Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 1

GeorgeC. Micros UIN: 00757376

Honor Pledge

I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed ________________________________
September 11, 2014

George C. Micros

Written Assignment 1

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

September 11, 2014

Contents

Chapter 1
Written Assignment 1

1.1 Question 1

1.1.1 The Question

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.

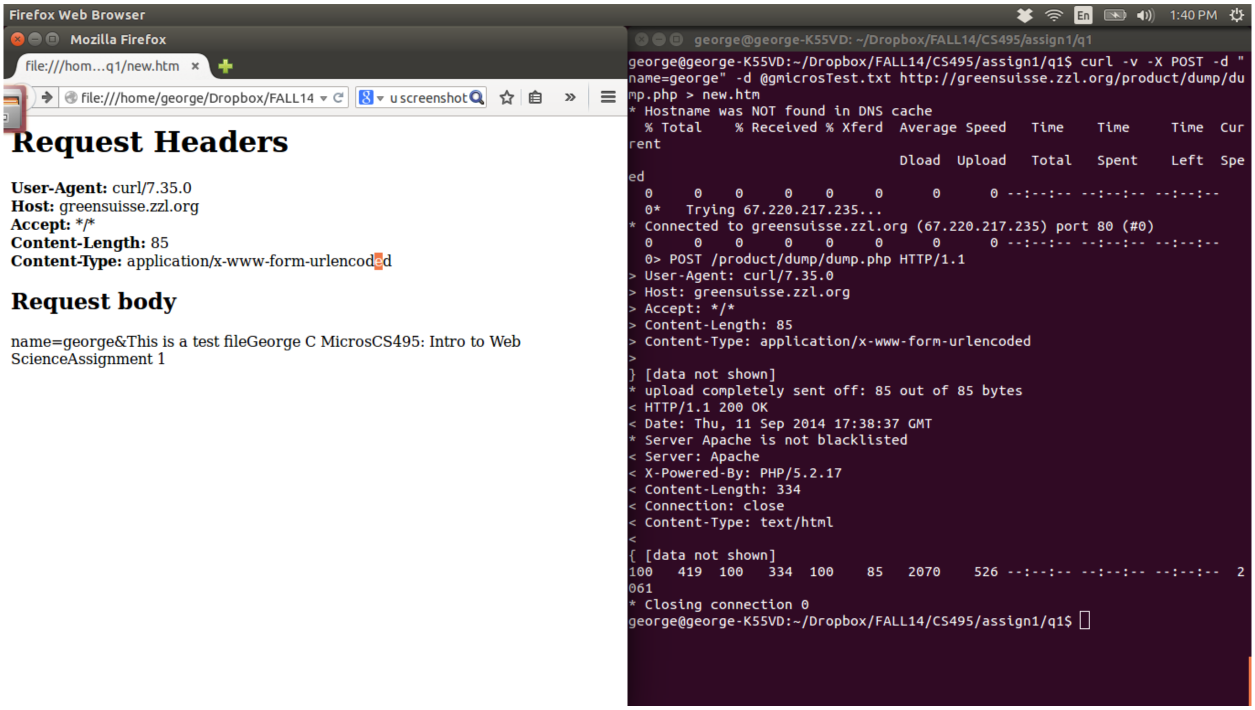
1.1.2 The Answer

```
#! /bin/bash
rm -f *.htm
#curl -v -X POST -d ”name=george” -d @gmicrosTest.txt http://www.posttestserver.com/post.php > postTest.htm
#curl -v -X POST -d ”name=george” -d @gmicrosTest.txt http://httpbin.org/post > httpBin.htm
curl -v -X POST -d "name=george" -d @gmicrosTest.txt http://greensuisse.zzl.org/product/dump/dump.php > green.htm
```

Listing 1: Bash script to call cURL

```
This is a test file
George C Micros
CS495: Intro to Web Science
Assignment 1
```

Listing 2: Test file to post on server


Firefox Web Browser
Mozilla Firefox
file:///hom...q1/new.htm
file:///home/george/Dropbox/FALL14
u screenshot
En
1:40 PM
george@george-K55VD: ~/Dropbox/FALL14/CS495/assign1/q1

