Конспект к занятию 2

strim_13_2

Для решения задач этого типа будем использовать встроенный в язык Python модуль «ipaddress».

```
from ipaddress import *
```

В рамках данного модуля рассматриваются два основных инструмента для работы с IP-адресами и сетями: функции ip_address и ip_network. Первая предназначена для создания объектов IP-адреса, а вторая — для описания целых IP-сетей.

Функция ip_address

Данная функция позволяет создать объект IP-адреса. Пример использования выглядит следующим образом:

```
from ipaddress import *
ip = ip_address('123.20.103.136')
print(f'{ip:b}') #двоичная запись IP-адреса
```

Здесь создается объект IP-адреса с указанным значением. Этот объект предоставляет возможность получения различных характеристик IP-адреса, таких как его двоичное представление.

Результатом выполнения этого кода является двоичная запись IP-адреса длиной в 32 бита. Важно отметить, что независимо от исходного значения IP-адреса двоичная строка всегда будет содержать ровно 32 символа, при необходимости добавляя ведущие нули.

Результат:

```
01111011000101000110011110001000
```

Функция ip_network

Вторая функция, ip_network, служит для описания целой IP-сети. Ее использование включает указание диапазона сети и маски подсети. Пример вызова функции:

```
from ipaddress import *
ip = ip_address('123.20.103.136')
net = ip_network('192.168.0.0/255.255.255.240') #маска задана как IP-адрес
net = ip_network('192.168.0.0/28') #маска задана количеством единиц
#перебор IP-адресов в сети
```

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

В данном примере создается объект сети с заданным адресом и маской подсети. Эта функция также поддерживает альтернативный способ задания маски подсети через количество единиц в префиксе. Оба варианта создают одинаковый объект сети, позволяя затем выполнять операции над сетью, такие как перебор всех IP-адресов, входящих в данную сеть. Этот код выводит все возможные IP-адреса, принадлежащие указанной сети:

```
192.168.0.0
192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5
192.168.0.6
192.168.0.7
192.168.0.8
192.168.0.9
192.168.0.10
192.168.0.11
192.168.0.12
192.168.0.13
192.168.0.14
192.168.0.15
```

Таким образом, функции ip_address и ip_network предоставляют удобный интерфейс для работы с отдельными IP-адресами и сетями, обеспечивая широкие возможности для анализа и обработки данных в контексте сетевых взаимодействий.

Для работы с IP-сетями необходимо учитывать, что не любой адрес может быть использован в качестве аргумента функций ip_address и ip_network. Эти функции требуют корректных значений, соответствующих стандартам IP-адресации. Рассмотрим несколько важных аспектов.

Контроль ошибок

Функции ipaddress имеют встроенный механизм контроля ошибок. Если указать неверный IP-адрес или адрес сети, функция выдаст исключение, информирующее об ошибке. Например, если попытаться создать объект сети с неверным адресом:

```
from ipaddress import ip_network
# Некорректный адрес сети
net = ip_network('192.168.0.24.11')
```

Будет выдано сообщение об ошибке, указывающее на недопустимость такого формата адреса.

Нестрогий режим

Однако функции ipaddress поддерживают так называемый нестрогий режим (strict=False) , который позволяет избежать генерации исключений при вводе некорректных данных. Вместо этого функция попытается скорректировать введённые данные. Чтобы активировать этот режим, достаточно передать дополнительный аргумент 0 при создании объекта сети:

```
from ipaddress import ip_network
# Активация нестрогого режима
net = ip_network('192.168.0.24', 0)
```

В результате функция самостоятельно исправит указанный адрес сети на корректный, соответствующий правилам IP-адресации. Таким образом, вместо исключения пользователь получит объект сети, созданный на основе исправленного адреса.

Важность правильного указания адреса сети

Необходимо помнить, что адрес сети — это специальный адрес, имеющий свои особенности. Он не может быть произвольным и должен соответствовать определённым требованиям. Если адрес сети указан неверно, функция ip_network в строгом режиме выдаст ошибку, требующую исправления пользователем. Однако в нестрогом режиме функция автоматически внесёт необходимые изменения, приведя адрес к корректному виду. Таким образом, функции ipaddress обеспечивают гибкость в работе с IP-адресами и сетями, предоставляя возможность выбора между строгим контролем ошибок и автоматическим исправлением некорректных данных.

