

## 一、实验要求：

1. 匹配 IP 地址
2. 在此基础上，在一段包含有多个 IP 地址的文字里，导出所有的正确的 IP。（"IP.txt"）  
"211.64.154.21 %^ &\* 2222.17.76.0 0.0.0.256 (^&\$# 211.64.154.256 371.-1.4.999 )  
( &%..... 137.1.1.15 %.....& 1.2.3.4 %.....&\* 0.0.10.0 %.....& aab.c34.62.5 %^& 10.1.1.1"
3. 抓取页面（ <http://www.ouc.edu.cn/main.htm> ）
4. 所有 a 标签中链接地址

## 二、实验环境：

VC++ 6.0, Windows 7, Greta 库

## 三、问题分析：

1. Greta 库的如何使用？  
看示例文档，主要有两个类 `rpattern` 正则表达式类、`match_results` 执行结果类，包括其成员函数的调用方法，除了文档还可以通过网上查找更详细的使用例子。另外这个库是在 VC++ 编译器下才能正确编译通过的。
2. 匹配 IP 地址的正则表达式？  
`((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|0[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|0[0-9])` 网上有很多，也可以自己写，正则表达式不唯一。表达式思路是把 0~255 分成四个范围进行表示。
3. 匹配 a 标签中链接地址的正则表达式？  
由于一个 html 文档中，不只有 a 标签才有链接，所以直接写一个匹配链接的正则表达式并不能符合实验要求。因此，分两步：第一步先通过正则表达式找到所有的 a 标签，再在这些 a 标签里匹配 href 属性的值，也就是链接。  
匹配 a 标签的正则表达式是 `<a.*href=.*>`  
匹配链接的正则表达式是 `(href=\"([^\"]+)\")|(href='([^\']*)*')`

## 四、实验过程、步骤及原始记录：

1. 程序通用流程：读取文件→通过正则表达式匹配需要信息→输出内容，典型 IPO 模式。
2. IP 地址的匹配源代码：

```

#include<iostream>
#include "regexr2.h"
#include<fstream>

using namespace std;
using namespace regex;

#pragma comment(lib, "Greta.lib")

int main()
{
    match_results results;
    ifstream fileStream("IP.txt");
    if(!fileStream)
    {
        cout << "file open failed!" << endl;
        return 0;
    }

    string str = "";
    getline(fileStream, str);
    fileStream.close();
    //string str = "0.0.0.256 ";
    rpattern pat
        ("((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])\\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])")
        , GLOBAL | ALLBACKREFS);
    match_results::backref_type br = pat.match(str, results);
    if(br.matched)
    {
        match_results::backref_vector vec = results.all_backrefs();
        int a = results.cbackrefs();
        for(int i = 0; i < a; i+=4)//the reason "+4", try i++, you will know. I don't know why.
        {
            cout << vec[i] << endl;
        }
    }
    else
        cout << "error" << endl;

    return 0;
}

```

3. 匹配所有 a 标签中链接地址的源代码:

```

#include<iostream>
#include "regexr2.h"
#include<fstream>
#include<vector>
#include<string>

using namespace std;
using namespace regex;

#pragma comment(lib, "Greta.lib")

int a;

typedef struct{
    int lineNum;
    string href;
}printResultFormat;

int main()
{
    match_results results;
    vector<printResultFormat>hrefBuff;

    ifstream fileStream("main.html");
    if(!fileStream)
    {
        cout << "file open failed!" << endl;
        return 0;
    }

    int lineCount = 0;
    match_results aTagResult;
    rpattern aTagPat("\\<a.*href=.*\\>");
    rpattern hrefPat("(href=\\\"([^\"]+\\\")| (href='([^']*)'"))");