Request Headers

User-Agent: curl/7.35.0
Host: greensuisse.zzl.org
Accept: */*
Content-Length: 85
Content-Type: application/x-www-form-urlencoded

Request body

name=george&This is a test fileGeorge C MicrosCS495: Intro to Web ScienceAssignment 1

```
george@george-K55VD:~/Dropbox/FALL14/CS495/assign1/q1$ curl -v -X POST -d "
name=george" -d @gmicrosTest.txt http://greensuisse.zzl.org/product/dump/du
mp.php > new.htm
* Hostname was NOT found in DNS cache
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Cur
rent
                                 Dload  Upload   Total   Spent    Left  Spe
ed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0*   Trying 67.220.217.235...
* Connected to greensuisse.zzl.org (67.220.217.235) port 80 (#0)
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0> POST /product/dump/dump.php HTTP/1.1
> User-Agent: curl/7.35.0
> Host: greensuisse.zzl.org
> Accept: */*
> Content-Length: 85
> Content-Type: application/x-www-form-urlencoded
>
} [data not shown]
* upload completely sent off: 85 out of 85 bytes
< HTTP/1.1 200 OK
< Date: Thu, 11 Sep 2014 17:38:37 GMT
* Server Apache is not blacklisted
< Server: Apache
< X-Powered-By: PHP/5.2.17
< Content-Length: 334
< Connection: close
< Content-Type: text/html
<
{ [data not shown]
100   419  100   334  100    85   2070    526 --:--:-- --:--:-- --:--:--  2
061
* Closing connection 0
george@george-K55VD:~/Dropbox/FALL14/CS495/assign1/q1$ 
```

Fig. 1.1: Screenshot of curl request and return

1.2 Question 2

1.2.1 The Question

Write a Python program that:

- takes one argument, like ”Old Dominion” or ”Virginia Tech”
- takes another argument specified in seconds (e.g., ”60” for one minute).
- takes a URI as a third argument:
  `http://sports.yahoo.com/college-football/scoreboard/`
  or
  `http://sports.yahoo.com/college-football/scoreboard/?week=2&conf=all`
  or
  `http://sports.yahoo.com/college-football/scoreboard/?week=1&conf=72`
  etc.
- dereferences the URI, finds the game corresponding to the team argument, prints out the current score (e.g., ”Old Dominion 27, East Carolina 17), sleeps for the specified seconds, and then repeats (until control-C is hit).

1.2.2 The Answer

The beautiful soup library converts the complex HTML file tree to a python object tree that contains methods that permit the easy traversal and search of the tree. Initially the structure site of interest was understood by inspecting the page in a web client, i.e. Chrome. This way the specific tags of interest can be easily known without exploring the entire tree in python. Then by targeting these tags we can find all of their instances and find the one that matched our search. Therefore all the away team tags are searched for the team of interest, then all the home teams. Once the tag of interest is found we can use it as a frame of reference to find the other items we need.

Of course there is a condition for the possibility that the desired team was not available. Assuming it was located its sibling and parents can be used to easy find the location of the opponent team and score without searching the entire tree again. The time library allows for a sleep to be inserted at any point in the code.

```bash
#! /bin/bash

./urlCode.py "Old Dominion" 5 "http://sports.yahoo.com/college-football/scoreboard/?week=2&conf=all"
```

Listing 3: Command line input

```python
#! /usr/bin/python
from bs4 import BeautifulSoup
import bs4
import urllib
import urllib2
import sys
import re
import time
from datetime import datetime

def oops():
 print "Didnt find it";
 exit()

print "URI program"
team = sys.argv[1];
sec = float(sys.argv[2]);
site = sys.argv[3];

while(1):
```

