

# **CS-532 Web Science: Assignment #1**

Due on Thursday, January 28, 2016

*Dr. Michael L. Nelson*

**Plinio Vargas**  
pvargas@cs.odu.edu

## Contents

Problem 1	1
Problem 2	4
Problem 3	8

## List of Figures

1	Test of <i>a1_v1.1.py</i> at <a href="http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html">http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html</a> with updated code . . . . .	7
2	Test of <i>a1.py</i> at <a href="http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html">http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html</a> with original code . . . . .	8
3	Test of <i>a1.py</i> at <a href="http://www.cs.odu.edu/">http://www.cs.odu.edu/</a> . . . . .	8
4	Test of <i>a1.py</i> at <a href="http://www.vbschools.com/curriculum/gifted/">http://www.vbschools.com/curriculum/gifted/</a> . . . . .	9
5	Bow-Tie Graph Representation . . . . .	9

## Problem 1

Demonstrate that you know how to use “curl” well enough to correctly POST data to a form. Show that the HTML response that is returned is “correct”. That is, the server should take the arguments you POSTed and build a response accordingly. Save the HTML response to a file and then view that file in a browser and take a screen shot.

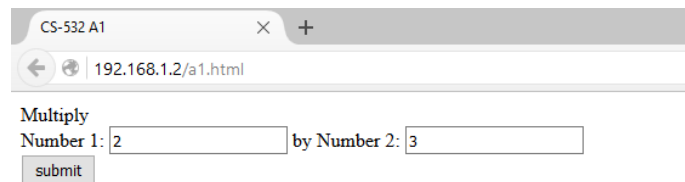
### SOLUTION

[http://curl.haxx.se\[1\]](http://curl.haxx.se[1]) was an excellent resource to unveil the power of ***curl*** command. There are many options available with *curl*, including passing of cookies, which are intensely used by current servers for Cross-Site Request Forgery (CSRF). Then, in order to simplify our demonstration I created a simple html page in my local server that takes two fields: *number1* and *number2*. I named this resource *a1.html*:

a1.html

```
.....
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4  <title>CS-532 A1</title>
5  </head>
6
7  <body>
8  <form method="post" action="a1.php">
9  <label>Multiply</label><br />
10 <label>Number 1:</label>
11 <input name="number1" type="text" value="2" />
12 <label> by </label>
13 <label>Number 2:</label>
14 <input name="number2" type="text" value="3" /> <br />
15 <input name="submit" type="submit" value="submit">
16 </form>
17 </body>
18 </html>
```

### Screen Shot of html page



CS-532 A1

192.168.1.2/a1.html

Multiply

Number 1:  by Number 2:

Below, is the action script *<a1.php>* run by the server when the form is submitted:

### a1.php

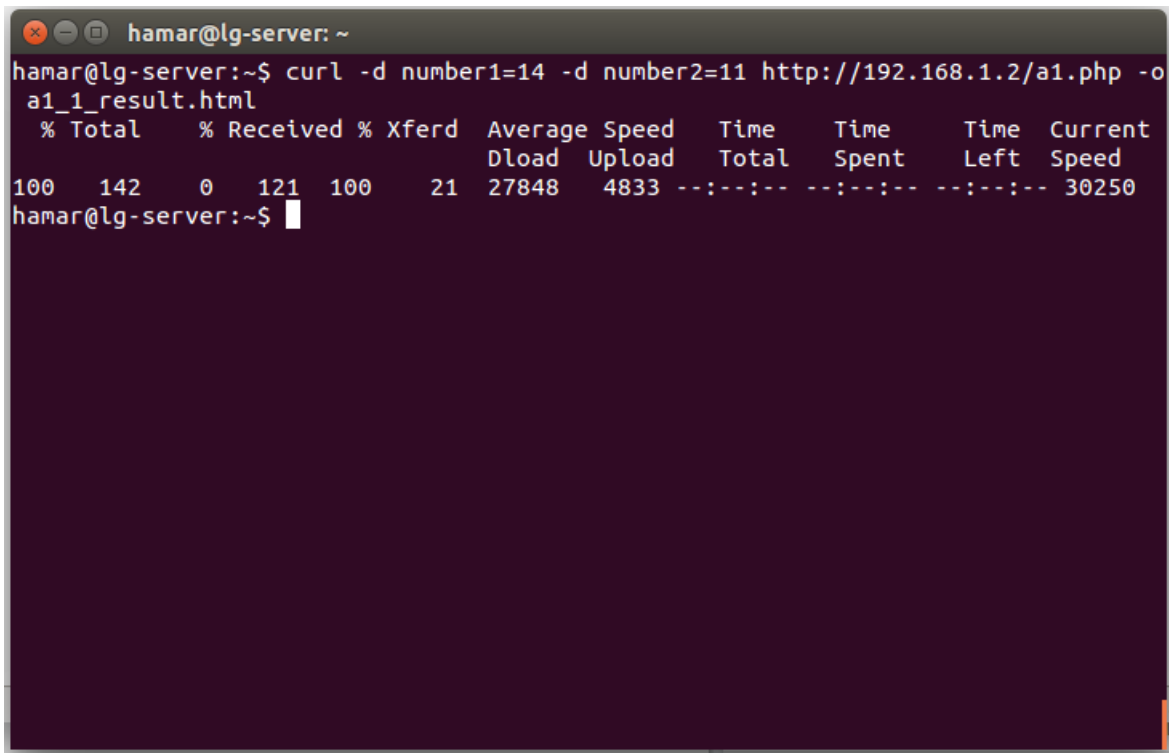
```
.....
1  <html>
2  <head>
3  <title>CS-532 A1 Result</title>
4  </head>
5
6  <body>
7  <?php
8      $x = $_POST['number1'];
9      $y = $_POST['number2'];
10
11      echo 'The multiplication of '$x.' by '$y.' is '$x * $y;
12  ?>
13 </body>
14 </html>
```