Одним из ключевых критериев определения корректности адреса сети является операция поразрядного умножения на соответствующую маску подсети. Если результат этой операции совпадает с исходным адресом сети, то данный адрес считается корректным и действительно обозначает начало диапазона сети. Формально это выражается следующим условием:

```
Адрес сети & Маска подсети=Адрес сети
```

Если указанное равенство соблюдается, то это свидетельствует о том, что адрес сети является корректным начальным адресом для данной подсети. В противном случае, если результат поразрядной операции отличается от исходного адреса, это указывает на необходимость коррекции адреса сети.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=50s

Задача № 1 (252)

По заданным IP-адресу узла сети и маске определите адрес сети:

IP-адрес: 135.12.170.217

Маска: 255.255.248.0

При записи ответа выберите из приведенных в таблице чисел 4 фрагмента четыре элемента IP-адреса и запишите в нужном порядке соответствующие им буквы без точек.

A	B	C	D	E	F	G	H
0	12	16	132	135	160	168	170

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=15m20s

Решение

Для решения задачи используем функцию ip_network, передав в нее IP-адрес и маску, для автоматического исправления некорректных данных укажем дополнительный аргумент 0

```
from ipaddress import *
net = ip_network('135.12.170.217/255.255.248.0',0)
print(net)
```

Результат работы программы:

```
135.12.168.0/21
```

Соотнесем полученный результат с приведенной таблицей.

Ответ:

EBGA

Задача №2 (16)

Для узла с IP-адресом 111.81.208.27 адрес сети равен 111.81.192.0. Чему равно наименьшее возможное значение третьего слева байта маски?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=21m55s

Решение

Для решения этой задачи нужно перебрать разные варианты маски, для каждого из них проверить, какой адрес сети получится с заданным IP-адресом, и найти те варианты, в которых получается заданный адрес. Перебор маски удобнее осуществлять именно через количество единиц, поскольку это упрощает процесс и делает его более понятным. Префикс маски задаётся в виде целого числа, которое указывает, сколько первых бит в маске равны единице. Остальные биты автоматически устанавливаются в ноль.

Алгоритм действий следующий:

1. Определить диапазон значений для перебора количества единиц в маске (обычно от 0 до 32).

2. Для каждой маски построить соответствующую сеть с использованием заданного IP-адреса.

3. Проверить, соответствует ли полученный адрес сети нужному адресу.

4. Сохранить результаты для тех случаев, когда соответствие достигается.

Код для решения задачи:

```
from ipaddress import *
for mask in range(33):
   net = ip_network(f'111.81.208.27/{mask}',0)
   print(net)
```

результат

```
0.0.0.0/0
0.0.0.0/1
...
111.81.0.0/16
111.81.128.0/17
111.81.192.0/18
111.81.192.0/19
111.81.208.0/20
...
```

Очевидно есть две маски, которые дадут нужный результат, т.е. заданный адрес сети. 111.81.192.0. Это 18 и 19 единиц.

Наименьшее возможное значение третьего слева байта маски 18 единиц. Рассмотрим два варианта получения ответа.

1 вариант:

Когда речь идёт о маске подсети с 18 единицами, это означает, что первые 18 бит маски установлены в значение '1'. Оставшиеся биты (в общей сложности должно быть 32 бита) заполняются нулями. Разберём это подробнее.

Маску подсети обычно представляют в формате четырёх октетов (байтов). Каждый октет состоит из восьми бит. Следовательно, 18 единиц можно разложить следующим образом:

- Первые восемь бит: 111111111
- Следующие восемь бит: 11111111
- Третий байт маски: 11000000
- Оставшиеся шесть бит дополняются нулями: 00000000

Это даёт нам следующую маску в бинарном представлении:

11111111 | 11111111 | 11000000 | 00000000

Можно перевести значение третьего бита маски 11000000 в десятичную форму, используя функцию int

```
int('11000000',2)
```

результат:

192

2 вариант:

Используем net.netmask . Это свойство объекта сети, которое хранит маску подсети. Его можно также вывести на печать:

```
from ipaddress import *
for mask in range(33):
   net = ip_network(f'111.81.208.27/{mask}',0)
   print(net, net.netmask)
```

Результат работы программы:

```
0.0.0.0/0 0.0.0.0
...
111.81.0.0/16 255.255.0.0
111.81.128.0/17 255.255.128.0
111.81.192.0/18 255.255.192.0
111.81.192.0/19 255.255.224.0
111.81.208.0/20 255.255.240.0
...

111.81.192.0/18 255.255.192.0
111.81.192.0/19 255.255.224.0
```