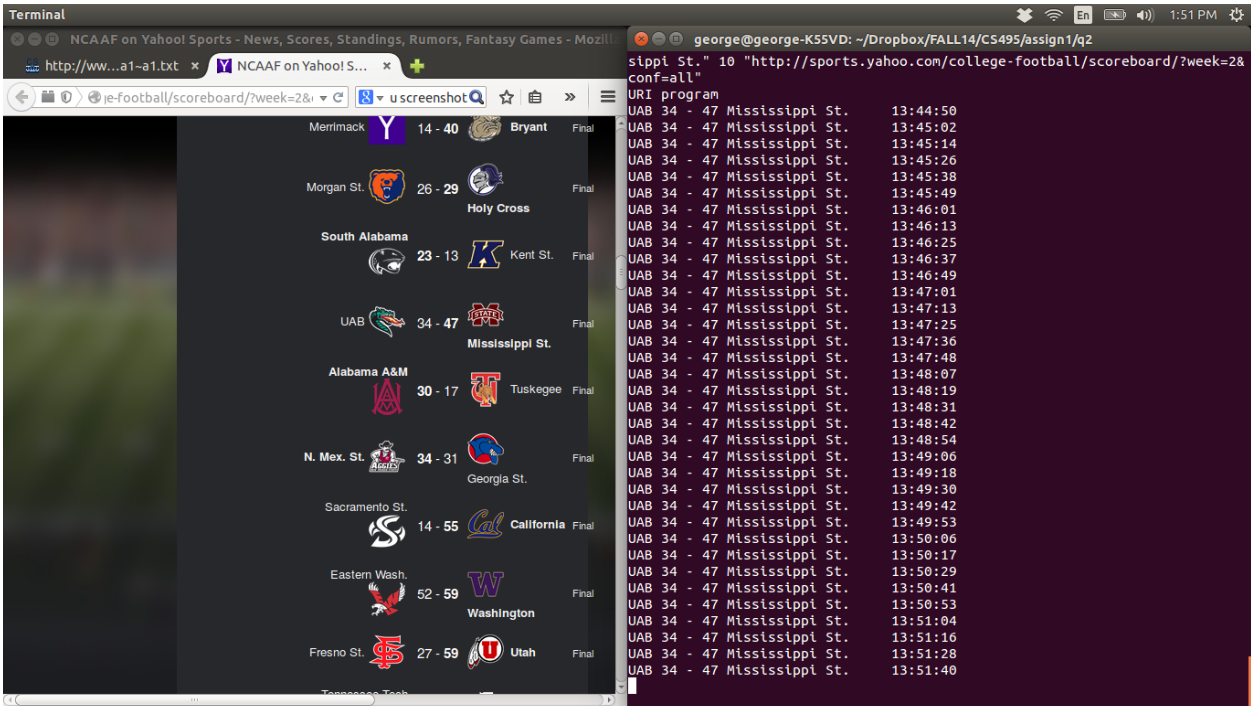
```
# fetching the page from the internet
request = urllib2.Request(site);
file = urllib2.urlopen(request);
# getting the soup from the html
html = file.read();
soup = BeautifulSoup(html);

team1 = [];
reverse = 0;
# check the away tsd first
tds = soup.find_all("td", "away");
for td in tds:
 for kids in td.children:
  if isinstance(kids, bs4.element.Tag):
   for kid in kids.children:
    if isinstance(kid, bs4.element.Tag) and kid.string == team:
     there = kid;
     team1 = there.string;
# the team was found in aways
if team1 != []:
 # get parent tag
 away = there.parent.parent
 # hop over to score
 score =  away.next_sibling.next_sibling
 # weird cntr to store values in list
 cnt = 0
 for kids in score.h4.a.children:
  if isinstance(kids, bs4.element.Tag):
   score[cnt] = kids.string
   cnt = cnt +1;
 # hop over to home
 home = score.next_sibling.next_sibling;
 for kids in home.children:
  if isinstance(kids, bs4.element.Tag):
   for kid in kids.children:
    if isinstance(kid, bs4.element.Tag) and kid.string != None:
     team2 = kid.string;

# check for team in home
else:
 reverse = 1
 tds = soup.find_all("td", "home");
 for td in tds:
  for kids in td.children:
   if isinstance(kids, bs4.element.Tag):
    for kid in kids.children:
     if isinstance(kid, bs4.element.Tag) and kid.string == team:
      there = kid;
      team1 = there.string;
 if team1 == []:
  oops();
 else:
  home = there.parent.parent
  score = home.previous_sibling.previous_sibling;
  cnt = 1;
  for kids in score.h4.a.children:
   if isinstance(kids, bs4.element.Tag):
    score[cnt] = kids.string;
    cnt = cnt - 1;
  away = score.previous_sibling.previous_sibling;
  for kids in away.children:
   if isinstance(kids, bs4.element.Tag):
    for kid in kids.children:
     if isinstance(kid, bs4.element.Tag) and kid.string != None:
      team2 = kid.string;
if team1 != [] and team !=[]:
 if reverse == 0:
  print team1, score[0],"-", score[1], team2,"\t", datetime.now().strftime('%H:%M:%S')
 elif reverse == 1:
  print team2, score[1],"-", score[0], team1, "\t", datetime.now().strftime('%H:%M:%S')
 else:
  print "Something went wrong"
else:
                                   print "Sorry cant find that team. check your spelling";i

time.sleep(sec);
```

