Задание №13, IP-адресация

Теория

Байт — восьмибитное (=восьмиразрядное) двоичное число, принимающее значения от 0 до 255 в десятичном виде, или от 8 нулей до 8 единиц в двоичном. Все IP-адреса и маски состоят из 4-ёх байтов.

Узел сети — конкретное устройство, соединённое с другими устройствами как часть компьютерной сети.

IP-адрес — уникальный сетевой адрес узла в компьютерной сети. Состоит из четырёх байтов. IP-адрес в двоичной записи состоит из двух частей: сначала адрес сети (одинаковая часть для всех IP в одной сети), а за ним адрес узла (для каждого IP уникальная).

IP-адреса можно записывать как в двоичном, так и в десятичном виде:

IP:

172 . 16 . 157 . 240

‖ ‖ ‖ ‖

10101100.00010000.10011101.11110000

Маска подсети — определяет какая часть IP-адреса является адресом сети (1 в разряде), а какая – адресом узла (0 в разряде). В начале всегда идут только единицы, а после нули.

Маску тоже можно записать несколькими способами: как количество единиц (0–32), двоичный вид, десятичный.

Маска:
количество единиц: 19 =
двоичный вид: 11111111.11111111.11100000.00000000 =
десятичный вид: 255.255.224.0

Адрес сети — результат побитовой конъюнкции между маской и IP-адресом. Где у маски 1, там остаётся значение узла и там фиксированная часть для всех адресов узла в этой сети. Где у маски 0, там у адреса сети тоже 0, а у адреса узла может быть любое двоичное число в разряде.

Широковещательный адрес — специализированный адрес, в котором в узловой части стоят единицы.

Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств, но учитываются при подсчёте IP-адресов в сети.

Маска:

11111111.11111111.11100000.00000000

IP-адрес узла:

10101100.00010000.10011101.11110000

сетевая часть (фиксированная) узловая часть (любые комбинации 1 и 0)

Адрес сети:

10101100.00010000.10000000.00000000

Широковещательный адрес:

10101100.00010000.10011111.11111111

Решение кодом

Условия в этой задаче могут быть максимально разнообразные, поэтому нет единого шаблона решения, но абсолютно все задачи связаны с применением маски к заданному IP-адресу. В Python можно получить список всех возможных IP-адресов в сети, заданной IP-адресом и маской, с помощью библиотеки **ipaddress** и команды **ip_network.**

```
from ipaddress import *
ip_address = '192.16.224.228'
mask = '255.255.255.0' # или просто '24' - количество единиц
# второй аргумент 0 - указатель, что ip_address может быть не адресом сети
net = ip_network(f'{ip_address}/{mask}', 0)
for ip in net:
    b_ip = bin(int(ip))[2:] # двоичный вид IP, могут отсуствовать незначащие
    while len(b_ip) < 32: # длина должна быть ровно 32
        b_ip = '0' + b_ip # добавляем незнач. нули, если они влияют на реше

    # Выполняем условие задачи...

# доп. свойства у сети:
```

```
net.network_address # адрес сети
net.netmask # маска
net.broadcast_address # широковещательный адрес
```

Алгоритм решения

6B2DF4, аналогичная демоварианту 2025 и ЕГЭ 2024. Количество IP, в которых количество единиц не кратно n.

В терминологии сетей TCP/IP маской сети называют двоичное число, которое показывает, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу узла в этой сети. Адрес сети получается в результате применения поразрядной конъюнкции к заданному адресу узла и маске сети.

Сеть задана IP-адресом 122.159.136.144 и маской сети 255.255.255.248.

Сколько в этой сети IP-адресов, для которых количество единиц в двоичной записи IP-адреса **не кратно** 4?