```

```

    string str = "";
    while(getline(fileStream, str))
    {
        lineCount++;
        match_results::backref_type br = aTagPat.match(str, aTagResult);
        if(br.matched)
        {
            string atemp = br.str();
            match_results hrefResult;
            match_results::backref_type gethref = hrefPat.match(atemp, hrefResult);
            if(gethref.matched)
            {
                printResultFormat temp;
                temp.href = gethref.str();
                temp.lineNum = lineCount;
                hrefBuff.push_back(temp);
            }
        }
    }

    fileStream.close();

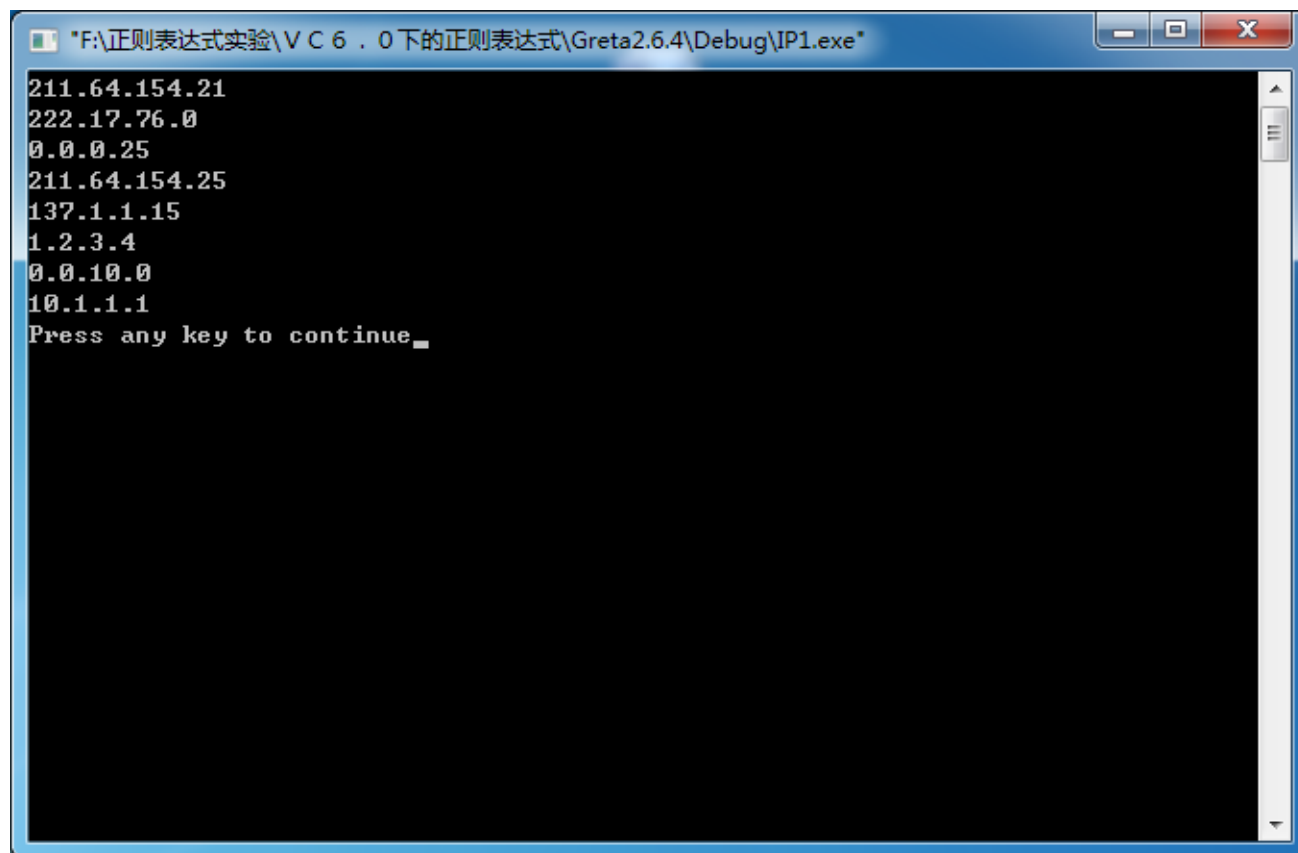
    for(int i = 0; i < hrefBuff.size(); i++){
        cout << hrefBuff[i].lineNum << "\t" << hrefBuff[i].href << endl;
    }

    return 0;
}

```

## 五、 实验结果：

识别 IP 地址



```
"F:\正则表达式实验\VC 6.0下的正则表达式\Greta2.6.4\Debug\IP1.exe"  
211.64.154.21  
222.17.76.0  
0.0.0.25  
211.64.154.25  
137.1.1.15  
1.2.3.4  
0.0.10.0  
10.1.1.1  
Press any key to continue_
```

说明：

- a) 2222.17.76.0 中的除了第一个 2 后的字符串是 IP 地址，因此会识别出 222.17.76.0
- b) 同理 0.0.0.256 中除了最后一个字符 6 的字符串也是 IP 地址，因此会识别出 0.0.0.25

