Amara Naas (anaas)
CS595 — Fall 2013

Assignment 7

11/9/2013

Problem 1

In my approach to solve this problem I used the data from `http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat` as specified in assignment 7. In order to use D3 as one of the assignment requirement I must convert the data from the url source to either JSON or CSV file since D3 can only handle these file type. In the file called *RScript.r*[1]:

- I imported the data from `http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat` and assigned it to "url" variable.

  ```
  url <- "http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/zachary.
  dat"
  ```

- By using the utilities in R I got the graph in pajek format and assigned it to a *graph* variable and than create a file called *table.txt*[1].

  ```
  graph <- graph.adjacency(l2, weighted=TRUE, mode="undirected")
  write.graph(graph, "table.txt", format=c("pajek"))
  ```

- I got the data needed from the table.txt file and add the names to the columns of the table so that I can save it as *JSON* file format.

- I created a file in *JSON* format using the data before the graph splitting and called it *databefore.json*[1] file.

- I split the graph based on the maximum edge betweenness as defined by Girvan-Newman Algorithm. The function *edge.betweenness* in R calculates the edge betweenness as described in the following equation:

$$E_B(e) = \sum_{s \neq e \neq t} g_{s_t}(e)/g_{s_t} \tag{1}$$

where as:

$g_{st}(e)$ is total number of shortest paths from node s to node t.

g_{st} is the number of those paths that pass through e.

```
loop while (graph cluster is one)
    find max edge betweenness
    if (max equal edges betweenness of an edge)
        delete the edges from the graph
```

[1] File uploaded to github

- I save the data of the new created graph in the file called *table1.txt*[1] so I can convert this data and save it a *JSON* file format.

  ```
  write.graph(graph, "table1.txt", format=c("pajek"))
  ```

- I created another file in *JSON* format using the data of the graph after the split and called it *dataAfter.json*[1] which contains the group member ship for each node.

I fed these files [*databefore.json* and *dataAfter.json*] to the D3 java script source which called *index.html*[1] in order to create a graph of the Karate club before and after the split. I built the java script source on top of the source from `http://bl.ocks.org/mbostock/4062045` and I got the *dragstart* function used in my source from `http://bl.ocks.org/mbostock/3750558`. Also I noticed that the code will not work if the source indexing starts from one, so I change the source indexing to start from zero. The D3 source reads the data from the *JSON* files and creates nodes and links. The first graph can be created after opening the *index.html* file on the web browser and load the *databefore.json* as shown in Figure 1, and by clicking on the button on the browser the *dataAfter.json* file will be loaded and the graph will be split based on the (1) as shown in Figure 2. The *index.html* file can be tested at `http://www.cs.odu.edu/~anaas/`.

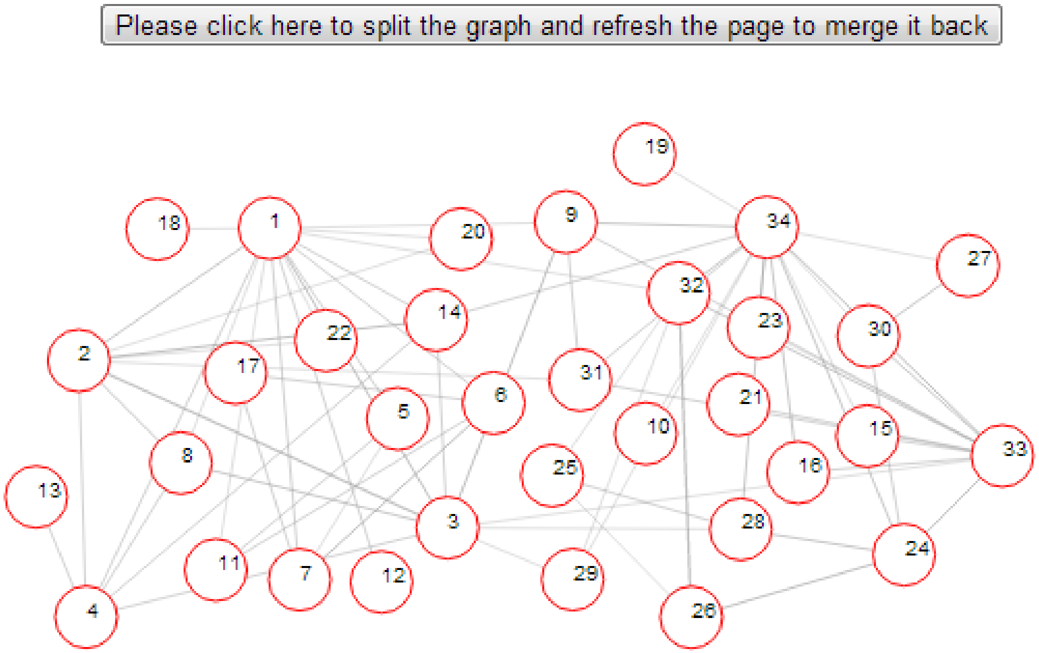
Please click here to split the graph and refresh the page to merge it back
19
18
1
20
9
34
27
32
14
23
30
22
2
17
31
6
5
21
15
33
10
8
25
16
13
3
28
24
11
7
12
29
26
4

