

HW 1 - Web Science Intro

Jenah Parman
September 15, 2024

Q1

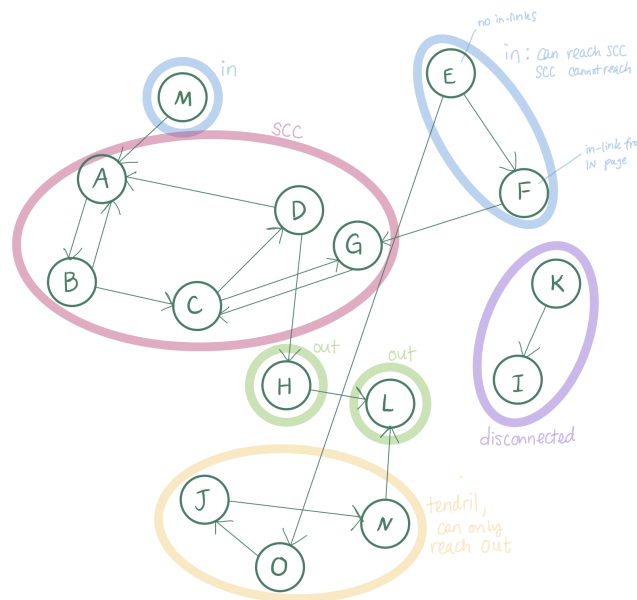


Figure 1: Directed Graph of Node Relations

In this directed graph, the nodes are categorized based on their connectivity:

- **Strongly Connected Component (SCC):** Nodes A, B, C, D, and G form the SCC. This component is characterized by the fact that every node is reachable from every other node in the group, creating a cycle.
- **IN Component:** Nodes M, E, and F belong to the IN component. These nodes can reach the SCC but are not reachable from it, indicating they only have outgoing paths to the SCC.
- **OUT Component:** Nodes H and L are in the OUT component. They can be reached from the SCC but cannot return to it, indicating they are endpoints of paths originating from the SCC.
- **Tendrils:** Nodes J, N, and O are classified as tendrils. Tendrils are connected to nodes in the OUT component but do not directly connect to the SCC.
- **Disconnected Component:** Nodes K and I are completely disconnected from the rest of the graph, meaning they have no direct or indirect path to or from the SCC or other components.

Q2

a) First, load the webpage at the URI in your web browser. The result should show the "User-Agent" HTTP request header that your web browser sends to the web server. Take a screenshot to include in your report.



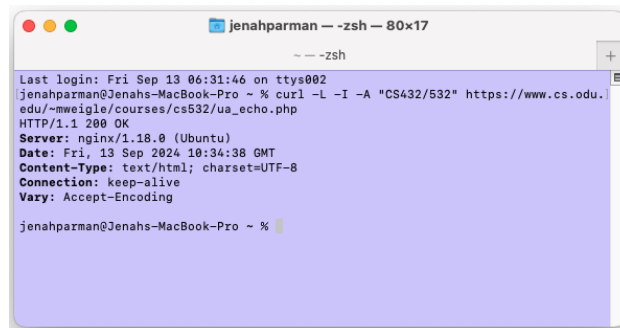
USER AGENT ECHO
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36

Figure 2: Screenshot of webpage showing "User-Agent" HTTP request header

b) Use a single curl command with the appropriate options to do the following:

- request the URI
- show the HTTP response headers
- follow any redirects
- change the User-Agent HTTP request field to "CS432/532"

Take a screenshot of the curl command and options you used and the result of your execution to include in your report.



```
jenahparman ~ -zsh - 80x17
~ -zsh
Last login: Fri Sep 13 06:31:46 on ttys002
jenahparman@Jenahs-MacBook-Pro ~ % curl -L -I -A "CS432/532" https://www.cs.odu.edu/~mweigle/courses/cs532/ua_echo.php
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Fri, 13 Sep 2024 10:34:38 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
jenahparman@Jenahs-MacBook-Pro ~ %
```

Figure 3: Screenshot of curl comand and result

c) Use a single curl command with the appropriate options to do the following:

- request the URI
- follow any redirects
- change the User-Agent HTTP request field to "CS432/532"
- save the HTML output to a file

Take a screenshot of the curl command and options you used and the result of your execution to include in your report.