识别 a 标签中所有链接地址

说明：

- a) 输出格式为：源文件中的行号\t 匹配到 href 属性的值。
- b) 不对 href 属性的具体内容检测，即使是通过 js 脚本生成，也识别为链接地址。

```
"F:\正则表达式实验\VC 6.0下的正则表达式\Greta2.6.4\Debug\html1.exe"
39 href="http://old.ouc.edu.cn/index.htm"
40 href="http://web.ouc.edu.cn/"
41 href="http://www.ouc.edu.cn/main.htm"
51 href="/main.htm"
58 href="javascript:void(0);"
62 href="/6698/list.htm"
71 href="/6697/list.htm"
80 href="/78/e7/c6696a30951/page.htm"
89 href="/5866/list.htm"
98 href="/lrld/list.htm"
107 href="/xrld/list.htm"
116 href="/xxjj/list.htm"
128 href="/bmsz/list.htm"
135 href="javascript:void(0);"
139 href="/yldxjs/list.htm"
148 href="/gjzdxk/list.htm"
160 href="http://web.ouc.edu.cn/rsc/1065/list.htm"
164 href="http://211.64.142.71/base/frame/login.jsp?FM_SYS_ID=zghydx"
173 href="http://web.ouc.edu.cn/rsc/923/list.htm"
182 href="http://web.ouc.edu.cn/rsc/921/list.htm"
194 href="javascript:void(0);"
198 href="http://web.ouc.edu.cn/jxjy/"
207 href="http://iec.ouc.edu.cn/article/"
216 href="http://web.ouc.edu.cn/xpb"
225 href="http://web.ouc.edu.cn/grad/"
234 href="http://web.ouc.edu.cn/jwc"
246 href="javascript:void(0);"
250 href="/xsqk/list.htm"
259 href="/dxkxyq/list.htm"
268 href="/dfhdcc/list.htm"
277 href="/bshkyldz/list.htm"
286 href="/lsjj/list.htm"
295 href="http://www2.ouc.edu.cn/cimst/"
304 href="/rwshkx/list.htm"
313 href="/zrkxyjskx/list.htm"
325 href="javascript:void(0);"
329 href="http://career.ouc.edu.cn/index.shtml"
338 href="http://web.ouc.edu.cn/jxjy/"
347 href="http://iec.ouc.edu.cn/article"
356 href="http://web.ouc.edu.cn/yzb/"
365 href="http://www2.ouc.edu.cn/zsh/zsh/"
377 href="javascript:void(0);"
381 href="http://iec.ouc.edu.cn/"
390 href="http://www2.ouc.edu.cn/international/"
402 href="/5518/list.htm"
```

```
"F:\正则表达式实验\VC 6.0下的正则表达式\Greta2.6.4\Debug\html1.exe"
377 href="javascript:void(0);"
381 href="http://iec.ouc.edu.cn/"
390 href="http://www2.ouc.edu.cn/international/"
402 href="/5518/list.htm"
406 href="http://library.ouc.edu.cn/"
415 href="http://222.195.158.249/"
424 href="http://www.ouc.edu.cn/xuebao/"
436 href="http://web.ouc.edu.cn/xxgk/"
461 href="javascript:void(0)"
674 href="javascript:w3openWindow();"
679 href="javascript:w3nextAd<' + i + '>"
681 href="javascript:w3nextAd<' + i + '>"
728 href="http://xinwen.ouc.edu.cn/"
738 href="http://xinwen.ouc.edu.cn/"
745 href="http://www.ouc.edu.cn/xsdt/list.htm"
756 href="http://urp.ouc.edu.cn/"
763 href="http://211.64.142.67/kjc/detail.asp?cat=A00020001000300050002&id=3902"
770 href="/b4/fb/c332a46331/page.htm"
777 href="http://urp.ouc.edu.cn/"
784 href="http://211.64.142.67/kjc/detail.asp?cat=A00120003&id=3922"
791 href="http://web.ouc.edu.cn/jwc/b4/8f/c6517a46223/page.htm"
798 href="http://web.ouc.edu.cn/rsc/b4/9e/c4042a46238/page.htm"
805 href="http://www2.ouc.edu.cn/hqc/Article_Show.asp?ArticleID=896"
811 href="/xsdt/list.htm"
827 href="http://xinwen.ouc.edu.cn/Index.html"
849 href="http://www.ouc.edu.cn/gz/main.htm"
852 href="http://www.ouc.edu.cn/xs/main.htm"
855 href="http://www.ouc.edu.cn/jzg/main.htm"
858 href="http://www.ouc.edu.cn/xy/main.htm"
861 href="http://www.ouc.edu.cn/ks/main.htm"
936 href="http://web.ouc.edu.cn/hydx/"
945 href="http://web.ouc.edu.cn/shisanwuguihua/"
949 href="http://web.ouc.edu.cn/shisanwuguihua/"
962 href="http://web.ouc.edu.cn/xiaoqing/"
967 href="http://xinwen.ouc.edu.cn/qzlx/"
984 href="http://weibo.com/oucnews"
985 href="http://t.qq.com/oucnews/"
986 href="http://t.people.com.cn/ouc1924"
987 href="javascript:void(0)"
998 href="" + p.url + ""
1013 href='http://web.ouc.edu.cn/xiaoqing/'
1018 href='http://web.ouc.edu.cn/xiaoqing/'
1027 href='http://web.ouc.edu.cn/xiaoqing/'
Press any key to continue_
```

