## 赛题解读

众所周知，N多年来内存的每个bit都存放着0和1两种状态，然而这一事实在多年前发了变化，内存一族的分支T突发了变异，产生了神奇的功能，它们的每个bit不仅可以存放0和1，还能存放一种被称作“\*”或者“任意”的状态，这个状态即可以看作是0又可以看作是1。T的这个有趣的特性立即引起了人们广泛的关注，吸引了众多的人来研究它，以便将其应用到各个领域。功夫不负有心人，经过若干次的试验，人们终于完全掌控了T的特性并发明了一些新的器件。

T在网络通信领域应用中自然也不会缺席，并且网络专家们提供了极其简单的方法来操控T。相比传统的内存往其地址上写0和1组成的data（数据），我们只要再加一个mask（掩码）便能显现T的特性。例如我们写一个data:0b00000010和一个mask:0b11111110（data、mask bits一一对应），它表示的数就是（result）：

0 b 0 0 0 0 0 0 1 0（data）

0 b 1 1 1 1 1 1 1 0（mask）

--------------------------------

0 b 0 0 0 0 0 0 1 \*（result）

也就是说mask中bit为1，result相应bit维持data原值；mask中bit为0，不管data相应bit原来是0还是1，result相应bit就为\*；反过来，如果bit要写0或者1，就置data bit为0或者1，mask为1，如果bit既要表示0又要表示1就置mask相应bit为0。示例中因为\*即可以表示0又可以表示1，所以我们实际写进去的result就表示了2（0b00000010）和3（0b00000011）两个数。操控So easy!

很快专家们发现这个属性非常适合用于网络通信中的报文分类，所谓报文分类就是指给定一些规则和报文，找出每个报文各属于哪个规则，规则中指定了几个字段的值和匹配的规则，本质上就是数值的比较，常见的匹配规则有：

* 前缀匹配：仅前缀部分相同
* 精确匹配：每位都必须相同
* 范围匹配：字段值必须在一个范围之内

IPv4地址（32bits)为前缀匹配如：格式1.1.1.0/24表示如果报文的IP跟规则的前24bits相同就算匹配，在T中可以表示为data:0x01010100，mask:0xffffff00即后8bits任意。

协议号（8bits)为精确匹配如：格式0x11/0xff表示每位都必须相同，在T中可以表示为data:0x11，mask:0xff即每个bits都关心。

TCP/UDP端口（16bits)为范围匹配：如格式1:5表示报文相应的端口号在1到5内就算匹配。这个在T中好像不太好表示啊，简单的方法就是分解成1、2、3、4、5即data:0x0001，mask:0xffff；data:0x0002，mask:0xffff；data:0x0003，mask:0xffff；data:0x0004，mask:0xffff；

data:0x0005，mask:0xffff。这个处理虽然简单，但是1条变成了5条太浪费资源了，能有好的解决方案吗？

## 赛题内容

对由五元组即目的IPv4地址、源IPv4地址、目的端口、源端口、协议5个字段以及结果组成的规则进行处理得到满足T属性的数据；给定报文进行正确的分类，找到报文匹配的规则对应的结果。标准输入包含了多个规则（包括规则对应的结果）以及多个报文，具体格式见后面的说明，程序需要实现的功能：

1.读取标准输入中的规则部分，生成满足T属性的数据并输出到标准输出。

2.读取标准输入中的报文部分，使用1中的输出数据（存放在内存中的数据）进行分类，找到匹配的条目对应的结果输出到标准输出。

**条件与要求：**

1. 输入中规则数最多为512个，报文数最多为100个，算法实现需要正确处理如此数量的规则和报文。
2. 1中对每个规则进行处理，输出成data、mask对的形式（见第3部分的示例），T每个单元有80bits、160bits、320bits这3种规格，只能选用一种。例如题目中的五元组总共104bits，若直接存放可以用160bits或者320bits，多余的bits都是可以使用的，320bits会产生比160bits多1倍的容量。
3. 规则的目的IP字段不存在交集。即如果用s1和s2分别表示匹配任意两个不同规则的目的IP字段的非负整数的集合，那么s1∩s2=Ø。
4. 标准输入中前面的规则优先级高，若存在报文匹配多个规则的情况，取前面规则对应的结果。
5. 规则和报文包含五元组信息（规则中还带有一个结果），即目的IP（IPv4）、源IP（IPv4）、目的端口、源端口、协议。

