MMD MOOC - Quiz 1

Sergey Legotsky

Monday, October 06, 2014

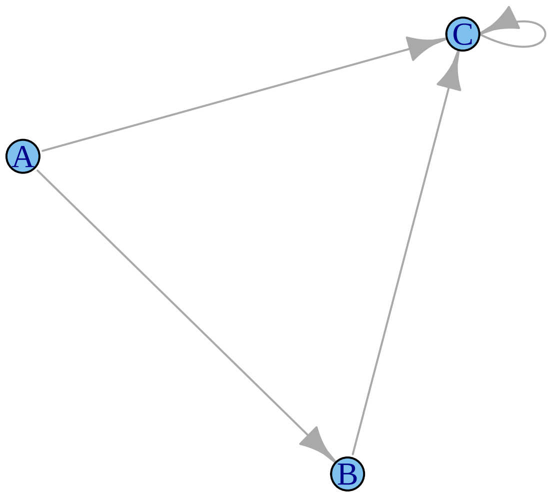
General options. We used `igraph` library to plot and compute some parameter of graph(s).

```
library(igraph)
```

========== **Task 1** ==========

Consider three Web pages with the following links:

```
a <- graph.formula(A--+B--+C, A--+C, C--+C, simplify=F)
plot(a)
```



Suppose we compute PageRank with a beta of 0.7, and we introduce the additional constraint that the sum of the PageRanks of the three pages must be 3, to handle the problem that otherwise any multiple of a solution will also be a solution. Compute the PageRanks a, b, and c of the three pages A, B, and C, respectively. Then, identify from the list below, the true statement.

- **b + c = 2.7**
- **a + b = 0.55**
- **b + c = 2.5**
- **a + c = 1.985**

Solution 1

```r
#Set beta value:
beta <- 0.7

# Create M matrix:
M <- cbind(c(0, 0.5, 0.5), c(0, 0, 1), c(0, 0, 1))
M
```

```
##      [,1] [,2] [,3]
## [1,]  0.0    0    0
## [2,]  0.5    0    0
## [3,]  0.5    1    1
```

```r
# Create matrix for Random Teleports:
RT <- matrix(rep(1/3, 9), nrow = 3, ncol = 3)
RT
```

```
##        [,1]   [,2]   [,3]
## [1,] 0.3333 0.3333 0.3333
## [2,] 0.3333 0.3333 0.3333
## [3,] 0.3333 0.3333 0.3333
```

```r
# Create A matrix--graph column-stochastic matrix which takes into account Random Teleports
A <- beta*M + (1 - beta)*RT

#By definition, PageRank is the main eigen vector of matrix A that corresponds to eigen value 1
pageRank <- eigen(A)$vectors[,1]

# Normalize PageRank in such way that the sum of entries equals 3:
pageRank <- pageRank*3/sum(pageRank)
pageRank
```

```
## [1] 0.300+0i 0.405+0i 2.295+0i
```

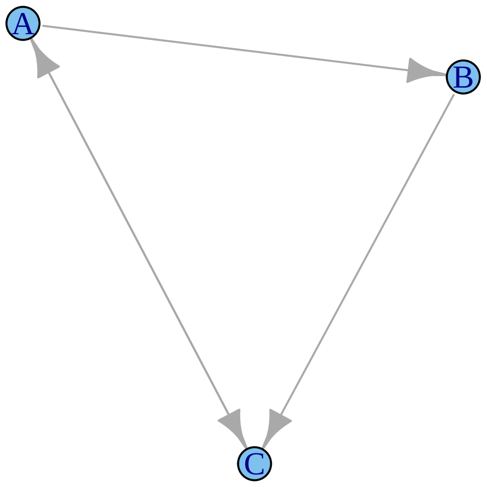
Now checking the initial equations in this task we see that $\mathbf{b + c = 2.7}$ is TRUE.

```r
# clean the workspace
rm(list = ls())
```

========== **Task 2.** ==========

Consider three Web pages with the following links:

```r
a <- graph.formula(A--+B--+C---+A, A--+C, simplify=F)
plot(a)
```



Suppose we compute PageRank with beta = 0.85. Write the equations for the PageRanks a, b, and c of the three pages A, B, and C, respectively. Then, identify in the list below, one of the equations:

- **c = .9b + .475a**
- **85a = c + .15b**
- **c = b + .575a**
- **.95c = .9b + .475a**

Solution 2

```
#Set beta value:
beta <- 0.85

# Create M matrix:
M <- cbind(c(0, 0.5, 0.5), c(0, 0, 1), c(1, 0, 0))
colnames(M) <- c("A", "B", "C")
rownames(M) <- c("A", "B", "C")
M
```

```
##     A B C
## A 0.0 0 1
## B 0.5 0 0
## C 0.5 1 0
```

```r
# Create matrix for Random Teleports:
RT <- matrix(rep(1/3, 9), nrow = 3, ncol = 3)
colnames(RT) <- c("A", "B", "C")
rownames(RT) <- c("A", "B", "C")
RT
```

```
##        A      B      C
## A 0.3333 0.3333 0.3333
## B 0.3333 0.3333 0.3333
## C 0.3333 0.3333 0.3333
```

```r
# Create A matrix--graph column-stochastic matrix which takes into account Random Teleports
A <- beta*M + (1 - beta)*RT
A
```

```
##       A    B    C
## A 0.050 0.05 0.90
## B 0.475 0.05 0.05
## C 0.475 0.90 0.05
```

This graph matrix corresponding to the following equations:

- **a = .05a + .05b + .9c**
- **b = .475a + .05b + .05c**
- **c = .475a + .9b + .05c**

Which in turn is equivalent to:

- **.95a = .05b + .9c**
- **.95b = .475a + .05c**
- **.95c = .475a + .9b**

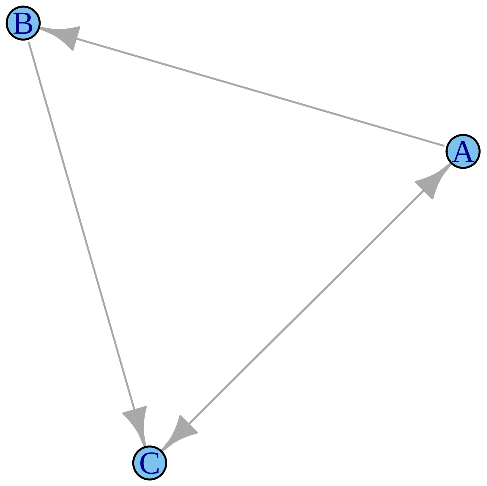
Last equation is the correct answer

```r
#cleaning workspace
rm(list=ls())
```

========== Task 3 ==========

Consider three Web pages with the following links:

```r
graph.a <- graph.formula(A--+B--+C---+A, A--+C, simplify=F)
plot(graph.a)
```



Assuming no "taxation", compute the PageRanks a, b, and c of the three pages A, B, and C, using iteration, starting with the "0th" iteration where all three pages have rank a = b = c = 1. Compute as far as the 5th iteration, and also determine what the PageRanks are in the limit. Then, identify the true statement from the list below.

- **In the limit, c = 6/5**
- **After iteration 4, b = 11/16**
- **After iteration 4, c = 11/8**
- **After iteration 4, b = 3/5**

Solutioin 3

```
# Create M matrix:
M <- t(as.matrix(get.stochastic(graph.a)))
M
```

```
##     A B C
## A 0.0 0 1
## B 0.5 0 0
## C 0.5 1 0
```

```
# set iteration counter
t <- 1

# set initial values for PageRank score:
```

```r
a <- 1; b <- 1; c <- 1

# compute final Page Ranks and normalize it to 1:
final.Page.Rank <- eigen(M)$vectors[,1]
final.Page.Rank <- final.Page.Rank/sum(final.Page.Rank)
final.Page.Rank
```

```
## [1] 0.4+0i 0.2+0i 0.4+0i
```

```r
# `for` loop to compute PageRank vector r at each power iteration algorythm:
## set PageRank vector r; create list for storage value of r vector at each iteration:

r <- as.numeric(c(a, b, c))
rank.list <- list("0" = r)


for (i in 1:10){
  # calculate new r value
  r <- M%*%r
  # put new r value to the rank.list
  rank.list[[i+1]] <- r
  # name the element of list as iteration number
  names(rank.list)[i+1] <- as.character(t)
  # count iteration
  t <- t+1
}

str(rank.list)
```

```
## List of 11
##  $ 0 : num [1:3] 1 1 1
##  $ 1 : num [1:3, 1] 1 0.5 1.5
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 2 : num [1:3, 1] 1.5 0.5 1
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 3 : num [1:3, 1] 1 0.75 1.25
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 4 : num [1:3, 1] 1.25 0.5 1.25
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 5 : num [1:3, 1] 1.25 0.625 1.125
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 6 : num [1:3, 1] 1.125 0.625 1.25
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 7 : num [1:3, 1] 1.25 0.562 1.188
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 8 : num [1:3, 1] 1.188 0.625 1.188
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 9 : num [1:3, 1] 1.188 0.594 1.219
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
##  $ 10: num [1:3, 1] 1.219 0.594 1.188
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "A" "B" "C"
##   .. ..$ : NULL
```

We see that after iteration **4** we have following PageRanks:

a = c = 5/4; b = 0.5

That is why by elimination we can state that correct answer is 1st:

at final, c = 6/5 and it is very close to the truth by the way :)

```
#cleaning workspace
rm(list=ls())
```

========== Task 4 ==========

Suppose our input data to a map-reduce operation consists of integer values (the keys are not important). The map function takes an integer i and produces the list of pairs (p,i) such that p is a prime divisor of i. For example, map(12) = [(2,12), (3,12)]. The reduce function is addition. That is, reduce(p, [i1, i2, ...,ik]) is (p,i1+i2+...+ik).

Compute the output, if the input is the set of integers 15, 21, 24, 30, 49. Then, identify, in the list below, one of the pairs in the output.

Solution 4

```
library(surveillance) # required to find prime divisors (factors of integer)
```

```
## Warning: package 'surveillance' was built under R version 3.1.1
## Warning: package 'sp' was built under R version 3.1.1
## Warning: package 'polyCub' was built under R version 3.1.1
```

```
# initial vector of integers:
values <- as.numeric(c(15, 21, 24, 30, 49))

#for each value calculate prime factors
results <- data.frame(divisor = NA, value = NA)


for (i in seq_along(values)){
  divisors <- unique(primeFactors(values[i]))

  #put each divisor together with its value to results dataframe
  for (j in 1:length(divisors)){
    results <- rbind(results, c(divisors[j], values[i]))
  }
}

# remove 1st NA row and ordering by divisor
results <- results[-1, ]
results <- results[order(results$divisor),]

#
unique.divisors <- unique(results$divisor)
final.df <- data.frame(divisor = numeric(0), values.sum = numeric(0))

for (k in seq_along(unique.divisors)){
  sub.results <- results[results$divisor == unique.divisors[k],]
  values.sum <- sum(sub.results[,2])
  final.df <- rbind(final.df, c(unique.divisors[k], values.sum))
}

final.df
```

```
##   X2 X54
## 1  2  54
## 2  3  90
## 3  5  45
## 4  7  70
```

```
rm(list=ls())
```

Now we see that only **(7, 70)** pair of values matches the option in question.

Nice done! Final grade is 4/4!!!