## 六、 实验中遇到的问题及解决方法:

### 1. 识别 IP 地址的二义性:

像 2222.17.76.0 这种字符串,我们第一反应是这个不是 IP 地址。但是程序会把这个串识别出一个 IP 地址“222.17.76.0”,这也没错,匹配时发现第一个 2 开始是无法匹配的,但第二个 2 开始是可以匹配的,因此程序会把该字符串除第一个字符外的串识别为 IP 地址。一开始看运行结果的时候以为自己的正则表达式错了。还特意换了好几个正则表达式去试,发现结果一样。之后和同学交流才发现这一点。类似的诸如 0.0.0.256 也会识别出 0.0.0.25 这个 IP 地

址等。

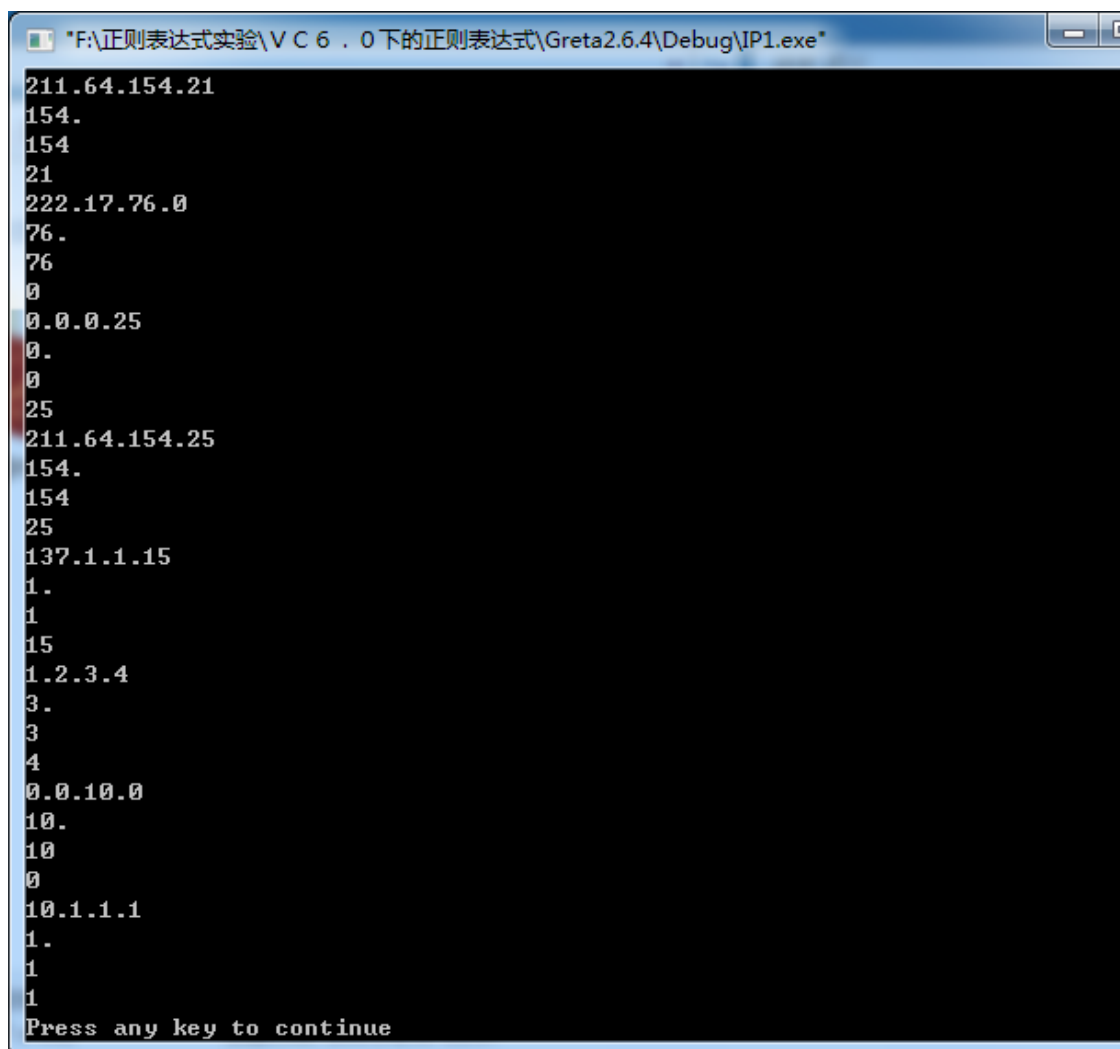
## 2. Greta 中 rpattern 正则表达式类中的 match 函数参数疑问：

