Faculty Development Programme on

Network Science: Foundation Of Social Network Analysis

Topological Properties of Networks

Hands-on Session (Day 2)

Swagata Duari
Kirti Jain

Create a graph from an edge list as matrix

```
> el <- matrix ( c ( "V1" , "V2",
                     "V1" , "V3",
                     "V1" , "V4",
                     "V1" , "V5",
                     "V2" , "V5",
                     "V3" , "V4",
                     "V4" , "V5",
                     "V4" , "V7",
                     "V5" , "V8",
                     "V6" , "V2",
                     "V7" , "V8" ) , nc = 2, byrow = TRUE )
> el
```

Data(vector)

#cols

Fill matrix by rows

OUTPUT :

```
      [,1] [,2]
 [1,] "V1" "V2"
 [2,] "V1" "V3"
 [3,] "V1" "V4"
 [4,] "V1" "V5"
 [5,] "V2" "V5"
 [6,] "V3" "V4"
 [7,] "V4" "V5"
 [8,] "V4" "V7"
 [9,] "V5" "V8"
[10,] "V6" "V2"
[11,] "V7" "V8"
```

Edge list or 2-column Matrix

Undirected graph

```
> G_el <- Graph_from_edgelist ( el, directed = FALSE )
```

OUTPUT :

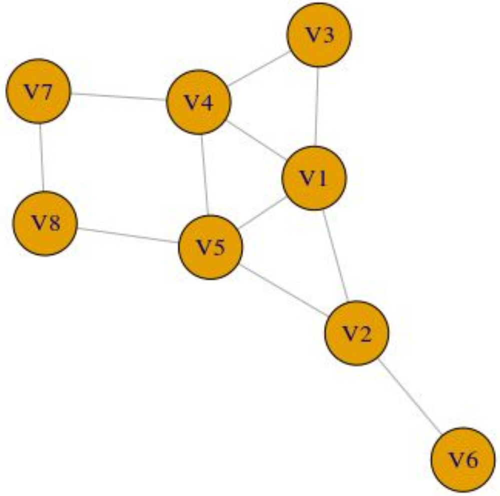
```
IGRAPH  UN-- 8 11 --
+ attr: name (v/c)
+ edges (vertex names):
 [1] V1--V2 V1--V3 V1--V4 V1--V5 V2--V5 V3--V4 V4--V5 V4--V7 V5--V8 V2--V6 V7--V8
```

Igraph graph, Undirected Named graph, 8=#vertices, 11=#edges

Attribute: named vertex/character

```
> plot ( G_el )
```

OUTPUT :



PART 1: Local Properties

Network Descriptive : Degree

```
> deg <- degree ( network )
> deg
```

OUTPUT :

V1	V2	V3	V4	V5	V6	V7	V8
4	3	2	4	4	1	2	2

```
> sort ( deg, decreasing = TRUE)
```

OUTPUT :

V1	V4	V5	V2	V3	V7	V8	V6
4	4	4	3	2	2	2	1

```
> max ( deg)
```

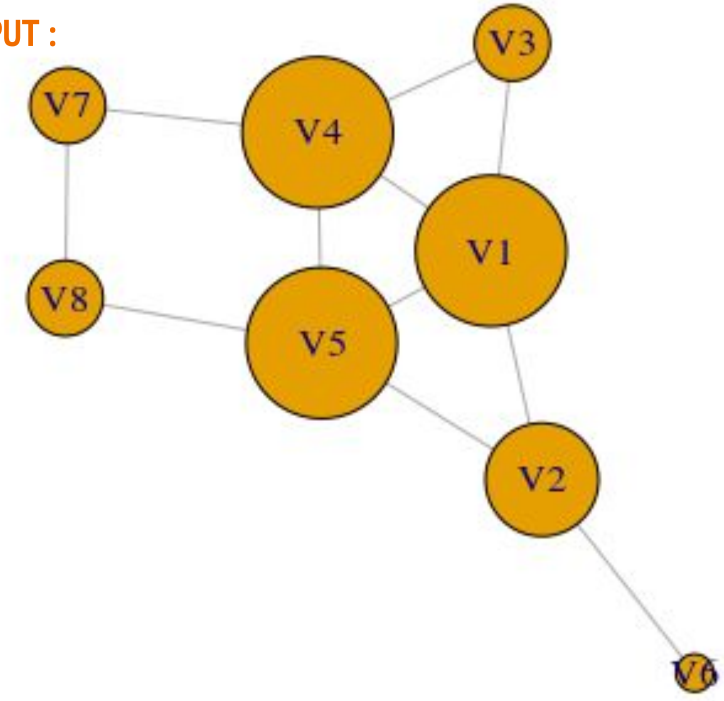
OUTPUT : 4

```
> min ( deg)
```

OUTPUT : 1

```
> plot ( network , vertex.size=deg*12)
# plot graph with node size according to degree of nodes.
```