If we type from the command line: **curl -d number1=14 -d number2=11 http://192.168.1.2/a1.php -o a1\_1\_result.html**, the -d option will combine the field names *number1* and *number2* similar to GET method which is in the format[1]

*< variable1 >=< data1 > & < variable2 >=< data2 > &...*

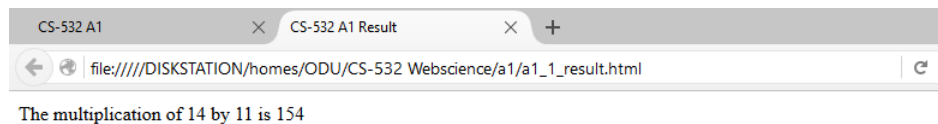
The -o option places the response into a given file-name: *< a1\_1\_result.html >*

### Screen Shot of curl POST usage



```
hamar@lg-server: ~  
hamar@lg-server:~$ curl -d number1=14 -d number2=11 http://192.168.1.2/a1.php -o  
a1_1_result.html  
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
                                 Dload  Upload  Total  Spent    Left  Speed  
100  142    0  121   100    21   27848   4833  --:--:-- --:--:-- --:--:-- 30250  
hamar@lg-server:~$
```

### Screen Shot of saved HTML response



## Problem 2

Write a Python program that:

1. takes as a command line argument a web page
2. extracts all the links from the page
3. lists all the links that result in PDF files, and prints out the bytes for each of the links. (note: be sure to follow all the redirects until the link terminates with a "200 OK".)
4. show that the program works on 3 different URIs, one of which needs to be:  
<http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html>

Python program below (**a1\_v1.1.py**) is attached to this file:

a1.py

```
.....
1  import locale
2  import sys
3  import requests
4  import validators
5  from urllib.parse import urlparse
6  from bs4 import BeautifulSoup
7  from time import strftime, localtime, time
8
9  """
10  This Python program
11  1. takes as a command line argument a web page
12  2. extracts all the links from the page
13  3. lists all the links that result in PDF files, and prints out
14  the bytes for each of the links. (note: be sure to follow
15  all the redirects until the link terminates with a "200 OK".)
16  """
17  __author__ = 'Plinio H. Vargas'
18  __date__ = 'Thu, Jan 21, 2016 at 22:22:11'
19  __email__ = 'pvargas@cs.odu.edu'
20
21
22  def main(url):
23      locale.setlocale(locale.LC_ALL, 'en-US.utf8')
24      # record running time
25      start = time()
26      print('Starting Time: %s' % strftime("%a, %b %d, %Y at %H:%M:%S", localtime()))
27      print('Extracting pdf links from: %s\n' % url)
28
29      # get uri status
```