看了示例文档后，发现 match 函数有三个参数，但文档对三个参数的具体约定并没有描述清楚；而且所给是示例代码中，match 函数只能匹配第一个匹配串，因为返回参数就是一个具体的匹配串而不是列表数组或迭代器之类的。经过在网上多次查找资料才发现，要想匹配全部的话，match 函数第三个参数应该是设置为 GLOBAL|ALLBACKREFS，其实文档中有提及，只是描述让人误解。同时作为一个优秀的第三方库，提供了迭代器 match\_results::backref\_vector，all\_backrefs() 函数正是返回该类型。配合使用即可匹配全部。

当然，也可以换一种思路，就是匹配一个替换一个，直到不能再匹配。但是显得不够清晰明了了。

## 3. 匹配问题：

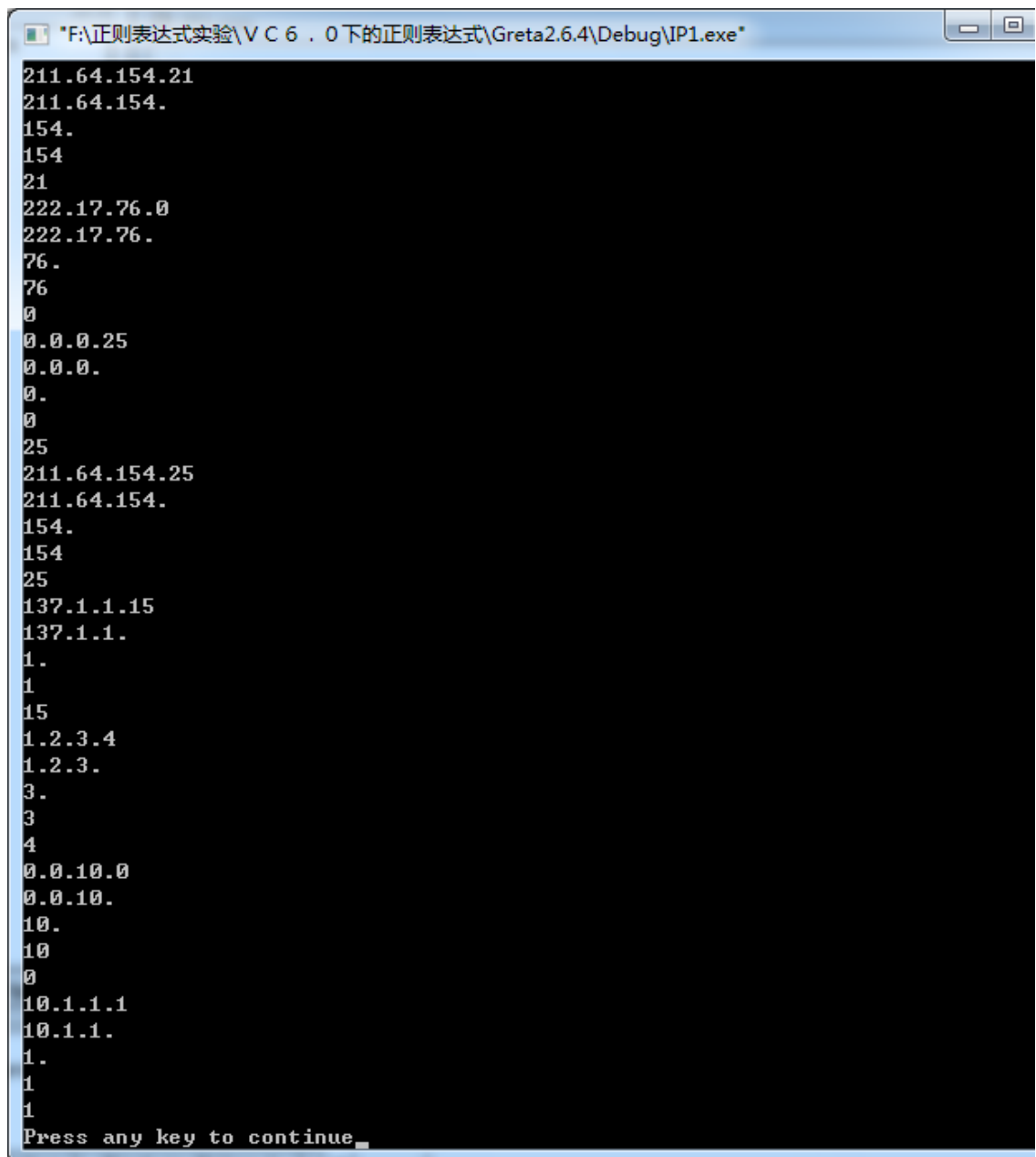
通过输出迭代器的内容发现，在 IP 地址实验中，迭代器里的内容并不只是 IP 地址，还有已经被正确识别的 IP 地址的部分串，而且这些串特别有规律，如图下图是上面代码所写的正则表达式匹配后对应的所有迭代器内容：



```
"F:\正则表达式实验\VC 6.0下的正则表达式\Greta2.6.4\Debug\IP1.exe"
211.64.154.21
154.
154
21
222.17.76.0
76.
76
0
0.0.0.25
0.
0
25
211.64.154.25
154.
154
25
137.1.1.15
1.
1
15
1.2.3.4
3.
3
4
0.0.10.0
10.
10
0
10.1.1.1
1.
1
1
Press any key to continue
```

当时我没搞懂，只是发现迭代器里的内容有规律，要想正确输出想要的结果，就得要跳着输出迭代器里的内容。检查实验时，我把问题告诉老师，老师说是这个库的函数问题，正则表达式里的括号的内容匹配后都算作匹配内容。我试着修改正则表达式，多加了一对可有可无的括号：