OUTPUT :



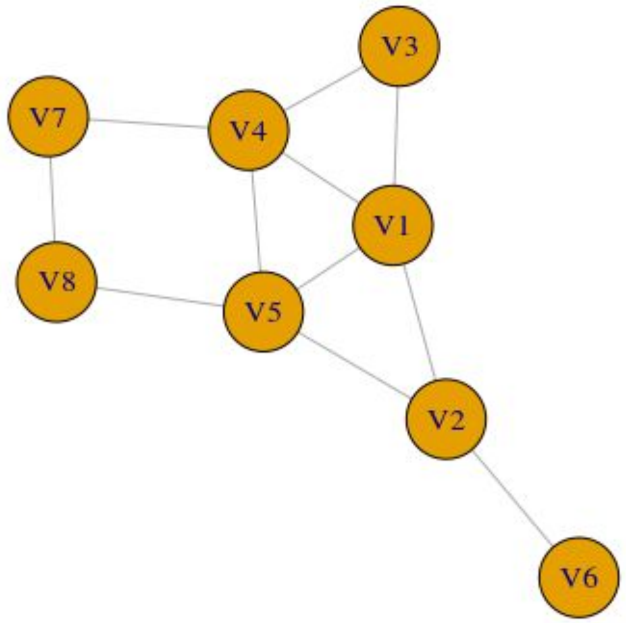
Computing degree from Adjacency Matrix

```
> am <- get.adjacency ( network ,sparse=FALSE)
> am
```

OUTPUT :

	V1	V2	V3	V4	V5	V6	V7	V8
V1	0	1	1	1	1	0	0	0
V2	1	0	0	0	1	1	0	0
V3	1	0	0	1	0	0	0	0
V4	1	0	1	0	1	0	1	0
V5	1	1	0	1	0	0	0	1
V6	0	1	0	0	0	0	0	0
V7	0	0	0	1	0	0	0	1
V8	0	0	0	0	1	0	1	0

Adjacency Matrix



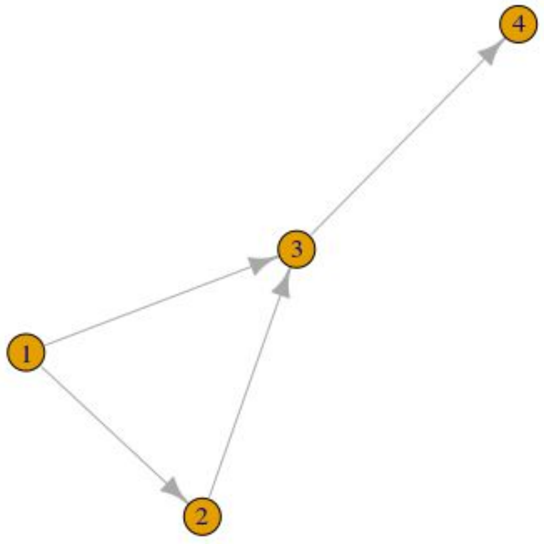

```
> rowSums ( am )
# sum of each row gives the degree of that node.
```

OUTPUT :

V1	V2	V3	V4	V5	V6	V7	V8
4	3	2	4	4	1	2	2

Degree

Indegree and Outdegree of Directed graph




```
> degree ( graph, mode="in")     #indegree
```

OUTPUT :

