Mohamed Aturban (maturban)
CS595 — Fall 2013

Assignment 7

November 8, 2013

All files mentioned in this document should be uploaded into the *github* repository.

Problem 1

The same R script, *rcode.r* from assignment 6, has been modified to ,first, read data from `http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat`, and then create two *.json* files (*before.json* and *after.json*) which, respectively, describe the Karate Club graph before and after the split by Girvan-Newman algorithm. The following piece of code is for achieving the first task of reading data– weighted edges from the above link and assigning this graph to a variable named *g*.

```
tmp <- getwd()
url <- "http://vlado.fmf.uni-lj.si/pub/networks/data/
       UciNet/zachary.dat"
dest <- paste(tmp, sep="/", "k.dat")
download.file(url=url, destfile=dest)
l <- readLines(dest)
l <- l[(grep("^DATA", l)+1):length(l)]
l1 <- matrix(scan(textConnection(paste(l[1:34],
      collapse="\n"))), nr=34)
l2 <- matrix(scan(textConnection(paste(l[1:34+34],
      collapse="\n"))), nr=34)
karate_unweighted <- graph.adjacency(l1, weighted=TRUE,
                     mode="undirected")
karate_weighted <- graph.adjacency(l2, weighted=TRUE,
                     mode="undirected")
g <- karate_unweighted
```

While the following lines of code is to produce the two output files (*before.json* and *after.json*). The code is not too long(one page and a half), so it is shown below:

```
# Create Edges vector Before split
original_vec <- numeric()
for (i in 1:ecount(g)) {
       for (j in 1:ecount(karate_weighted)) {
              s1 <- get.edgelist(g)[i,1]
              s2 <- get.edgelist(g)[i,2]
              d1 <- get.edgelist(karate_weighted)[j,1]
              d2 <- get.edgelist(karate_weighted)[j,2]
              if ((s1 == d1) && (s2 == d2))
              {
                     original_vec <- append(original_vec,get.edgelist
                                    (karate_weighted)[j,1] - 1)
                     original_vec <- append(original_vec,get.edgelist
                                    (karate_weighted)[j,2] - 1)
                     original_vec <- append(original_vec,
                                        E(karate_weighted)$weight[j])
                }
        }
}
# Create Edges vector After split
vec <- numeric()
for (i in 1:ecount(g)) {
        for (j in 1:ecount(karate_weighted)) {
                s1 <- get.edgelist(g)[i,1]
                s2 <- get.edgelist(g)[i,2]
                d1 <- get.edgelist(karate_weighted)[j,1]
                d2 <- get.edgelist(karate_weighted)[j,2]
                if ((s1 == d1) && (s2 == d2))
                {
                        vec <- append(vec,get.edgelist
                                        (karate_weighted)[j,1] - 1 )
                        vec <- append(vec,get.edgelist
                                        (karate_weighted)[j,2] - 1 )
                        vec <- append(vec,E(karate_weighted)$weight[j])
                }
        }
}

# Create Data Frames for Edges and nodes before and after split

# Edges before split
o <- matrix(original_vec,ncol=3,byrow = TRUE)
edgeBefore <- as.data.frame(o)
colnames(edgeBefore) <- c("source", "target","value")

# Edges after split
m <- matrix(vec,ncol=3,byrow = TRUE)
edgeAfter <- as.data.frame(m)
colnames(edgeAfter) <- c("source", "target","value")

# Nodes before split
c <- 1
a <- as.numeric( unlist(clusters(karate_unweighted)['membership']))
vecnode <- numeric()
for (i in 1:34) {
        vecnode <- append(vecnode,c)
        vecnode <- append(vecnode,a[i])
        c <- c + 1
}
matnode <- matrix(vecnode,ncol=2,byrow = TRUE)
nodeBefore <- as.data.frame(matnode)
```

```
colnames(nodeBefore) <- c("name","group")

# Nodes after split
c <- 1
a <- as.numeric( unlist(clusters(g)[’membership’]))
vecnode <- numeric()
for (i in 1:34) {
        vecnode <- append(vecnode,c)
        vecnode <- append(vecnode,a[i])
        c <- c + 1
}
matnode <- matrix(vecnode,ncol=2,byrow = TRUE)
nodeAfter <- as.data.frame(matnode)
colnames(nodeAfter) <- c("name","group")


# Create Output Files in JSON format
toJSONarray <- function(dtf){
        clnms <- colnames(dtf)
        name.value <- function(i){
                quote <- ’’;
                if(class(dtf[, i])!=’numeric’ && class(dtf[, i])!=
                ’integer’){
                        quote <- ’’;
                }
                paste(’"’, i, ’" : ’, quote, dtf[,i], quote, sep=’’)
        }
        objs <- apply(sapply(clnms, name.value), 1, function(x){paste(x,
        collapse=’, ’)})
        objs <- paste(’{’, objs, ’}’)
        res <- paste(’[’, paste(objs, collapse=’,\n’), ’]’)
        return(res)
}

# JSON file before split
outputstr = paste(’{\n"nodes":\n’, toJSONarray(dataFramenodebefore))
outputstr = paste(outputstr, ’,\n"links":\n’)
outputstr = paste(outputstr, toJSONarray(dataFrame1))
outputstr = paste(outputstr, "\n}\n")
fileConn<-file("before.json")
writeLines(outputstr, fileConn)
close(fileConn)
# JSON file after split
outputstr = paste(’{\n"nodes":\n’, toJSONarray(dataFramenode))
outputstr = paste(outputstr, ’,\n"links":\n’)
outputstr = paste(outputstr, toJSONarray(dataFrame2))
outputstr = paste(outputstr, "\n}\n")
fileConn<-file("after.json")
writeLines(outputstr, fileConn)
close(fileConn)
```

