CS-532 Web Science: Assignment #1

Due on Thursday, January 28, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

List of Figures

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

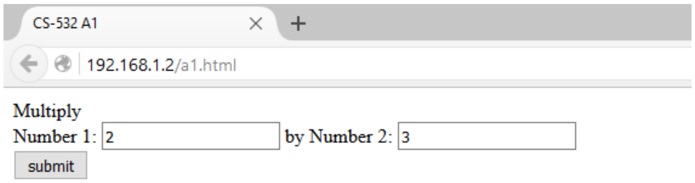
http://curl.haxx.se[1] was an excellent resource to unveil the power of ***curl*** command. The are many options available with *curl*, including passing of cookies, which are intensely used by current servers for Cross-Site Request Forgery (CSRF). Then, in order to simplify our demonstration I created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

a1.html

```
<html>
<head>
<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8” />
<title>CS-532 A1</title>
</head>

<body>
<form method=”post” action=”a1.php”>
<label>Multiply</label><br />
 <label>Number 1:</label>
 <input name=”number1” type=”text” value=”2” />
 <label> by </label>
 <label>Number 2:</label>
 <input name=”number2” type=”text” value= ”3” /> <br />
 <input name=”submit” type=”submit” value=”submit”>
 </form>
 </body>
 </html>
```

Problem 1 continued on next page...

Screen Shot of html page



Multiply
Number 1: 2 by Number 2: 3

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

a1.php

```
<html>
<head>
<title>CS-532 A1 Result</title>
</head>

<body>
<?php
    $x = $_POST[’number1’];
    $y = $_POST[’number2’];

      echo ’The multiplication of ’.$x.’ by ’.$y.’ is ’.$x * $y;
 ?>
 </body>
 </html>
```

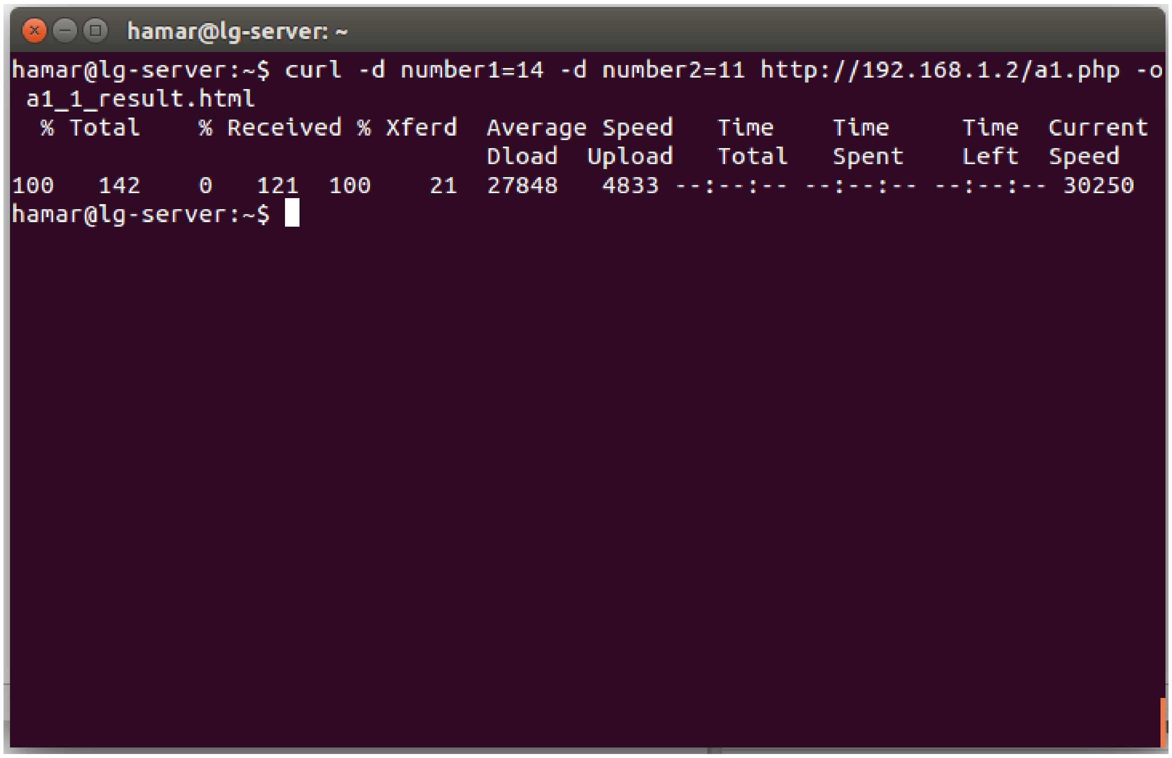
If we type from the command line: **curl -d number1=14 -d number2=11 http://192.168.1.2/a1.php -o a1_1_result.html**, the -d option will combine the field names *number1* and *number2* similar to GET method which is in the format][1]

