Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

LAB- 4

1. Read the given adjacency matrix into R (adjacency.csv)

```
adj = read.csv(file = 'Adjacency.csv',header= TRUE, sep= ',',as.is= TRUE)

m = as.matrix(adj)

View(m)
```

rstudio

	X	X2000	X2001	X2002	X2003	X2004	X2005	X2006	X2007	X2008	X2009	X2010
1	2000	0	1	0	1	0	1	0	1	0	0	0
2	2001	1	1	1	0	1	0	1	1	0	1	1
3	2002	1	1	0	0	1	0	0	0	0	0	0
4	2003	1	1	1	0	0	1	0	1	0	1	0
5	2004	1	0	1	0	0	1	0	1	0	1	1
6	2005	0	0	1	1	1	0	1	0	0	0	1
7	2006	0	0	1	0	1	0	0	0	1	0	1
8	2007	0	0	1	1	1	0	1	1	0	0	1
9	2008	1	0	0	0	1	0	1	0	0	1	0
10	2009	1	1	1	0	1	1	0	1	1	1	0
11	2010	0	1	1	1	0	1	1	1	1	1	1

2. Read the given edge matrix into R(edges.csv)

```
edge= read.csv(file = 'Edges.csv', header = TRUE, sep= ',')

a = as.matrix(edge)

View(a)
```

m × Lab5.R* × a ×

Filter

	V1	V2
1	2000	2001
2	2001	2002
3	2002	2004
4	2003	2004
5	2004	2003
6	2004	2002
7	2006	2008
8	2006	2007
9	2008	2010
10	2008	2000
11	2010	2000

Showing 1 to 11 of 11 entries

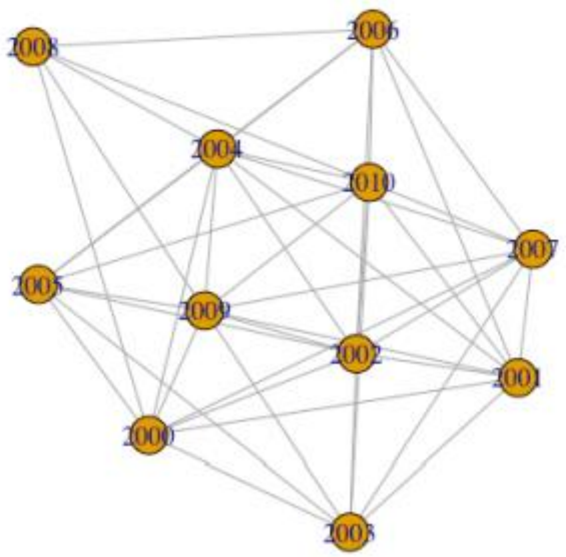
Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

3. **Create and plot the graph from the adjacency matrix and edge matrix (customize the vertex color edge size, vertex frame and label)**

A. Using adjacency matrix

n = graph.adjacency(n, mode = "undirected", weighted = TRUE, diag = FALSE)

plot.igraph(n, layout = layout.fruchterman.reingold, vertex.label = V(n)$name, edge.size = 1)



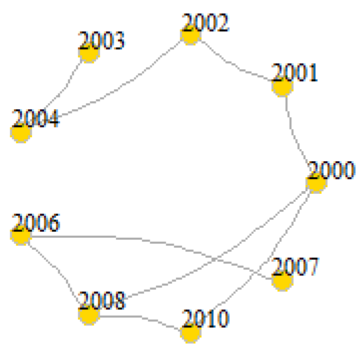
B. Using edge list

gg = graph.data.frame(edge, directed = FALSE)

plot(gg,layout= layout.circle, edge.arrow.size=.5, vertex.color="gold", vertex.size=15,

vertex.frame.color="gray", vertex.label.color="black",

vertex.label.cex=0.8, vertex.label.dist=2, edge.curved=0.2)



Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

4. **Display the edges & vertices**

V(gg)

E(gg)

```
> V(gg)
+ 9/9 vertices, named, from 1d8e4d8:
[1] 2000 2001 2002 2003 2004 2006 2008 2010 2007
> E(gg)
+ 11/11 edges from 1d8e4d8 (vertex names):
 [1] 2000--2001 2001--2002 2002--2004 2003--2004 2003--2004
 [6] 2002--2004 2006--2008 2006--2007 2008--2010 2000--2008
[11] 2000--2010
> 
```