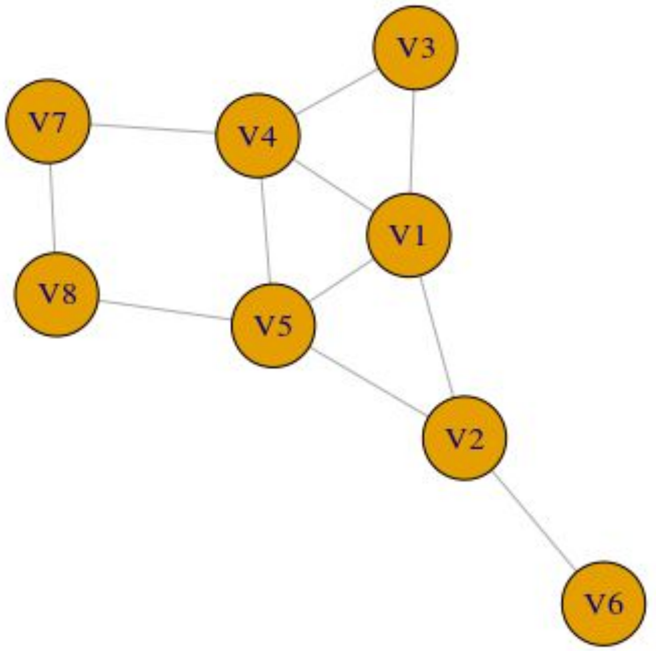
Vertex: 1 2 3 4
Indegree: 0 1 2 1

```
> degree ( graph, mode="out")    #outdegree
```

OUTPUT :

Vertex: 1 2 3 4
Outdegree: 2 1 1 0

Clustering Coefficient of a node




```
> transitivity ( network, type="local" )  #clustering coeff of each node
```

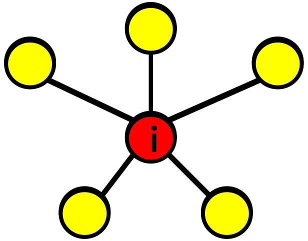
OUTPUT :

	V1	V2	V3	V4	V5	V6	V7	V8
CC:	0.5	0.3	1	0.3	0.3	NaN	0	0

Since V6 has only one neighbor.

Clustering Coefficient of a node

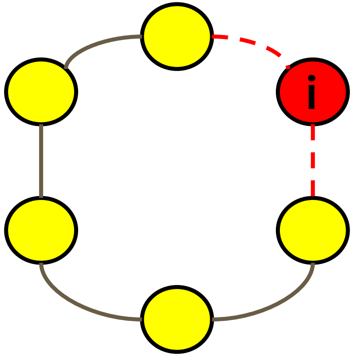
Star:



$cc_i = 0$

No connection between neighboring nodes.

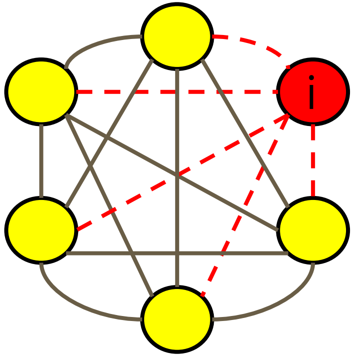
Ring:



$cc_i = 0$

No connection between neighboring nodes.

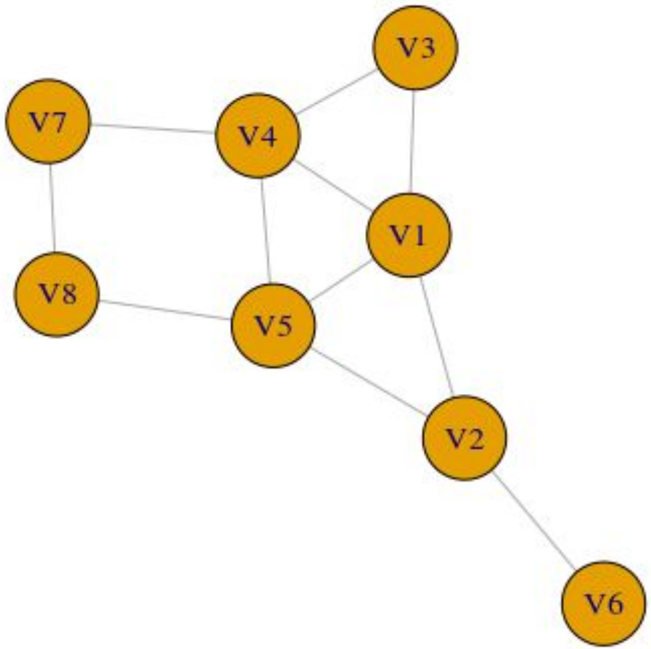
Complete:



$cc_i = 1$

Neighboring nodes are fully connected to each other.