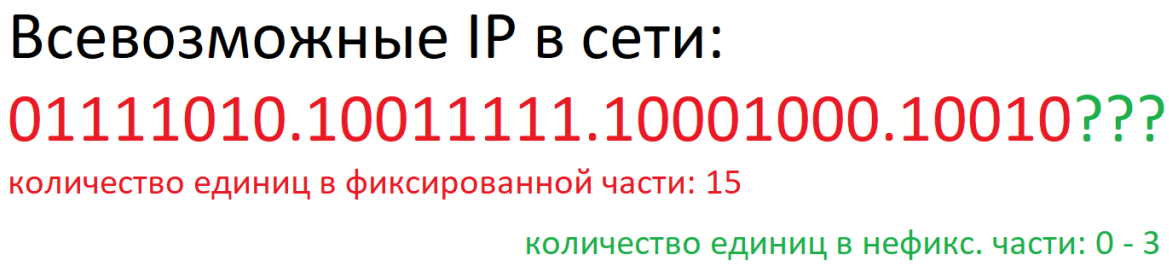
В ответе укажите только число.

Руками:

1. Нам дана маска и IP-адрес узла сети (ВАЖНО: указанный IP-адрес - это не всегда адрес сети). Переводим IP и маску в двоичный вид и находим сетевую (1 в маске) и узловую (0 в маске) часть.

Маска:
255.255.255.248 =
11111111.11111111.11111111.11111000
IP-адрес:
122.159.136.144 =
01111010.10011111.10001000.10010000

2. На изображении выше: красные разряды - сетевые = зафиксированные, зеленые - узловые = незафиксированные. В данной задаче IP-адрес совпал с адресом сети, т.к. все узловые разряды равны 0. У IP-адресов в этой сети узловые разряды могут быть какие угодно, поэтому всего различных IP-адресов (считая адрес сети и широковещательный): **2^3 = 8**, где **2** - количество вариантов либо 0, либо 1, а **3** - количество не зафиксированных узловых разрядов.



3. Т.к. нас интересует количество единиц в двоичной записи, то у каждого IP оно будет от 15 до 18 (15 в фиксированной части и от 0 до 3 нефиксированной). Нам нужно посчитать количество IP, в которых количество единиц не кратно 4, поэтому мы не должны считать IP с 16-ю единицами, т.е. в узловой части не должно быть ровно 1-ой единицы. Таким образом, ответом будет разность количества всех возможных вариантов IP и количества IP с ровно 1-ой единицей в узловой части, что равно количеству сочетаний из 3 по 1:

$$2^3 - C_3^1 = 8 - 3!/(2! * 1!) = 8 - 3 = 5$$

4. Получаем ответ **5.**

Кодом

1. С помощью библиотеки **ipaddress** и функции **ip_network** создаём сеть в переменной **net**, содержащей список всех возможных IP в сети, заданной маской и IP-адресом из условия. Создаём переменную **cnt** для подсчёта нужных для ответа IP.

```
from ipaddress import ip_network
```

```
net = ip_network('122.159.136.144/255.255.255.248',0)
cnt = 0
```

2. Циклом перебираем все IP в полученной сети и переводим их в двоичный вид, приведя их сначала к десятичному виду int. Так как нас не интересует количество нулей в двоичной записи, то можем не беспокоиться о добавлении незначащих нулей слева для нужной длины 32.

```
for ip in net:
    b_ip = bin(int(ip))[2:]
```

3. Проверяем условие и увеличиваем счётчик, если IP адрес подходит.

```
    if b_ip.count('1') % 4 != 0:
        cnt += 1
```

4. Выводим ответ после цикла и получаем **5**. Весь код:

```
from ipaddress import ip_network

net = ip_network('122.159.136.144/255.255.255.248',0)
cnt = 0
for ip in net:
    b_ip = bin(int(ip))[2:]
    if b_ip.count('1') % 4 != 0:
        cnt += 1
print(cnt)
```

Досрок 2025. На максимальный IP

В терминологии сетей TCP/IP маской сети называют двоичное число, которое показывает, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу узла в этой сети. Адрес сети получается в результате применения поразрядной конъюнкции к заданному адресу узла и маске сети. Широковещательным адресом называется специализированный адрес, в котором на месте нулей в маске стоят единицы. Адрес сети и широковещательный адрес не могут быть использованы для адресации сетевых устройств.
Сеть задана IP-адресом одного из входящих в неё узлов 143.168.72.213 и сетевой маской 255.255.255.240.
Определите наибольший IP-адрес данной сети, который может быть присвоен компьютеру. В ответе укажите найденный IP-адрес без разделителей.
Например, если бы найденный адрес был равен 111.22.3.44, то в ответе следовало бы записать 11122344.