$$< variable1 >=< data1 > \& < variable2 >=< data2 > \&...$$

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

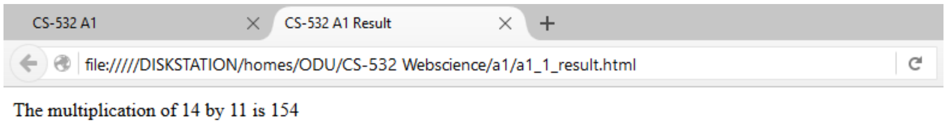
Problem 1 continued on next page. . .

Screen Shot of curl POST usage



```
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



file:////DISKSTATION/homes/ODU/CS-532 Webscience/a1/a1_1_result.html

The multiplication of 14 by 11 is 154

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.py***) is attached to this file:

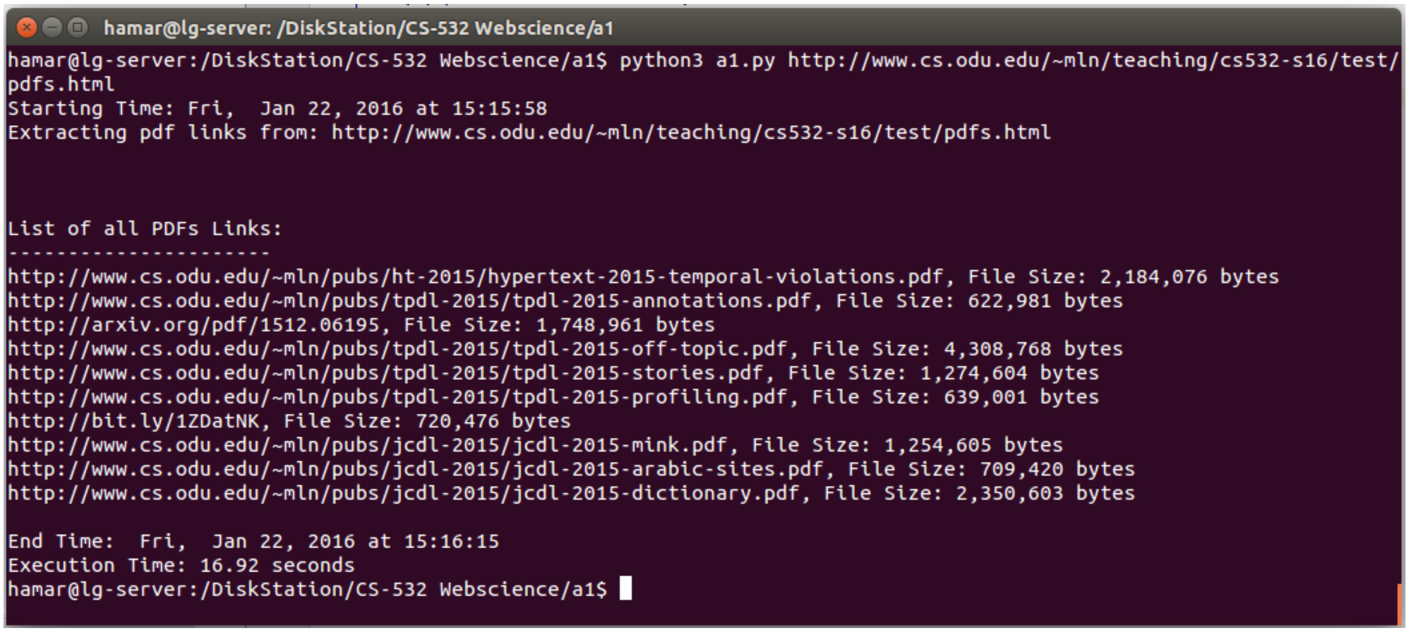
a1.py