В данном случае мы сразу получим ответ – наименьший третий байт маски

Ответ: 192

```
#вывод только подходящих сетей
from ipaddress import *
for mask in range(33):
   net = ip_network(f'111.81.208.27/{mask}',0)
   if str(net) == f'111.81.192.0/{mask}':
```

```
    print(net, net.netmask)
```

результат:

```
111.81.192.0/18 255.255.192.0
111.81.192.0/19 255.255.224.0
```

Ответ:

192

Комментарий от Джобса: количество единиц в маске стоит перебирать до 30, так как для маски из 31 единицы мы получим пустую сеть, а маска с 32 единицами описывает специфическую сеть, которая в рамках ЕГЭ не используется. Также из-за особенностей работы с маской из 31 и 32 единиц может возникнуть коллизия ответа для некоторых постановок.

Задача № 3 (10160)

В терминологии сетей TCP/IP маска сети – это двоичное число, меньшее 2^{32}; в маске сначала (в старших разрядах) стоят единицы, а затем с некоторого места нули. Маска определяет, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес – в виде четырёх байт, причём каждый байт записывается в виде десятичного числа. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске.

Для узла с IP-адресом 76.155.48.2 адрес сети равен 76.155.48.0. Для скольких различных значений маски это возможно?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=35m5s

Решение

Для решения этой задачи следует перебрать разные значения маски, поразрядно перемножить маску на IP-адрес и проверить, что адрес сети будет соответствует заданному. Всё это выполнит программа аналогичная разобранной в предыдущей задаче.

```
from ipaddress import *
for mask in range(33):
   net = ip_network(f'76.155.48.2/{mask}',0)
   print(net)
```

Результат работы программы:

```
0.0.0.0/0
```

```
0.0.0.0/1
...
76.155.0.0/18
76.155.32.0/19
76.155.48.0/20
76.155.48.0/21
76.155.48.0/22
76.155.48.0/23
76.155.48.0/24
76.155.48.0/25
76.155.48.0/26
76.155.48.0/27
76.155.48.0/28
76.155.48.0/29
76.155.48.0/30
76.155.48.2/31
```

При 11 масках значение IP- сети соответствует заданному.

Ответ: 11

Задача № 4 (7018)

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. Для узла с IP-адресом 151.168.147.193 адрес сети равен 151.168.147.128. Каково наибольшее возможное количество единиц в двоичной записи маски?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=40m50s

Решение

Решение этой задачи очень похоже на решение предыдущей. Отличием является то, что здесь не нужно выводить запись маски, а нужно определить маску с наибольшим количеством единиц в ней.

```
from ipaddress import *
for mask in range(33):
   net = ip_network(f'151.168.147.193/{mask}',0)
   print(net)
```

Результат работы программы:

```
0.0.0.0/0
128.0.0.0/1
```

```
...
151.168.144.0/21
151.168.144.0/22
151.168.146.0/23
151.168.147.0/24
151.168.147.128/25
151.168.147.192/26
...
```

Адресу 151.168.147.128 соответствуют одна маска. Количеством единиц этой маски 25

Ответ:25

Задача №5 ()

В терминологии сетей TCP/IP маска сети – это двоичное число, меньшее 2^{32}; в маске сначала (в старших разрядах) стоят единицы, а затем с некоторого места нули.

Маска определяет, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес – в виде четырёх байт, причём каждый байт записывается в виде десятичного числа. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске.

Например, если IP-адрес узла равен 131.32.255.131, а маска равна 255.255.240.0, то адрес сети равен 131.32.240.0.

Два узла, находящиеся в одной сети, имеют IP-адреса 112.117.107.70 и 112.117.121.80. Укажите наименьшее возможное количество адресов в этой сети.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=43m50s

Решение

Для решения задачи подобрать такую маску, при которой оба заданных IP-адреса будут находиться в одной сети.

```
from ipaddress import *
for mask in range(33):
   net1 = ip_network(f'112.117.107.70/{mask}' ,0)
```

```
    net2 = ip_network(f'112.117.121.80/{mask}' ,0)
    if net1 == net2:
        print(net1, net2, 2**(32-mask))
```

Для решения будем переберем разные варианты масок для двух указанных подсетей. В качестве адреса одной из них будет использоваться первый IP-адрес, для второй используем второй IP-адрес. Вспомним, что количество возможных IP – адресов связано с количеством нулей в маске и может быть вычислено, как $2^{32 - mask}$, где 32 – это количество битов в маске, а mask – количество единиц.