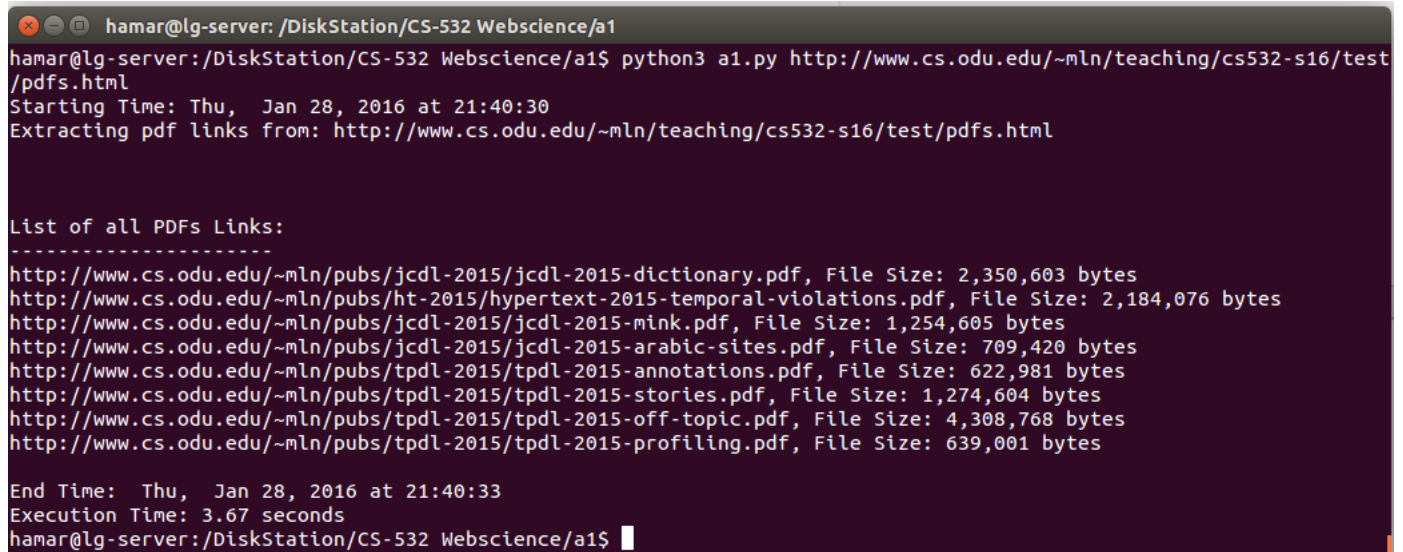
```
30     if requests.get(url).status_code != 200:
31         print('\n\nURI is not available from SERVER. Verify URI.\n')
32         return
33
34     # get source from URI
35     page = requests.get(url).text
36
37     # get parse hostname from URI
38     url = 'http://' + urlparse(url).netloc
39
40     # create BeautifulSoup Object
41     soup = BeautifulSoup(page, 'html.parser')
42
43     # place source link into list
44     all_links = []
45     for link in soup.find_all('a'):
46         uri = link.get('href')
47         # include hostname if url is provided by reference
48         if ((len(uri) > 6 and uri[:7].lower() != 'http://') or len(uri) < 7) and
49             uri[:8].lower() != 'https://':
50             if uri[:2] == '//': # if url has double backslash then url is not provided by reference
51                 uri = 'http:' + uri
52             elif uri[0] != '/': # include backslash if it was not include by reference
53                 uri = url + '/' + uri
54             else:
55                 uri = url + uri
56
57     # for debugging
58     # print(uri)
59
60     try:
61         r = requests.head(uri)
62         if 'Content-Type' in r.headers and r.headers['Content-Type'] == 'application/pdf':
63             if r.status_code == 200:
64                 # ensure server provides Content-Length
65                 try:
66                     all_links.append((uri, r.headers['Content-Length']))
67                 except KeyError:
68                     # make Content-Length unknown
69                     r.headers['Content-Length'] = '???'
70                     all_links.append((uri, r.headers['Content-Length']))
71             elif 'location' in r.headers and (r.status_code == 301 or r.status_code == 302):
72                 counter = 1
73                 while counter < 7:
74                     uri = r.headers['location']
75                     r = requests.head(r.headers['location'])
76                     if 'location' in r.headers and (r.status_code == 301 or r.status_code == 302):
77                         counter += 1
```

```
77         elif r.status_code == 200:
78             # ensure server provides Content-Length
79             try:
80                 all_links.append((uri, r.headers['Content-Length']))
81             except KeyError:
82                 # make Content-Length unknown
83                 r.headers['Content-Length'] = '???'
84                 all_links.append((uri, r.headers['Content-Length']))
85         else:
86             break
87
88     except requests.exceptions.SSLError:
89         print('Couldn\'t open: %s. URL requires authentication.' % uri)
90     except requests.exceptions.ConnectionError:
91         print('Couldn\'t open: %s. Connection refused.' % uri)
92
93     print('\n\nList of all PDFs Links:')
94     print('-' * len('List of all PDFs Links'))
95
96     pdf_links = set(all_links)
97     all_links = list(pdf_links)
98     if len(all_links) > 0:
99         for i in range(len(pdf_links)):
100             if all_links[i][1] == '???':
101                 # don't format Content-Lenght
102                 print('%s, File Size: %s bytes' % (all_links[i][0], all_links[i][1]))
103             else:
104                 print('%s, File Size: %s bytes' % (all_links[i][0],
105                                                     locale.format("%d", int(all_links[i][1]), grouping=True)))
106     else:
107         print('No PDFs links for above URI.')
108
109     print('\nEnd Time: %s' % strftime("%a, %b %d, %Y at %H:%M:%S", localtime()))
110     print('Execution Time: %.2f seconds' % (time()-start))
111     return
112
113 if __name__ == '__main__':
114     # checks for argument
115     if len(sys.argv) != 2:
116         print('Please, provide url\nUsage: python3 a1.py [url]')
117         sys.exit(-1)
118     if not validators.url(sys.argv[1]):
119         print('URL is invalid, please correct url and try again')
120         sys.exit(1)
121
122     # call main
123     main(sys.argv[1])
124
125     sys.exit(0)
```