```
import locale
import sys
import requests
import validators
from urllib.parse import urlparse
from bs4 import BeautifulSoup
from time import strftime, localtime, time

”””
 This Python program
    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”.)
 ”””
 __author__ = ’Plinio H. Vargas’
 __date__ = ’Thu,   Jan 21, 2016 at 22:22:11’
 __email__ = ’pvargas@cs.odu.edu’


 def main(url):
     locale.setlocale(locale.LC_ALL, ’en_US.utf8’)
     # record running time
     start = time()
     print(’Starting Time: %s’ % strftime(”%a,   %b %d, %Y at %H:%M:%S”, localtime()))
     print(’Extracting pdf links from: %s\n’ % url)

     # get uri status
```

Problem 2 continued on next page. . .

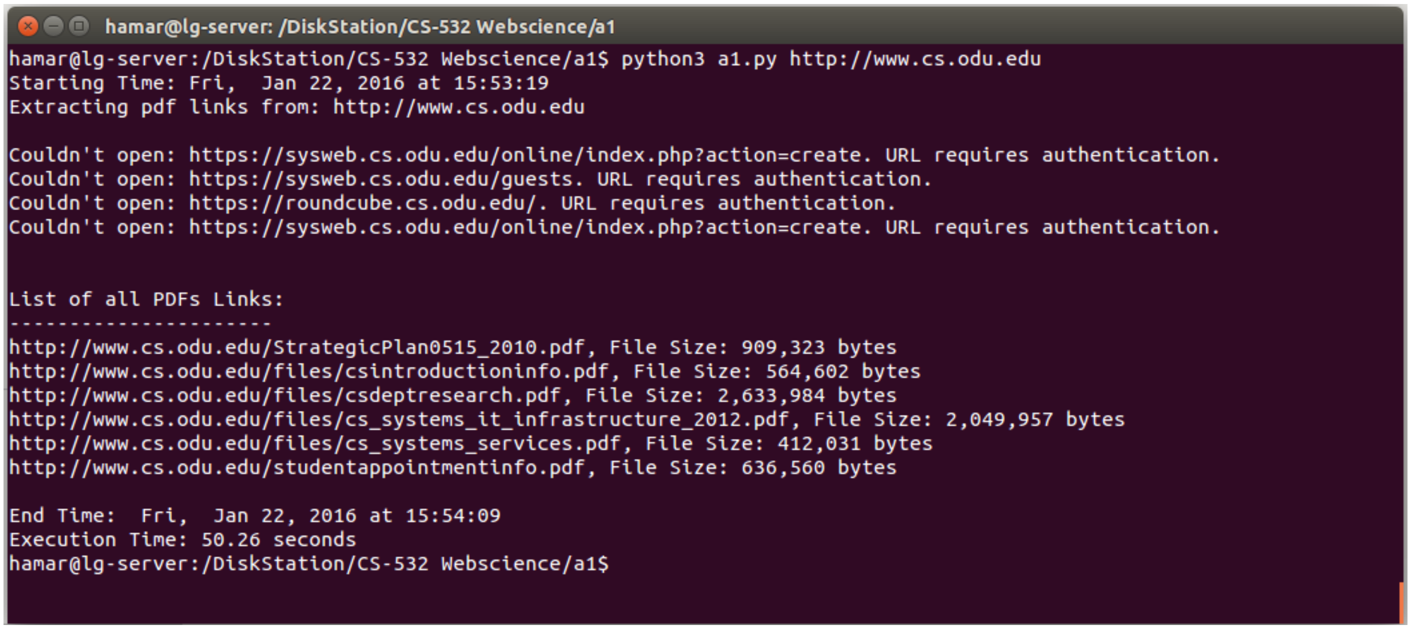
```
    if requests.get(url).status_code != 200:
        print(’\n\nURI is not available from SERVER. Verify URI.\n’)
        return