Результат работы программы:

```
0.0.0.0/0 0.0.0.0/0 4294967296
0.0.0.0/1 0.0.0.0/1 2147483648
64.0.0.0/2 64.0.0.0/2 1073741824
96.0.0.0/3 96.0.0.0/3 536870912
112.0.0.0/4 112.0.0.0/4 268435456
112.0.0.0/5 112.0.0.0/5 134217728
112.0.0.0/6 112.0.0.0/6 67108864
112.0.0.0/7 112.0.0.0/7 33554432
112.0.0.0/8 112.0.0.0/8 16777216
112.0.0.0/9 112.0.0.0/9 8388608
112.64.0.0/10 112.64.0.0/10 4194304
112.96.0.0/11 112.96.0.0/11 2097152
112.112.0.0/12 112.112.0.0/12 1048576
112.112.0.0/13 112.112.0.0/13 524288
112.116.0.0/14 112.116.0.0/14 262144
112.116.0.0/15 112.116.0.0/15 131072
112.117.0.0/16 112.117.0.0/16 65536
112.117.0.0/17 112.117.0.0/17 32768
112.117.64.0/18 112.117.64.0/18 16384
112.117.96.0/19 112.117.96.0/19 8192
```

Максимальное количество адресов 4,3 миллиарда – это общее IP-пространство всех адресов. Минимальное количество адресов в последней строке – 8192.

Ответ: 8192

Задание №6 (262)

Два узла, находящиеся в одной сети, имеют IP-адреса 121.171.15.70 и 121.171.3.68. Укажите наибольшее возможное значение третьего слева байта маски сети. Ответ запишите в виде десятичного числа.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=51m50s

Решение

```
from ipaddress import *
for mask in range(33):
   net1 = ip_network(f'121.171.15.70/{mask}',0)
   net2 = ip_network(f'121.171.3.68/{mask}',0)
   if net1 == net2:
      print(net1, net1.netmask)
```

Результат работы программы:

```
0.0.0.0/0 0.0.0.0
0.0.0.0/1 128.0.0.0
64.0.0.0/2 192.0.0.0
96.0.0.0/3 224.0.0.0
112.0.0.0/4 240.0.0.0
120.0.0.0/5 248.0.0.0
120.0.0.0/6 252.0.0.0
120.0.0.0/7 254.0.0.0
121.0.0.0/8 255.0.0.0
121.128.0.0/9 255.128.0.0
121.128.0.0/10 255.192.0.0
121.160.0.0/11 255.224.0.0
121.160.0.0/12 255.240.0.0
121.168.0.0/13 255.248.0.0
121.168.0.0/14 255.252.0.0
121.170.0.0/15 255.254.0.0
121.171.0.0/16 255.255.0.0
121.171.0.0/17 255.255.128.0
121.171.0.0/18 255.255.192.0
121.171.0.0/19 255.255.224.0
121.171.0.0/20 255.255.240.0
```

Для сети 121.171.0.0 соответствуют 5 вариантов масок. При этом наибольшее значение третьего байта маски – это 240.

Ответ: 240

Задание № 7(16260)

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.

Как адрес сети, так и широковещательный адреса не могут использоваться в качестве IP-адресов узлов сети.

Известно два узла с IP-адресами 123.20.103.136 и 123.20.103.151 принадлежат разным сетям с одинаковой маской.

Определите значение 4 байта маски в этих сетях. Найденное значение представьте в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=55m25s

Решение

Для понимания особой специфики вопроса этой задачи сначала выполним её решение в аналитической форме.

Переведём IP-адреса в двоичную форму. Адрес первого узла:

```
print(f'{123:08b}', f'{20:08b}', f'{103:08b}', f'{136:08b}')
```

```
01111011 00010100 01100111 10001000
 Адрес второго узла:
```

```
print(f'{123:08b}', f'{20:08b}', f'{103:08b}', f'{151:08b}')
```

```
01111011 00010100 01100111 10010111
```

Анализируя задачу, можно сделать вывод, что рассматриваемые IP-адреса принадлежат двум разным сетям, каждая из которых использует одинаковую маску. Согласно условиям задачи, ни один из указанных IP-адресов не может быть адресом сети или широковещательного адреса. Это накладывает ограничения на возможное значение маски.