Listing 4: Python code implementing the score search



Terminal
NCAAF on Yahoo! Sports - News, Scores, Standings, Rumors, Fantasy Games - Mozill
http://ww...a1~a1.txt
NCAAF on Yahoo! S...
e-football/scoreboard/?week=2&
u screenshot
Merrimack 14 - 40 Bryant Final
Morgan St. 26 - 29 Holy Cross Final
South Alabama 23 - 13 Kent St. Final
UAB 34 - 47 Mississippi St. Final
Alabama A&M 30 - 17 Tuskegee Final
N. Mex. St. 34 - 31 Georgia St. Final
Sacramento St. 14 - 55 California Final
Eastern Wash. 52 - 59 Washington Final
Fresno St. 27 - 59 Utah Final
1:51 PM
george@george-K55VD: ~/Dropbox/FALL14/CS495/assign1/q2

```
sippi St." 10 "http://sports.yahoo.com/college-football/scoreboard/?week=2&
conf=all"
URI program
UAB 34 - 47 Mississippi St.     13:44:50
UAB 34 - 47 Mississippi St.     13:45:02
UAB 34 - 47 Mississippi St.     13:45:14
UAB 34 - 47 Mississippi St.     13:45:26
UAB 34 - 47 Mississippi St.     13:45:38
UAB 34 - 47 Mississippi St.     13:45:49
UAB 34 - 47 Mississippi St.     13:46:01
UAB 34 - 47 Mississippi St.     13:46:13
UAB 34 - 47 Mississippi St.     13:46:25
UAB 34 - 47 Mississippi St.     13:46:37
UAB 34 - 47 Mississippi St.     13:46:49
UAB 34 - 47 Mississippi St.     13:47:01
UAB 34 - 47 Mississippi St.     13:47:13
UAB 34 - 47 Mississippi St.     13:47:25
UAB 34 - 47 Mississippi St.     13:47:36
UAB 34 - 47 Mississippi St.     13:47:48
UAB 34 - 47 Mississippi St.     13:48:07
UAB 34 - 47 Mississippi St.     13:48:19
UAB 34 - 47 Mississippi St.     13:48:31
UAB 34 - 47 Mississippi St.     13:48:42
UAB 34 - 47 Mississippi St.     13:48:54
UAB 34 - 47 Mississippi St.     13:49:06
UAB 34 - 47 Mississippi St.     13:49:18
UAB 34 - 47 Mississippi St.     13:49:30
UAB 34 - 47 Mississippi St.     13:49:42
UAB 34 - 47 Mississippi St.     13:49:53
UAB 34 - 47 Mississippi St.     13:50:06
UAB 34 - 47 Mississippi St.     13:50:17
UAB 34 - 47 Mississippi St.     13:50:29
UAB 34 - 47 Mississippi St.     13:50:41
UAB 34 - 47 Mississippi St.     13:50:53
UAB 34 - 47 Mississippi St.     13:51:04
UAB 34 - 47 Mississippi St.     13:51:16
UAB 34 - 47 Mississippi St.     13:51:28
UAB 34 - 47 Mississippi St.     13:51:40
```