Figure 1: Community Before Splitting

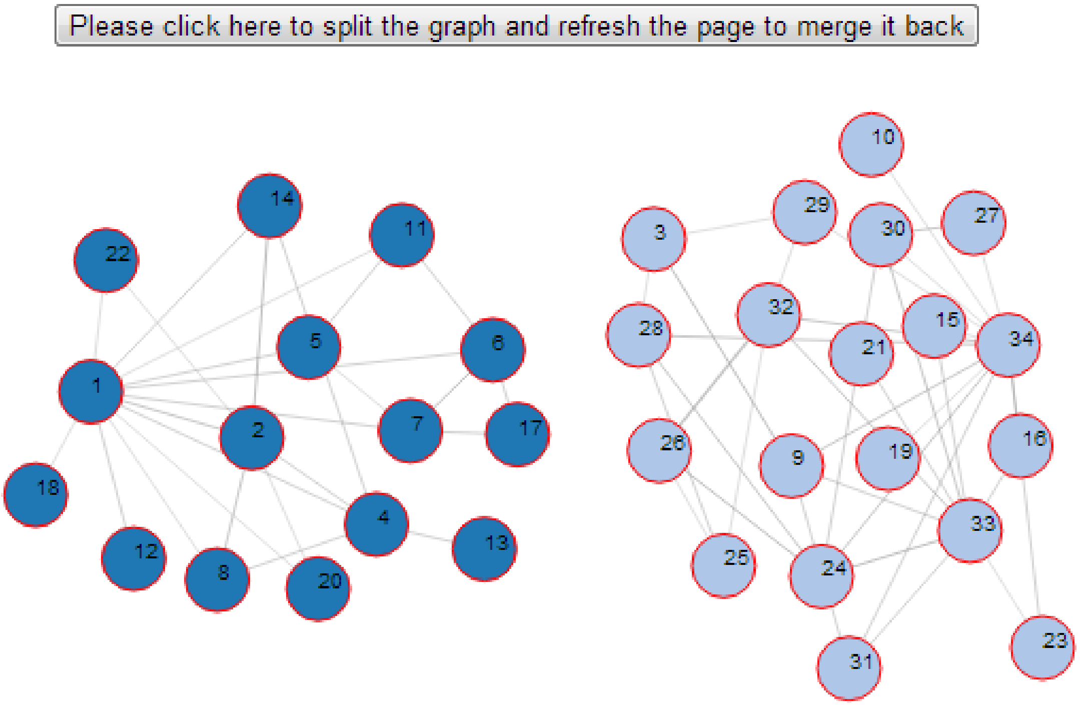
Please click here to split the graph and refresh the page to merge it back
10
14
29
30
27
22
11
3
32
15
28
5
6
21
34
1
2
7
17
26
9
19
16
18
4
33
13
12
8
20
25
24
31
23

Figure 2: Community After Splitting