CS311, PA2 Report

Connor Ruggles

April 5th, 2017

The data structures I used for Q and *visited* were LinkedList and a HashSet, respectively. I chose the LinkedList because they are easily used as queues, because you can add and remove elements specifically to and from the beginning/end. I chose the HashSet because it is an efficient implementation of a Set, which is all that is needed when checking 'visited' vertices.

- Number of edges and vertices in the graph
  - Edges: 23,942
  - Vertices: 500
- Vertex with largest out degree: /wiki/Computer_science
- Number of strongly connected components: 1
- Size of the largest component: 500

The data structures that I built and used in GraphProcessor were largely made up of HashMaps. How I made the graph was by making a type WebGraph, that holds a HashMap<String, Vertex>, Vertex being another type that I made. It holds a HashMap<String, Edge>, a String with the name of the Vertex, and Edge being another type I made. Edge holds a 'to' Vertex and a 'from' Vertex. All three of those types are private and held in GraphProcessor.java, since that is the only file that uses them. I chose HashMaps for those structures because as we have discussed before this semester, hashing is a fast way to map values to objects, and that is what is needed since vertices would be accessed often.
For storing finish times with SCC, I just used an array of type Vertex, and the index is that Vertex's finish time, starting at 1. For actually storing the SCCs, I used an ArrayList<ArrayList<String>>, because ArrayLists are an efficient implementation of an array.

Asymptotic run times for public methods of GraphProcessor:

- outDegree(String v): *O(1)*
- sameComponent(String u, String v): *O(n)*, n being the number of SCCs in the graph
- componentVertices(String v): *O(n)*, n being the number of SCCs in the graph
- largestComponent(): *O(n)*, n being the number of SCCs in the graph
- numComponents(): *O(1)*
- bfsPath(String u, String v): *O(V+E)*, where V is the number of vertices in the graph, and E is the number of edges