Fig. 1.2: Screenshot of the script execution

1.3 Question 3

1.3.1 The Question

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

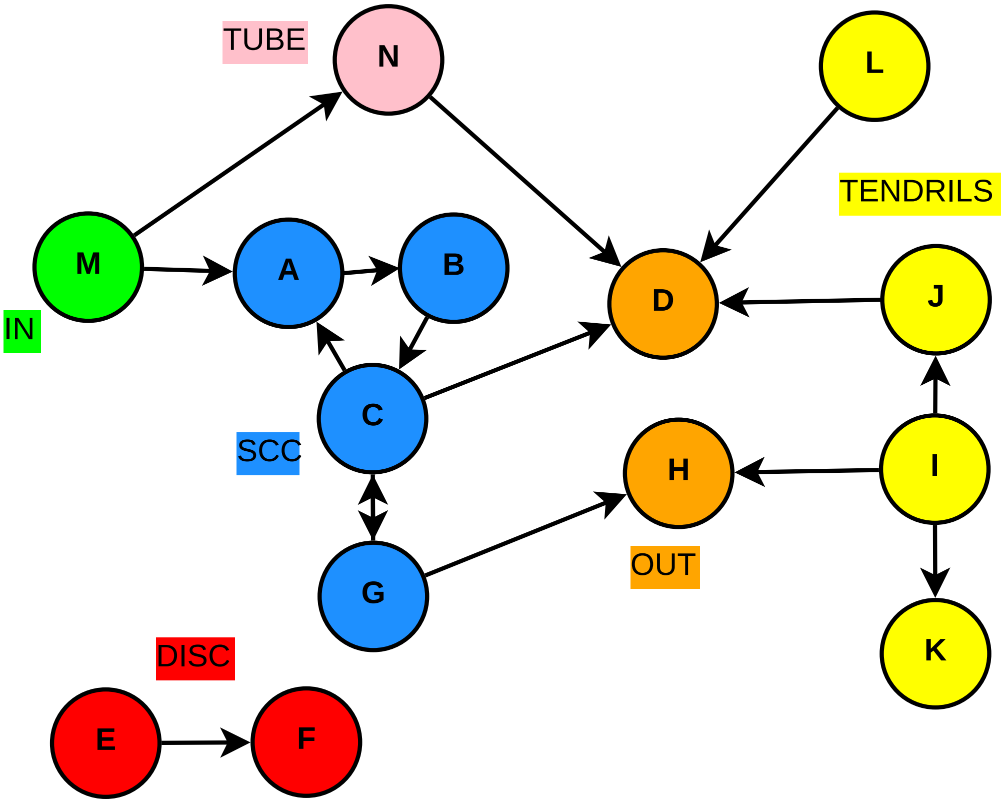
M → N

N → D

For the above graph, give the values for: IN, SCC, OUT, Tendrils, Tubes, Disconnected

1.3.2 The Answer

The most critical part of analyzing a graph and identifying the constituent components is to find the strongly connected component. This can be done visually or heuristically in this case. However, in the case of complex and large networks this is not an efficient method. Thankfully algorithms have been developed to solve this problem and they are implemented in python libraries. This way we can verify the result we obtain manually.



```python
#! /usr/bin/python

# script that imports graph data and finds the strongly connected groups

from collections import defaultdict
import numpy as np
from scipy.sparse import csgraph


print "Graph Script\n"
f = open("graph.txt", 'r');
graph = defaultdict(list);
array = np.zeros((14,14))

for line in f:
 line = line.replace(" ", "");
 print line
 line = line.replace("-->","");
 key = line[0];
 value = line[1];
 i = ord(key) - ord('A');
 j = ord(value) - ord('A')
 array[i][j] = 1;
 graph[key].append(value);
f.close();

graph = sorted(graph.items())
comp, comp_list =  csgraph.connected_components(array, True, "strong")
a =  [list((np.where(comp_list == i))) for i in range(comp)]

final = [];
for i in range(len(a)):
  b =  (a[i][0].tolist());
  out = []
  for j in range(len(b)):
    out.append( chr(b[j] + ord('A')))
  final.append(out)

print "The strongly connected groups are:"
print final
```

Listing 5: Code Listing the relevance file corrector

References

1. http://www.google.com
2. http://jakevdp.github.io/blog/2012/10/14/scipy-sparse-graph-module-word-ladders/
3. http://curl.haxx.se/docs/httpscripting.html
4. http://www.crummy.com/software/BeautifulSoup/bs4/doc/
5. http://www.rmi.net/~lutz/
6. http://www.cs.cornell.edu/home/kleinber/networks-book/