    # get source from URI
    page = requests.get(url).text

    # get parse hostname from URI
    url = ’http://’ + urlparse(url).netloc

    # create BeautifulSoup Object
    soup = BeautifulSoup(page, ’html.parser’)

    # place source link into list
    all_links = []
    for link in soup.find_all(’a’):
        uri = link.get(’href’)
        # include hostname if url is provided by reference
        if ((len(uri) > 6 and uri[:7].lower() != ’http://’) or len(uri) < 7) and
            uri[:8].lower() != ’https://’:
            if uri[:2] == ’//’:      # if url has double backslash then url is not provided by reference
                uri = ’http:’ + uri
            elif uri[0] != ’/’:      # include backslash if it was not include by reference
                uri = url + ’/’ + uri
            else:
                uri = url + uri

        # for debugging
        # print(uri)

        try:
            r = requests.get(uri)
            if ’content-type’ in r.headers and r.headers[’content-type’] == ’application/pdf’:
                if r.status_code == 200:
                    all_links.append((uri, r.headers[’content-length’]))
        except requests.exceptions.SSLError:
            print(’Couldn\’t open: %s. URL requires authentication.’ % uri)
        except requests.exceptions.ConnectionError:
            print(’Couldn\’t open: %s. Connection refused.’ % uri)

    print(’\n\nList of all PDFs Links:’)
    print(’-’ * len(’List of all PDFs Links’))

    pdf_links = set(all_links)
    all_links = list(pdf_links)
    if len(all_links) > 0:
        for i in range(len(pdf_links)):
            print(’%s, File Size: %s bytes’ % (all_links[i][0],
                                             locale.format(”%d”, int(all_links[i][1]), grouping=True)))
    else:
        print('No PDFs links for above URI.')

    print('\nEnd Time:   %s' % strftime("%a,   %b %d, %Y at %H:%M:%S", localtime()))
    print('Execution Time: %.2f seconds' % (time()-start))
    return

if __name__ == '__main__':
    # checks for argument
    if len(sys.argv) != 2:
        print('Please, provide url\nUsage: python3 a1.py [url]')
        sys.exit(-1)
    if not validators.url(sys.argv[1]):
        print('URL is invalid, please correct url and try again')
        sys.exit(1)