5. **Display the network as matrix**

```
> get.adjacency(gg)
9 x 9 sparse Matrix of class "dgCMatrix"
     2000 2001 2002 2003 2004 2006 2008 2010 2007
2000    .    1    .    .    .    .    1    1    .
2001    1    .    1    .    .    .    .    .    .
2002    .    1    .    .    2    .    .    .    .
2003    .    .    .    .    2    .    .    .    .
2004    .    .    2    2    .    .    .    .    .
2006    .    .    .    .    .    .    1    .    1
2008    1    .    .    .    .    1    .    1    .
2010    1    .    .    .    .    .    1    .    .
2007    .    .    .    .    .    1    .    .    .
> 
```

6. **Display the names of vertices**

```
> V(gg)$name
[1] "2000" "2001" "2002" "2003" "2004" "2006" "2008" "2010"
[9] "2007"
> for (i in 1:vcount(gg)) {
+   if(as.numeric(V(gg)[i]$name) %% 4 == 0) {
+     V(gg)[i]$name = 'Leap'
+   } else {
+     V(gg)[i]$name = 'Non-leap'
+   }
+ }
> V(gg)$name
[1] "Leap"     "Non-leap" "Non-leap" "Non-leap" "Leap"
[6] "Non-leap" "Leap"     "Non-leap" "Non-leap"
> 
```

Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

7. **Find the count of vertices and edges of the created graph**

ecount(gg)

vcount(gg)

```
> ecount(gg)
[1] 11
> vcount(gg)
[1] 9
>
```

8. **Display the adjacency vertices of each vertex(individual) in the created graph**

V(gg)

for(i in 1:vcount(gg)) {

print(neighbors(gg, V(gg)[i]))

}

```
> for(i in 1:vcount(gg)) {
+   print(neighbors(gg, V(gg)[i]))
+ }
+ 3/9 vertices, named, from 1d8e4d8:
[1] Non-leap Leap     Non-leap
+ 2/9 vertices, named, from 1d8e4d8:
[1] Leap     Non-leap
+ 3/9 vertices, named, from 1d8e4d8:
[1] Non-leap Leap     Leap
+ 2/9 vertices, named, from 1d8e4d8:
[1] Leap Leap
+ 4/9 vertices, named, from 1d8e4d8:
[1] Non-leap Non-leap Non-leap Non-leap
+ 2/9 vertices, named, from 1d8e4d8:
[1] Leap     Non-leap
+ 3/9 vertices, named, from 1d8e4d8:
[1] Leap     Non-leap Non-leap
+ 2/9 vertices, named, from 1d8e4d8:
[1] Leap Leap
+ 1/9 vertex, named, from 1d8e4d8:
[1] Non-leap
>
```

9. **Find the min and max degree of the created graph**

degree(gg, V(gg))

max(degree(gg, V(gg)))

min(degree(gg, V(gg)))

```
> degree(gg, V(gg))
    Leap Non-leap Non-leap Non-leap     Leap Non-leap     Leap 
       3        2        3        2        4        2        3 
Non-leap Non-leap 
       2        1 
> max(degree(gg, V(gg)))
[1] 4
> min(degree(gg, V(gg)))
[1] 1
>
```

Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

10. Create & set vertex attribute property named profit and values("+", "-", "+", "-", "+", "-", "+", "-", "+")

```
for(i in 1:vcount(gg)) {
 if(i %% 2 == 0) {
   V(gg)[i]$profit = '-'
 } else {
   V(gg)[i]$profit = '+'
 }
}
V(gg)$profit
```

```
> for(i in 1:vcount(gg)) {
+   if(i %% 2 == 0) {
+     V(gg)[i]$profit = '-'
+   } else {
+     V(gg)[i]$profit = '+'
+   }
+ }
> V(gg)$profit
[1] "+" "-" "+" "-" "+" "-" "+" "-" "+"
>
```

11. Create & set vertex attribute property named type and values(either leap or nonleap year)

for (i in 1:vcount(g)) { if(as.numeric(V(g)[i]$name) %% 4 == 0) { V(g)[i]$type = 'Leap' } else { V(g)[i]$type = 'Non-leap' } } V(g)$type

```
> for (i in 1:vcount(g)) {
+   if(as.numeric(V(g)[i]$name) %% 4 == 0) {
+     V(g)[i]$type = 'Leap'
+   } else {
+     V(g)[i]$type = 'Non-leap'
+   }
+ }
> V(g)$type
[1] "Leap"     "Non-leap" "Non-leap" "Non-leap" "Leap"     "Non-leap" "Leap"     "Non-leap" "Non-leap"
>
```

Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

12. Create & set edge attribute named weight and values (if edge exits in between leap year vertices then 5 else 1)

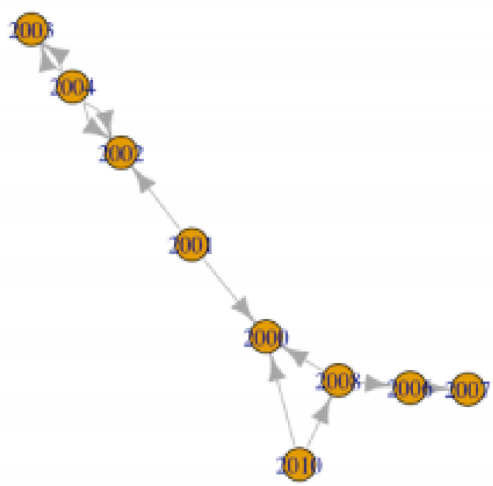
```
l = get.edgelist(gg)

for(i in 1:ecount(gg)) {

  if(as.numeric(l[i]) %% 4 == 0 && as.numeric(l[i + ecount(gg)]) %% 4 == 0) {

    E(gg)[i]$weight = 5

  } else {

    E(gg)[i]$weight = 1

  }

}

get.edgelist(gg)

E(gg)$weight
```

```
> l = get.edgelist(g)
> for(i in 1:ecount(g)) {
+   if(as.numeric(l[i]) %% 4 == 0 && as.numeric(l[i + ecount(g)]) %% 4 == 0) {
+     E(g)[i]$weight = 5
+   } else {
+     E(g)[i]$weight = 1
+   }
+ }
> get.edgelist(g)
      [,1]   [,2]
 [1,] "2000" "2001"
 [2,] "2001" "2002"
 [3,] "2002" "2004"
 [4,] "2003" "2004"
 [5,] "2003" "2004"
 [6,] "2002" "2004"
 [7,] "2006" "2008"
 [8,] "2006" "2007"
 [9,] "2008" "2010"
[10,] "2000" "2008"
[11,] "2000" "2010"
> E(g)$weight
 [1] 1 1 1 1 1 1 1 1 1 5 1
>
```

Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

13. **Convert the created un-directed graph into directed graph based on the following rule.**
    a. **edge directed towards high value vertex**
    b. **if any one of the vertex is leap year then put the reverse edge with same weight.**

```
c1 = c()

c2 = c()

for(i in 1:ecount(gg)) {

 if(as.numeric(l[i]) %% 4 == 0 || as.numeric(l[i + ecount(gg)]) %% 4 == 0) {

   c1 = c(c1, as.numeric(l[i + ecount(gg)]))

   c2 = c(c2, as.numeric(l[i]))

 } else {

   if(as.numeric(l[i]) > as.numeric(l[i + ecount(gg)])){

     c1 = c(c1, as.numeric(l[i + ecount(gg)]))

     c2 = c(c2, as.numeric(l[i]))

   } else {

     c2 = c(c2, as.numeric(l[i + ecount(gg)]))

     c1 = c(c1, as.numeric(l[i]))

   }

 }

}

data = cbind(c1, c2)

p = graph.data.frame(data,
directed = TRUE)

plot(p)
```



Name: Samriddhi Verma
Reg. No.: 16BCE1375
Slot: L49+L50
Prof. Tulasi Prasad Sariki

14. Display the adjacency matrix of the resultant directed graph.

get.adjacency(p)

```
> get.adjacency(p)
9 x 9 sparse Matrix of class "dgCMatrix"
     2001 2004 2008 2006 2010 2000 2002 2003 2007
2001    .    .    .    .    .    1    1    .    .
2004    .    .    .    .    .    .    2    2    .
2008    .    .    .    1    .    1    .    .    .
2006    .    .    .    .    .    .    .    .    1
2010    .    .    1    .    .    1    .    .    .
2000    .    .    .    .    .    .    .    .    .
2002    .    .    .    .    .    .    .    .    .
2003    .    .    .    .    .    .    .    .    .
2007    .    .    .    .    .    .    .    .    .
> 
```

15. Display the in-degree and out-degree of each vertex of resultant directed graph

degree(p, mode = "out")

degree(p, mode = 'in')

```
> degree(p, mode = "out")
2001 2004 2008 2006 2010 2000 2002 2003 2007 
   2    4    2    1    2    0    0    0    0 
> degree(p, mode = 'in')
2001 2004 2008 2006 2010 2000 2002 2003 2007 
   0    0    1    1    0    3    3    2    1 
> 
```