
cmb = list(combinations('ABC', 2))
print( cmb )
// Результат: [('A', 'B'), ('A', 'C'), ('B', 'C')]
```

permutations – перестановки, например,

```
from itertools import permutations
cmb = list(permutations('ABC'))
print( cmb )
// Результат: [('A', 'B', 'C'), ('A', 'C', 'B'),
// ('B', 'A', 'C'), ('B', 'C', 'A'), ('C', 'A', 'B'),
// ('C', 'B', 'A')]
```

product – декартово произведение (все возможные слова заданной длины, составленные из данного алфавита), например:

```
from itertools import product
cmb = list(product('ABC',repeat=2))
print( cmb )
// Результат: [('A', 'A'), ('A', 'B'), ('A', 'C'), ('B', 'A'),
// ('B', 'B'), ('B', 'C'), ('C', 'A'), ('C', 'B'), ('C', 'C')]
```

Как видно из этих примеров, результат работы этих трёх функций – массив кортежей.

В нём удобно работать с отдельными символами, но неудобно искать сочетания букв. Если нужно работать с сочетаниями букв, нужно «склеить» символы каждого кортежа в строки с **помощью метода .join:**

```
from itertools import product
cmb = product('ABC',repeat=2)
cmb = list( map( "".join, cmb ) )
print( cmb )
// Результат: ['AA', 'AB', 'AC', 'BA', 'BB', 'BC', 'CA',
// 'CB', 'CC']
```

или так:

```
from itertools import product
cmb = product('ABC',repeat=2)
for p in cmb:
  s = "".join(p)
  print( s )
// Результат: AA AB AC BA BB BC CA CB CC
```

Пример 1.

Игорь составляет таблицу кодовых слов для передачи сообщений, каждому

сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует трёхбуквенные слова, в которых могут быть только буквы Ш, К, О, Л, А, причём буква К появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

Решение (теоретическое):

1) буква К может стоять на одном из трёх мест, остальные две буквы выбираются из оставшихся четырёх: Ш, О, Л или А
2) пусть К – первая буква, тогда оставшиеся две буквы можно выбрать $4^2 = 16$ способами
3) так как К может стоять на одной из трёх позиций, общее количество подходящих слов – $3 \cdot 16 = 48$
4) Ответ: 48.

Пример 1.

Игорь составляет таблицу кодовых слов для передачи сообщений, каждому

сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует трёхбуквенные слова, в которых могут быть только буквы Ш, К, О, Л, А, причём буква К появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

Способ 2. Программный 1.

1) для построения множества всевозможных слов можно использовать функцию **`product`** из модуля **`itertools`**; затем остаётся выбрать и пересчитать подходящие слова:

```
from itertools import product
p = product('ШКОЛА', repeat=3)
n = 0
for x in p:
  if x.count('К') == 1:
    n += 1
print(n)
```

1) Ответ: 48.

Пример 1.

Игорь составляет таблицу кодовых слов для передачи сообщений, каждому

сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует трёхбуквенные слова, в которых могут быть только буквы Ш, К, О, Л, А, причём буква К появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

Способ 2. Программный 2.

Более компактная запись программы с генератором списка

```
from itertools import product
p=[x for x in product('ШКОЛА',repeat=3)
    if x.count('К')==1 ]
print(len(p))
```

Ответ: 48.

Пример 1.

Игорь составляет таблицу кодовых слов для передачи сообщений, каждому

сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует трёхбуквенные слова, в которых могут быть только буквы Ш, К, О, Л, А, причём буква К появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

Способ 2. Программный 3.

Решение перебором всех возможных комбинаций символов (a, b, c) и подсчет среди них комбинаций, где 'к' встречается только один раз.

```
n=0
s='школа'
for a in s:
  for b in s:
    for c in s:
        if (a+b+c).count('к')==1:
          n+=1
print(n)
```

Ответ: 48.

Задача 1. Герасим составляет 7-буквенные коды из букв Г, Е, Р, А, С, И, М. Каждую букву нужно использовать ровно 1 раз, при этом нельзя ставить подряд две гласные или две согласные. Сколько различных кодов может составить Герасим?

Решение.

Заметим, что поскольку гласных букв три, а согласных четыре, слово должно начинаться с согласной, поскольку иначе нельзя будет составить слово, в котором не будут встречаться подряд две согласные буквы. Также учтём то, что каждую букву нужно использовать только один раз.

На первое место можно поставить любую из четырех согласных. На второе — любую из трех гласных. На третье — любую из трех оставшихся согласных. На четвертое — любую из двух оставшихся гласных. На пятое — любую из двух оставшихся согласных. На шестое — одну оставшуюся гласную. На седьмое место можно поставить последнюю оставшуюся согласную. По правилу произведения, соответствующие количества способов перемножаем. Таким образом, Герасим может составить $4 \cdot 3 \cdot 3 \cdot 2 \cdot 2 \cdot 1 \cdot 1 = 144$ различных кода.

Ответ: 144.

Задача 1. Решение программой