    # call main
    main(sys.argv[1])

    sys.exit(0)
```

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



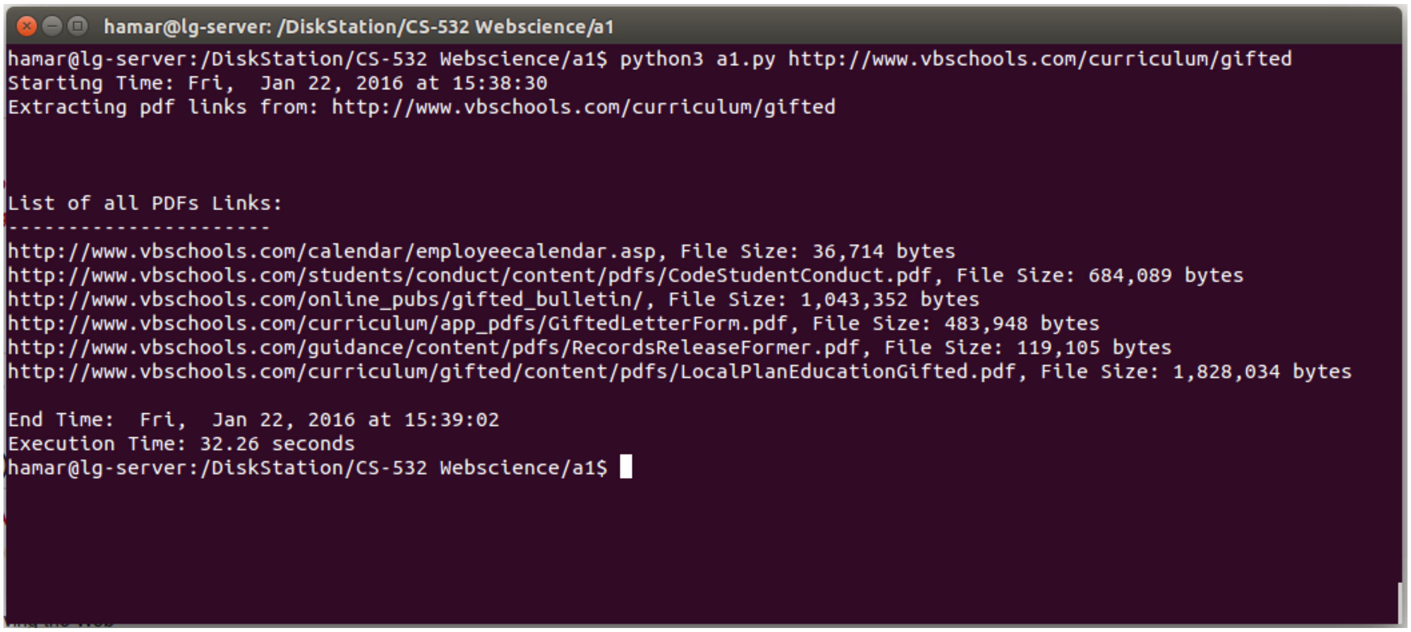
```
hamar@lg-server:/DiskStation/CS-532 Webscience/a1$ python3 a1.py http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/
pdfs.html
Starting Time: Fri,  Jan 22, 2016 at 15:15:58
Extracting pdf links from: http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html



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 1: Test of *a1.py* at `http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html`



```
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 2: Test of *a1.py* at `http://www.cs.odu.edu/`



```
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/employeecalendar.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 3: 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:

A	→	B
B	→	C
C	→	D
C	→	A
C	→	G
E	→	F
G	→	C
G	→	H
I	→	H
I	→	J
I	→	K
J	→	D
L	→	D
M	→	A
M	→	N
N	→	D
O	→	A
P	→	G

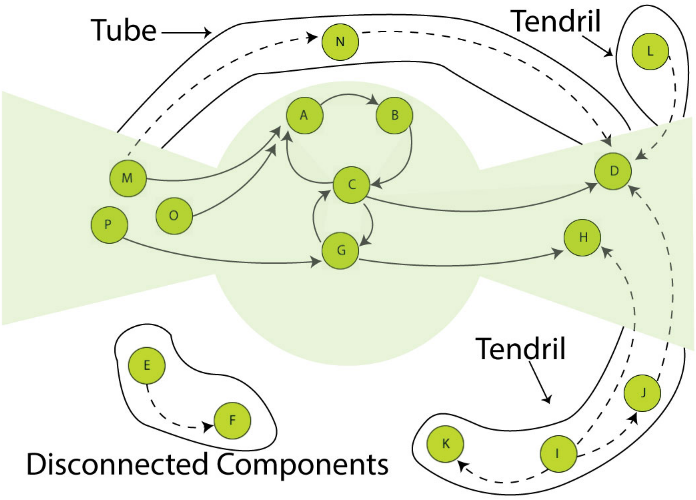


Figure 4: Bow-Tie Graph Representation

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`