After, successfully, getting the two files (*before.json* and *after.json*) , it is time now to use *d3 js* to simulate the two graphs before and after the split. The following steps, briefly, describe how the code, in *index.html*, works:

- Read the graph vertices and edges from *before.json*
- Draw its nodes, edges, and labels.
- When clicking, by mouse, on any of the nodes (here, the click should not be on the node label), the current graph will be removed from the screen. Instead, vertices and edges from *after.json* will be drawn.
- If you click , again, on any node within the two graphs, the same scenario will be repeated from step 1.

Output samples:

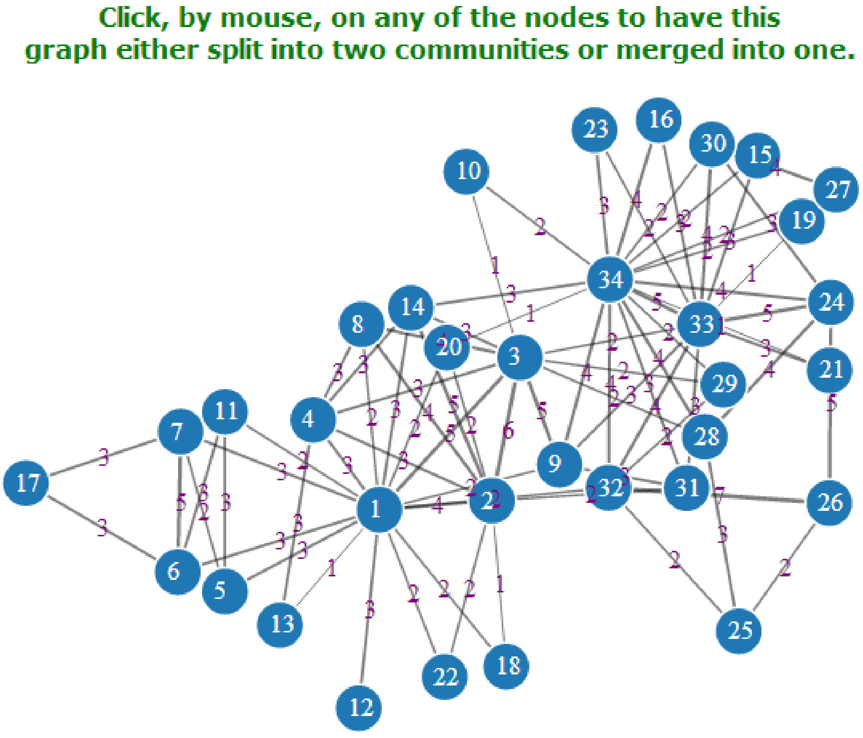
1. The code can be tested online at:

   `http://www.cs.odu.edu/~maturban/indexmoh.html`
2. Figure 1. shows the web-page when first loaded. It contains a green sentence which guides the user of how the graph(s) can be split/merged.
3. Now, if clicking on any node, not on the node label, the graph will be split into two clusters as you can see in figure 2.
4. By clicking again on any node within the two graphs, they will be merged into one graph as shown in figure 1.

index.html is uploaded into the course repository. Also, I would like to mention that I uploaded the file *reference.txt* which contains all resources I have used for this assignment. As you suggested, starting from next assignment (no. 8), I will try to use BibTex for references (I promise).

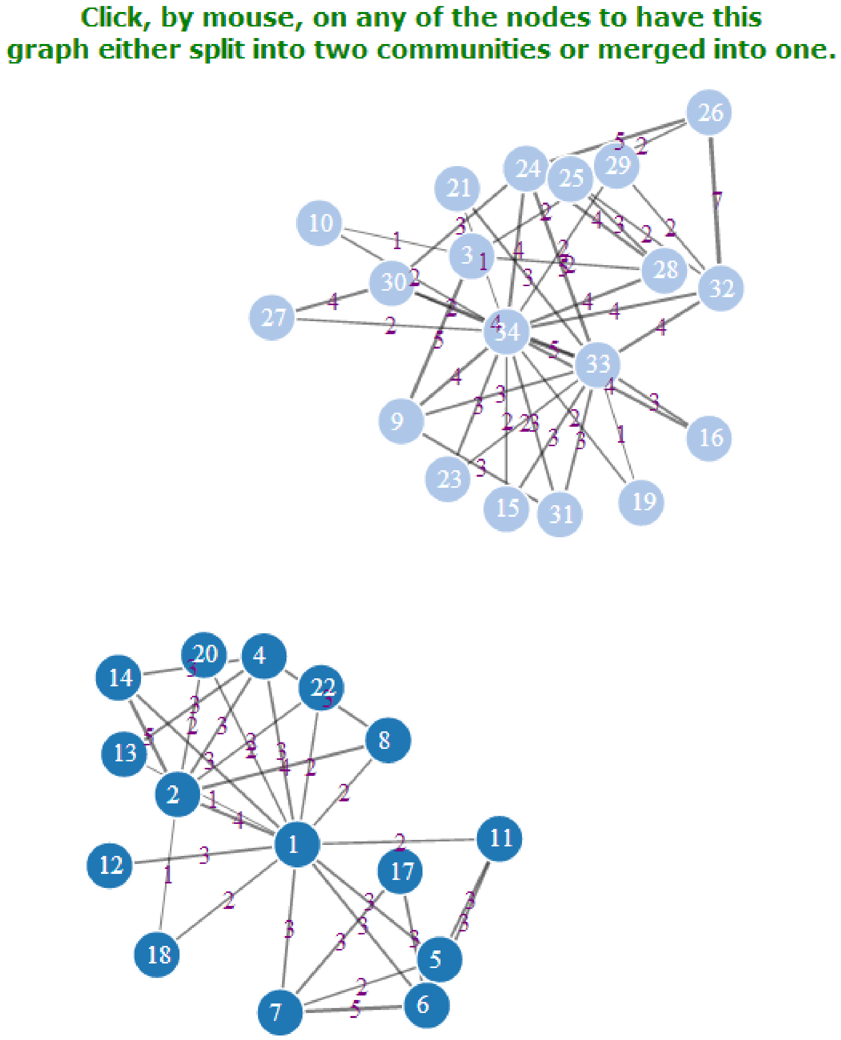
Problem 2

I was interested to see my followers graph on Twitter using D3, but it is matter of time.



Click, by mouse, on any of the nodes to have this graph either split into two communities or merged into one.

Figure 1: Karate Club graph before split



Click, by mouse, on any of the nodes to have this graph either split into two communities or merged into one.

Figure 2: Karate Club graph after split