```
import itertools
a = "ГЕРАСИМ"
con = 'ГРСМ'
vol = 'ЕАИ'
ar = itertools.permutations(a) #Перестановка
arl = [ ]
for i in ar:
    arl.append(list(i))
count = 0
for e in arl:
    flag = True
    for i in range(len(e) - 1):
        if (e[i] in con and e[i + 1] in con) or (e[i] in vol and e[i + 1] in vol):
            flag = False
    if flag == True: count += 1
print(count)
```

Задача 2. Все 5-буквенные слова, составленные из букв А, О, У, записаны в обратном алфавитном порядке. Вот начало списка:

1. УУУУУ
2. УУУУО
3. УУУУА
4. УУУОУ

……

Запишите слово, которое стоит на 240-м месте от начала списка.

Решение.

Всего из трёх букв можно составить $3^5 = 243$ слова. Выпишем обратный порядок букв У, О, А. Очевидно, что последнее слово ААААА. Тогда слово с номером 242 запишется как ААААО, 241 — ААААУ, 240 — АААОА.

Ответ: АААОА.

Задача 2.

```
a = {0: "У", 1: "О", 2: "А"}
k = 0
for i in range(0, len(a)):
    for j in range(0, len(a)):
        for g in range(0, len(a)):
            for m in range(0, len(a)):
                for n in range(0, len(a)):
                    k += 1
                    if k == 240:
                        print(a[i], a[j], a[g], a[m], a[n], end=" ")
```

Задача 3. Составляют 5-буквенные слова из букв слова ПЯТНИЦА. Найти количество слов, которые не начинаются с Н и в которых есть только одна буква Я. Буквы могут в слове могут повторяться.

Решение.

Приведём решение на языке Python.

```
from itertools import product
count = 0
for p in product("ПЯТНИЦА", repeat=5):
    if p.count("Я") == 1 and p[0]!="Н":
        count+=1
print(count)
```

Задача 3. Составляют 5-буквенные слова из букв слова ПЯТНИЦА. Найти количество слов, которые не начинаются с Н и в которых есть только одна буква Я. Буквы могут в слове могут повторяться.

Решение.

Приведём решение на языке Python.

```
a = {0: "П", 1: "Я", 2: "Т", 3: "Н", 4: "И", 5: "Ц", 6: "А"}
count = 0
for i in range(0, len(a)):
    for j in range(0, len(a)):
        for g in range(0, len(a)):
            for m in range(0, len(a)):
                for n in range(0, len(a)):
                    s = a[i] + a[j] + a[g] + a[m] + a[n]
                    if a[i] != 'Н' and s.count('Я') == 1:
                        count += 1
print(count)
```

Задача 4. Все шестибуквенные слова, составленные из букв М, А, Н, Г, У, С, Т, записаны в алфавитном порядке и пронумерованы.
Вот начало начало списка:

1. АААААА
2. АААААГ
3. АААААМ
4. АААААН
5. АААААС
6. АААААТ
7. АААААУ

...

Под каким номером в списке стоит последнее слово, которое не начинается с буквы У, содержит только две буквы М и не более одной буквы Г.

Задача 4. Все шестибуквенные слова, составленные из букв М, А, Н, Г, У, С, Т, записаны в алфавитном порядке и пронумерованы.
Вот начало начало списка:

1. ААААА
2. АААААГ
3. АААААМ
4. АААААН
5. АААААС
6. АААААТ
7. АААААУ

...

Под каким номером в списке стоит последнее слово, которое не начинается с буквы У, содержит только две буквы М и не более одной буквы Г.

```
from itertools import product
count = 0
m=[]
for p in product(sorted("МАНГУСТ"), repeat=6):
    count+=1
    if p.count("М") == 2 and p[0]!="У" and p.count("Г") <=1:
        m.append(count)
print(max(m))
```

Шифр кодового замка представляет собой последовательность из пяти символов, каждый из которых является цифрой от 1 до 4. Сколько различных вариантов шифра можно задать, если известно, что цифра 1 встречается ровно два раза, а каждая из других допустимых цифр может встречаться в шифре любое количество раз или не встречаться совсем?

Решение.

Количество способов поставить две 1 на пять позиций — $5 \cdot \frac{4}{2} = 10.$

После того, как определили позиции двух 1, на оставшиеся позиции можем поставить любое из трёх чисел, это можно сделать $3^3 = 27$ способами.

Итого всего $10 \cdot 27 = 270$ кодов.

Ответ: 270.

```
from itertools import product
alphabet = '1234'
ap=[ ]
for i in product(alphabet, repeat=5):
    if i.count('1') == 2:
        ap.append(i)
print(len(ap))
```

Задача 6. Игорь составляет таблицу кодовых слов для передачи сообщений, каждому сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует 5-буквенные слова, в которых есть только буквы A, B, C, X, причём буква X появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

Решение.

Пусть X стоит в слове на первом месте. Тогда на каждое из оставшихся 4 мест можно поставить независимо одну из 3 букв. То есть всего $3 \cdot 3 \cdot 3 \cdot 3 = 81$ вариант.

Таким образом, X можно по очереди поставить на все 5 мест, в каждом случае получая 81 вариант.

Итого получается $81 \cdot 5 = 405$ слов.

Ответ: 405.

```
import itertools
a = "ABCX"
ar = itertools.product(a, repeat=5) #Размещение с повторением
arl = []
for i in ar:
    arl.append(list(i))
count = 0
for e in arl:
    if e.count('X') == 1:
        count += 1
print(count)
```