In the previous version of *a1.py* line **60** had the statement *r = requests.get(uri)*. This was causing object *r* to behave as web-browser deal with re-direction. All URIs were getting STATUS 200 regardless if the URI was getting re-directed. Figure 2 shows all resources pointing to a PDF application, including the ones with STATUS CODE 301 or 302. After correcting line **60** with the statement *r = requests.get(uri)* the code was able to obtain all redirect, but after running code for required URI: url-<http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.htm>, the number of resources appeared to change. See figure 1.

Below is the result of testing *a1.py* on 3 different URIs:



```
hamar@lg-server: /DiskStation/CS-532 Webscience/a1
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$ python3 a1.py http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html
Starting Time: Thu, Jan 28, 2016 at 21:40:30
Extracting pdf links from: http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html

List of all PDFs Links:
-----
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf, File Size: 2,350,603 bytes
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf, File Size: 2,184,076 bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf, File Size: 1,254,605 bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf, File Size: 709,420 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf, File Size: 622,981 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf, File Size: 1,274,604 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf, File Size: 4,308,768 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf, File Size: 639,001 bytes

End Time: Thu, Jan 28, 2016 at 21:40:33
Execution Time: 3.67 seconds
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$
```

Figure 1: Test of *a1.v1.1.py* at <http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html> with updated code

```

hamar@lg-server: /DiskStation/CS-532 Webscience/a1
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$ python3 a1.py http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/
pdfs.html
Starting Time: Fri, Jan 22, 2016 at 15:15:58
Extracting pdf links from: http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html

List of all PDFs Links:
-----
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf, File Size: 2,184,076 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf, File Size: 622,981 bytes
http://arxiv.org/pdf/1512.06195, File Size: 1,748,961 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf, File Size: 4,308,768 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf, File Size: 1,274,604 bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf, File Size: 639,001 bytes
http://bit.ly/1ZDatNK, File Size: 720,476 bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf, File Size: 1,254,605 bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf, File Size: 709,420 bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf, File Size: 2,350,603 bytes

End Time: Fri, Jan 22, 2016 at 15:16:15
Execution Time: 16.92 seconds
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$

```

Figure 2: Test of *a1.py* at <http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html> with original code

```

hamar@lg-server: /DiskStation/CS-532 Webscience/a1
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$ python3 a1.py http://www.cs.odu.edu
Starting Time: Fri, Jan 22, 2016 at 15:53:19
Extracting pdf links from: http://www.cs.odu.edu

Couldn't open: https://sysweb.cs.odu.edu/online/index.php?action=create. URL requires authentication.
Couldn't open: https://sysweb.cs.odu.edu/guests. URL requires authentication.
Couldn't open: https://roundcube.cs.odu.edu/. URL requires authentication.
Couldn't open: https://sysweb.cs.odu.edu/online/index.php?action=create. URL requires authentication.

List of all PDFs Links:
-----
http://www.cs.odu.edu/StrategicPlan0515_2010.pdf, File Size: 909,323 bytes
http://www.cs.odu.edu/files/csintroductioninfo.pdf, File Size: 564,602 bytes
http://www.cs.odu.edu/files/csdeptresearch.pdf, File Size: 2,633,984 bytes
http://www.cs.odu.edu/files/cs_systems_it_infrastructure_2012.pdf, File Size: 2,049,957 bytes
http://www.cs.odu.edu/files/cs_systems_services.pdf, File Size: 412,031 bytes
http://www.cs.odu.edu/studentappointmentinfo.pdf, File Size: 636,560 bytes

End Time: Fri, Jan 22, 2016 at 15:54:09
Execution Time: 50.26 seconds
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$