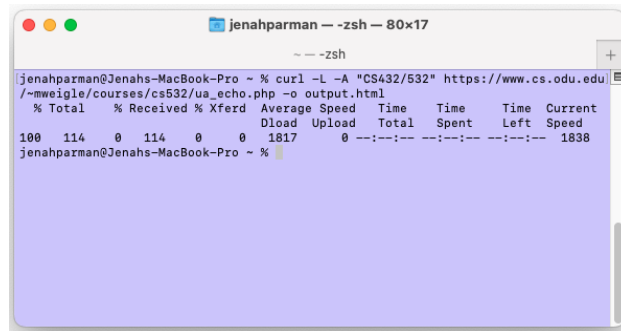


Figure 4: Screenshot of curl command and result of execution

d) View the HTML output file that was produced by curl from part c in a web browser and take a screenshot to include in your report.

```

USER AGENT ECHO
User-Agent: CS432/532

```

Figure 5: Screenshot of HTML output file that was produced by curl from part c in web browser

Q3

```

1 import requests
2 from bs4 import BeautifulSoup
3
4 seed_uri = "https://en.wikipedia.org/wiki/2024
   _United_States_presidential_election" # Starting URL
5 unique_uris = set()
6 to_crawl = [seed_uri] # Start with the initial seed URL
7
8 def get_links(uri):
9     try:
10         response = requests.get(uri, timeout=5)
11         if 'text/html' in response.headers.get('Content-Type', ''):
12             soup = BeautifulSoup(response.text, 'html.parser')
13             return [a['href'] for a in soup.find_all('a', href=True)]

```

```
14     except requests.exceptions.RequestException:
15         return []
16     return []
17
18 def check_uri(uri):
19     try:
20         response = requests.get(uri, timeout=5)
21         if 'text/html' in response.headers.get('Content-Type', ''):
22             content_length = response.headers.get('Content-Length')
23             if content_length and int(content_length) > 1000:
24                 return response.url
25             elif len(response.content) > 1000:
26                 return response.url
27     except requests.exceptions.RequestException:
28         return None
29     return None
30
31 # Keep collecting links until we hit 500 unique ones
32 while len(unique_uris) < 500 and to_crawl:
33     current_seed = to_crawl.pop(0) # Take the next URL to crawl
34     seed_links = get_links(current_seed)
35     for link in seed_links:
36         if not link.startswith("http"):
37             link = "https://en.wikipedia.org" + link # Adjust for
relative links
38         final_uri = check_uri(link)
39         if final_uri and final_uri not in unique_uris:
40             unique_uris.add(final_uri)
41             print(f"Collected: {len(unique_uris)} - {final_uri}")
42             to_crawl.append(final_uri) # Add new valid link to the
crawl list
43         if len(unique_uris) >= 500:
44             break
45
46 # Save the collected URIs to a file
47 with open('collected_uris.txt', 'w') as file:
48     for uri in unique_uris:
49         file.write(uri + '\n')
50
51 print("Collected URIs saved to collected_uris.txt")
```

The Python program uses the second method (have your program randomly pick a URI that you've collected and use that as the new seed until you've collected 500 unique URIs) to collect 500 unique URIs from webpages. It starts with a seed URL, extracts all links from the page using BeautifulSoup, and checks if each link points to an HTML page with a content size greater than 1000 bytes. The program automatically selects new links from the collected URIs and continues crawling deeper into new pages until it reaches 500 unique URIs. This process is automated,

requiring no manual intervention after the initial seed, making it an efficient way to gather the required data.

References

- Overleaf, Code Listing https://www.overleaf.com/learn/latex/Code_listing
- Python Requests, <https://docs.python-requests.org/en/latest/user/quickstart/>
- Beautiful Soup Documentation, <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>