CS51 Final project – Checkpoint 2
Cindy Liu, Carolina Nobre, Y. Jeanette Park
4/28/2013

<u>Checkpoint 2 Writeup</u>

Progress:

Over the past week we:

1) Finished coding all the functions in the Fibonacci heap. As mentioned in last week's writeup, The fibheap.pop() function is a pivotal part of the heap in that it allows us to quickly pick the highest-priority element from our heap, and is the key function needed for Dijkstra's algorithm. The pop function written last week required debugging so considerable time was spent in ensuring it was working properly for numerous test cases.
2) Implemented Dijkstra's algorithm using the fibheap for our graph.
3) Created a front end (connecting-urls.py) that calls all the necessary steps from crawling a set of web pages to returning the shortest path between the starting page and a given url.

All the code seems to be fully debugged so while we will continue to test it with new sets of pages, we do not anticipate finding any major bugs.

The final set of code can be found on our repo: https://github.com/cindyliu/e250-final-project.

In order to test the code, you only need to call the front-end .py file (connecting-urls.py). followed by how many "layers" deep you want to search, and the starting url. Here is an example of the proper usage:

```
connecting-urls.py -d 3 "http:///en.wikipedia.org/wiki/peafowl"
```

The code will then ask you for a destination url. Instead of typing in the entire destination, you can consult the output of the webcrawler (found in options_for_destination_urls.txt) and give it the number of the desired destination url.

Problems:

The only problem we ran into this week was finding the bug in a few of the Fibheap functions, particularly pop. The issue ended up being a combination several different types of bugs, including mistakes in the python syntax (reminder that we all learned Python from scratch for this project), logical flow issues, and problems with pointers.

Teamwork:

Working together on the project has been successful with the caveat that not all members of the group have the same time availability to spend on Harvard related work. Carolina works full time and can only spend evenings and weekends on Harvard while Cindy and Jeanette have been able to dedicate more hours to the project during the day. In order to avoid an imbalance in the workload division, Carolina will be responsible for a larger portion of the extensions planned for the project.

Plan:

At this point we are working on polishing the core code, and adding interesting extensions to the project. For this upcoming week we plan on:

- Use a data visualization algorithm (such as NetworkX or Ubigraph) to create a visual representation of the mapped links along with the shortest path between two given links. (Carolina)
- Implement a second extension where we apply the Fibheap and Dijkstra's to another type of network such as a social network. For example, we are considering implementing a "social web crawler" which identifies how a given person is connected to another person on Facebook. However, it looks like we aren't able to access a given person's friends unless we have explicit information from each person whose friends list we crawl. This will likely prove to be a logistical challenge – hence, <u>we would welcome ideas and guidance from the TF on what other networks are publicly accessible and that we can crawl to feed data into our fibheap/ crawler</u> (Cindy and Jeanette)