Руками

1. Переводим маску и IP-адрес узла в двоичный вид и находим узловую часть адреса:

   Маска : 11111111.11111111.11111111.11111000
   IP-адрес: 01111010.10011111.10001000.10010000

2. Очевидно, что максимальный IP-адрес в заданной сети получится, если заменить все узловые разряды на 1, но тогда мы получим широковещательный адрес, а условии указано, что он не может быть присвоен устройству. Поэтому максимальный IP, который может быть присвоен компьютеру, получается при замене всех узловых разрядов на 1 кроме самого правого, ему присваиваем 0.

   Max IP:
   01111010.10011111.10001000.10010110

3. Переводим в десятичный вид и получаем ответ: **143.168.72.222.** *Ответ записывается без точек.*

Кодом

1. Создаём сеть через ip_network, передав туда IP и маску из условия.

```
net = ip_network(f'143.168.72.213/255.255.255.240',0)
```

2. Пробегаемся циклом по всем IP в сети и выводим их в консоль. Последний выведенный будет широковещательный адрес, а предпоследний — максимальный IP, который можно присвоить устройству.

```
for ip in net:
    print(ip)
```

3. Получаем ответ **143.168.72.222.** Весь код:

```
net = ip_network(f'143.168.72.213/255.255.255.240',0)
for ip in net:
    print(ip)

# Вывод:
# ...
# 143.168.72.220
# 143.168.72.221
# 143.168.72.222 - нужный
# 143.168.72.223 - широковещательный
```

Крылов, 15 вариант, перебор байта

В терминологии сетей TCP/IP маской сети называют двоичное число, которое показывает, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу узла в этой сети. Адрес сети получается в результате применения поразрядной конъюнкции к заданному адресу узла и маске сети.

Сеть задана IP-адресом 32.0.*A*.5, где *A* — некоторое допустимое для записи IP-адреса число, и маской сети 255.255.240.0. Определите минимальное значение *A*, для которого для всех IP-адресов этой сети в двоичной записи IP-адреса суммарное количество единиц в левых двух байтах не более суммарного количества единиц в правых двух байтах.

В ответе укажите только число.

Руками

1. Переводим маску и IP-адрес узла в двоичный вид и находим узловую часть адреса:

Маска :
11111111.11111111.11110000.00000000
IP-адрес:
00100000.00000000.$A_1A_2A_3A_4A_5A_6A_7A_8$.00000101

2. Видим, что в левых двух байтах все разряды зафиксированы и там всегда будет ровно одна единица, тогда как в правых байтах у нас первые 4 разряда зафиксированы, но зависят от A, а остальные 12 будут уникальны для каждого IP адрес и поэтому там может быть от 0 до 12 единиц вне зависимости от A. Исходя из условия, нам нужно, чтобы в правых двух байтах была всегда как минимум одна единица. Для минимальности A ставим её в разряд A4 и получаем ответ A = **16**.

Кодом

Аналогично 15 заданию будем перебирать всевозможные A (от 0 до 255) для третьего байта в IP. Заведём флаг, который будет по умолчанию равен True, но если условие из задачи не будет выполняться хотя бы 1 раз, то оно будет становиться False и сигнализировать, что A нам не подходит. Если флаг остаётся True после перебора всех ip в полученной при переборе A сети, то выводим A и получаем ответ **16**.

```
for A in range(256):
    flag = True
    net = ip_network(f'32.0.{A}.5/20',0) # маска указана как кол-во единиц
    for ip in net:
        b = bin(int(ip))[2:]
        if b[:16].count('1') > b[16:].count('1'):
            flag = False
```

```
            break
    if flag:
        print(A)
        break
```