((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[0-9][0-9]|0[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[0-9][0-9]|0[0-9])，即把前面识别的三个数字加点的格式加一个括号，运行程序后结果如下：



```
F:\正则表达式实验\VC 6.0下的正则表达式\Greta2.6.4\Debug\IP1.exe
211.64.154.21
211.64.154.
154.
154
21
222.17.76.0
222.17.76.
76.
76
0
0.0.0.25
0.0.0.
0.
0
25
211.64.154.25
211.64.154.
154.
154
25
137.1.1.15
137.1.1.
1.
1
15
1.2.3.4
1.2.3.
3.
3
4
0.0.10.0
0.0.10.
10.
10
0
10.1.1.1
10.1.1.
1.
1
1
Press any key to continue.
```

这也能证明老师说得没错，并非我的代码有问题，而是这个库的函数的特点。

4. 不同语言，不同库，对正则表达式的规则的支持有所不同：

例如.需要转义，理论上\即可，但是在这里需要\\.双斜杠转义。在和同学交流的过程中我发现有些同学的环境只需要\即可，另外有些转义字符也是同理。此外，我还发现一些规则也是不一定适用的，例如\b理应表示单词边界，但是我的环境并不能实现。

5. 小小套路，正则表达式是贪婪匹配，Greta 库里也没有提供最小匹配的函数，若想最小匹配，某些情况可以改写正则表达式。例如在匹配 a 标签中所有链接，以双引号结尾，有可能 a 标签中的 href 值先写，而后面属性的值也有引号，那么就会和后面的内容一起匹配，这样就不是链接了。此时可以通过\b 单词边界实现对边界的固定。