匹配方式如下：

目的IP、源IP: 前缀匹配（如1.1.1.1/24，这里的前缀为24，前24bits相同就能匹配）

目的端口、源端口：范围匹配（如1:255，在1到255内都能匹配）

协议：精确匹配（如每个bits都相等才匹配）

1. 2中描述的对每个报文进行分类，是将每个报文与1中输出的若干对data、mask表示的数据顺序进行比较（必须使用data、mask对数据，从第1对到最后1对依次处理)。
2. 程序需要使用c或者c++语言实现。
3. 在网站上提交主要的算法说明文档，包括data、mask数据生成和分类实现的说明。

**特别说明：**

1. 2中报文跟T中的data、mask条目匹配是指报文的每个bit跟条目中的每个bit都匹配，单个bit匹配规则如下：

|  |  |  |  |
| --- | --- | --- | --- |
| mask bit | data bit | pkt bit | match result（1匹配，0不匹配） |
| 0 | 0或者1 | 0或者1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

1. 跟T中的data、mask条目进行匹配的数据无需为读取的原始报文。
2. 不同data、mask对是独立的，不能关联使用。
3. 规则中IP的前缀长度为8到32，端口为0到65535，协议为0到255，算法需要适应此范围的任何数据。
4. 规则的每个字段只能单独处理，2中不能根据原始规则数据进行分类。

## 程序输入和输出说明

1. 输入格式：输入分为两个部分，第一部分为规则，第二部分为报文。第一行为总的规则数，从第二行开始为具体的规则，每行包含目的IP、源IP、目的端口、源端口、协议以及结果共6个信息，各个字段之间用空格分隔。规则部分后面为报文部分，报文部分第一行为总的报文数，后面每一行是报文信息，包含目的IP、源IP、目的端口、源端口、协议共5个字段，各个字段之间用空格分隔。见下面的例子：

1

1.1.1.0/24 1.1.2.0/24 1:5 3:3 0x11/0xff 1

1

1.1.1.1 1.1.2.2 5 3 17

1. 输出格式：第一行为五元组每个字段占用的bits数以及数据条目数，后面每两行为一个data/mask对，data/mask为16进制表示，data行最后一个数值为规则对应的结果，如1）中的第一个条规则按照T的属性可以分解成下面的输出（数值32 32 16 16 8仅仅是示例）：

32 32 16 16 8 5

data:0x01010100 0x01010200 0x0001 0x0003 0x11 1//data和mask对

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0002 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0003 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0004 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0005 0x0003 0x111 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

1. 每个报文分类的结果输出到标准输出。

如1）中报文1.1.1.1 1.1.2.2 5 3 17得到下面的匹配结果：

1

因为：

* 报文目的IP 1.1.1.1前24bits跟规则中目的IP前24bits相同；
* 报文源IP 1.1.2.2前24bits跟规则中源IP前24bits相同；
* 报文目的端口5在规则中目的端口1到5的范围内；
* 报文源端口3在规则中源端口3到3的范围内；
* 协议17跟规则中协议0x11相等。

实际分类是使用data、mask对进行匹配，报文能跟2）中第5个data、mask对匹配上，取其结果是1。

最终输出内容如下：

32 32 16 16 8 5

data:0x01010100 0x01010200 0x0001 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0002 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0003 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0004 0x0003 0x11 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

data:0x01010100 0x01010200 0x0005 0x0003 0x111 1

mask:0xffffff00 0xffffff00 0xffff 0xffff 0xff

1

## 评分规则

1. 输出格式要正确，格式不符合不得分。
2. 输出的分类结果要正确，不正确不得分。
3. 对输入报文进行分类必须使用经过处理输出的数据，不符合要求不得分。
4. 输出的数据容量越小，排名越高。

容量的计算方法：容量 = 每个条目的bits数\*条目数

每个条目的bits数的计算方法为：设输出中第一行前五个字段之和sum，sum<=80，按80bits计算；sum>80bits，按照160bits计算;sum>160bits，按照320bits计算;sum>320不得分。

条目数：每对data、mask为一个条目。

1. 输出中每行data、mask中的各个字段必须在第一行给定的位数内（分别对应前面5个字段），不正确不得分。
2. 如果得分相同，程序运行时间短者胜出。
3. 若运行时间也相同，先提交者胜出。
4. 后台会有多组数据对算法性能进行全面测试，最终得分以所有用例的输出容量数值加和得到。并按评分规则小由到大排名。
5. 线上提供600MB内存供程序运行。
6. 除了线上跑分，赛题组会阅读文档和代码以确认算法的正确性，若发现任何作弊、算法不正确等情况则取消成绩。