Для начала рассмотрим вариант маски с 29 единичными битами. Такая маска в двоичном представлении будет иметь вид:

`11111111|11111111|11111110|00000000`
или в десятичном представлении:

`255.255.254.0255.255.254.0`
При использовании такой маски первый узел (123.20.103.136) окажется в одной сети, а второй узел (123.20.103.151) — в другой. Однако, при таком распределении адресов оба указанных IP окажутся адресами сети или широковещательными адресами, что противоречит условию задачи.

Следовательно, маска с 29 единицами не подходит.

```
from ipaddress import *
ip1 = ip_address('123.20.103.136')
ip2 = ip_address('123.20.103.151')
for mask in range(33):
   net1 = ip_network(f'{ip1}/{mask}',0)
   net2 = ip_network(f'{ip2}/{mask}',0)
   if net1!=net2 and net1[0]<ip1<net1[-1] and net2[0]<ip2<net2[-1]:
      print(net1, net2, net1.netmask)
```

Ответ: 28

Программная проверка адресов узлов

```
from ipaddress import *
ip1 = ip_address('123.20.103.136')
ip2 = ip_address('123.20.103.151')
for mask in range(33):
   net1 = ip_network(f'{ip1}/{mask}',0)
   net2 = ip_network(f'{ip2}/{mask}',0)
   if net1!=net2 and net1[0]<ip1<net1[-1] and net2[0]<ip2<net2[-1]:
      print(net1, net2, net1.netmask)
```

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=1h3m25s

Результат работы программы:

Задание №8 (11774)

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 214.96.0.0 и маской сети 255.240.0.0. Сколько в этой сети IP-адресов, у которых количество нулей в двоичной записи IP-адреса кратно трём?
В ответе укажите только число.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=1h17m55s

Решение

```
from ipaddress import *
```

```
net = ip_network('214.96.0.0/255.240.0.0')
k = 0
#получим двоичную запись для каждого значения ip-адреса сети
for ip in net:
   b = f'{ip:b}' #для Python 3.9+
   #b = f'{int(ip):032b}' #для Python 3.8 и ниже
#проверим заданное условие для двоичной записи ip-адреса
   if b.count('0') % 3 == 0:
      k+=1
print(k)
```

Результат работы программы:

349526
Ответ: 349526

Задание №9(6841)

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 192.168.248.176 и маской сети 255.255.255.240. Сколько в этой сети IP-адресов, для которых количество единиц в двоичной записи IP-адреса больше, чем количество нулей? В ответе укажите только число

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=1h34m40s

Решение

```
from ipaddress import *
net = ip_network('192.168.248.176/255.255.255.240' )
k = 0
#получим двоичную запись для каждого значения ip-адреса сети
for ip in net:
  b = f'{ip:b}'
#проверим заданное условие
  if b.count('1')>b.count('0'):
    k += 1
print(k)
```

Результат работы программы:

1

Ответ: 1

Задание №10 (7039)

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 154.24.165.32 и маской сети 255.255.255.224. Сколько в этой сети IP-адресов, для которых в двоичной записи IP-адреса суммарное количество единиц в левых двух байтах меньше суммарного количества единиц в правых двух байтах?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=1h37m20s

Решение

Определим подходящие под условие задачи значения. Для этого, используем срезы [:16], [16:32] и метод «count», сосчитаем количество единиц.

```
from ipaddress import *
net = ip_network('154.24.165.32/255.255.255.224')
k = 0
for ip in net:
  b = f'{ip:b}'
#проверим заданное условие, используя двоичную запись адреса
  if b[0:16].count('1') < b[16:32].count('1'):
    k += 1
print(k)
```

Результат работы программы:

26

Ответ : 26

Задача № 11()

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, в которой

содержится узел с IP-адресом 99.8.254.232, задана маской сети 255.255. A A.0, где A A - некоторое допустимое для записи маски число. Определите минимальное значение A 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

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241207?t=1h46m

Решение

Используем срезы [:16], [16:32] и метод «count», сосчитаем количество единиц IP-адресов. Запишем эти вычисления в условие if all ()

```
from ipaddress import *
for mask in range(16,25):
   net = ip_network(f'99.8.254.232/{mask}',0)
#проверим заданное условие, используя двоичную запись адреса
   if all(f'{ip:b}'[0:16].count('1') <= f'{ip:b}'[16:32].count('1') for ip in net):
    print(net.netmask)
```

Результат работы программы:

255.255.248.0 минимальное значение
255.255.252.0
255.255.254.0
255.255.255.0

Ответ: 248