```

Figure 3: Test of *a1.py* at <http://www.cs.odu.edu/>

```
hamar@lg-server: /DiskStation/CS-532 Webscience/a1
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$ python3 a1.py http://www.vbschools.com/curriculum/gifted
Starting Time: Fri, Jan 22, 2016 at 15:38:30
Extracting pdf links from: http://www.vbschools.com/curriculum/gifted

List of all PDFs Links:
-----
http://www.vbschools.com/calendar/employeeecalendar.asp, File Size: 36,714 bytes
http://www.vbschools.com/students/conduct/content/pdfs/CodeStudentConduct.pdf, File Size: 684,089 bytes
http://www.vbschools.com/online_pubs/gifted_bulletin/, File Size: 1,043,352 bytes
http://www.vbschools.com/curriculum/app_pdfs/GiftedLetterForm.pdf, File Size: 483,948 bytes
http://www.vbschools.com/guidance/content/pdfs/RecordsReleaseFormer.pdf, File Size: 119,105 bytes
http://www.vbschools.com/curriculum/gifted/content/pdfs/LocalPlanEducationGifted.pdf, File Size: 1,828,034 bytes

End Time: Fri, Jan 22, 2016 at 15:39:02
Execution Time: 32.26 seconds
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$
```

Figure 4: Test of *a1.py* at <http://www.vbschools.com/curriculum/gifted/>

## Problem 3

Consider the "bow-tie" graph in the Broder et al. paper (fig 9):

<http://www9.org/w9cdrom/160/160.html>

Now consider the following graph:

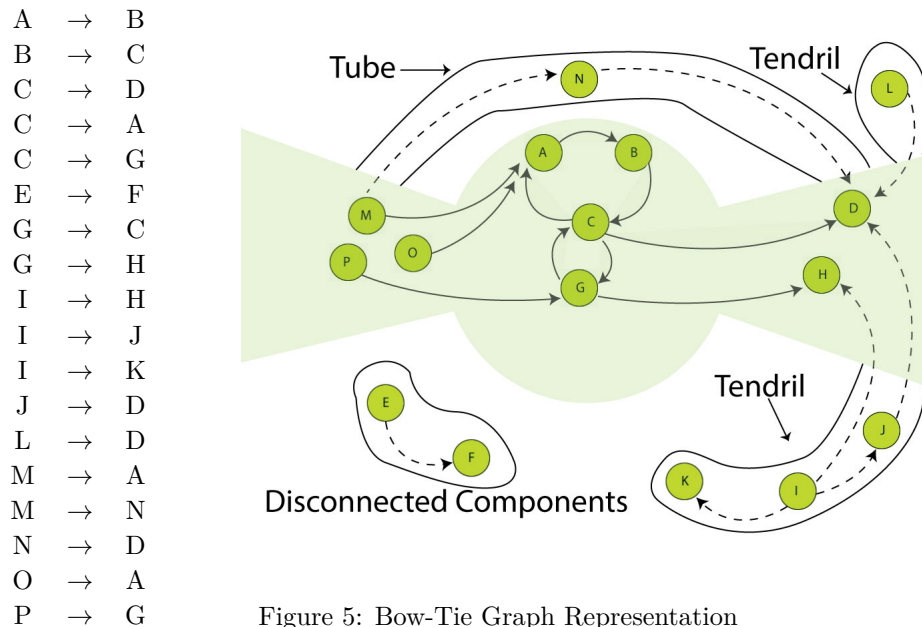


Figure 5: Bow-Tie Graph Representation

For the above graph, give the values for:

- IN:** {M,O,P} “pages that can reach the SCC, but cannot be reached from it”[2]
- SCC:** {A,B,C,G} “central core, all of whose pages can reach one another along directed links – this ”giant strongly connected component” (SCC) is at the heart of the web”. [2]
- OUT:** {D,H} “pages that are accessible from the SCC, but do not link back to it”[2]
- Tendrils:** {I,J,K,L} “ pages that cannot reach the SCC, and cannot be reached from the SCC”[2]
- Tubes:** {N} “passage from a portion of IN to a portion of OUT without touching SCC”[2]
- Disconnected:** {E,F} Everything NOT fitting all criteria above.

## References

- [1] Graph structure in the web. (n.d.) Retrieved January 23, 2016, from <http://curl.haxx.se/docs/manual.html>
- [2] Graph structure in the web. (n.d.) Retrieved January 23, 2016, from <http://http://www9.org/w9cdrom/160/160.html>