Efficiency (Smaller the distance between the nodes, more efficient is the communication)




```
> efficiency ( network, , type = c("local"))   #Efficiency of nodes
```

OUTPUT :

	V1	V2	V3	V4	V5	V6	V7	V8
Efficiency :	0.7	0.3	1	0.4	0.4	0	0	0

PART 2: Global Properties

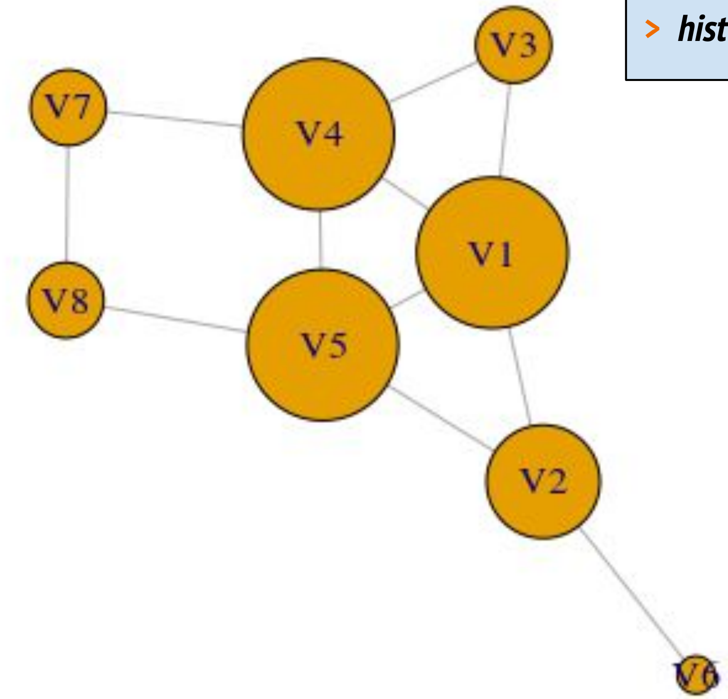
Histogram

Degree Sequence

X-axis = 1: max degree

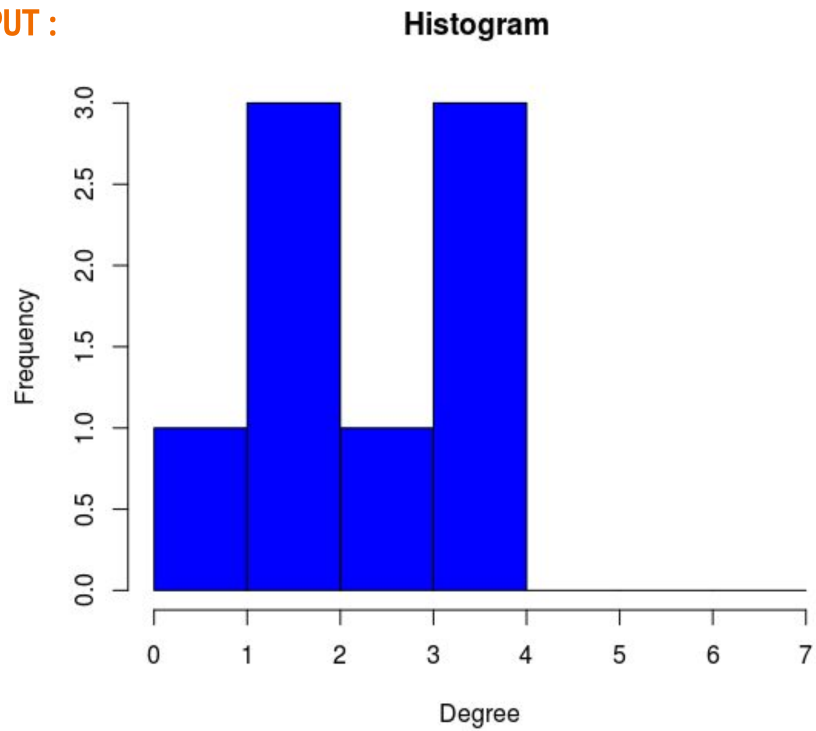
Title

```
> hist (degree, breaks=1: vcount (g)-1, main="Histogram", xlab ="Degree", col="blue")
```



	V1	V4	V5	V2	V3	V7	V8	V6
Degree:	4	4	4	3	2	2	2	1

OUTPUT :



Degree Distribution

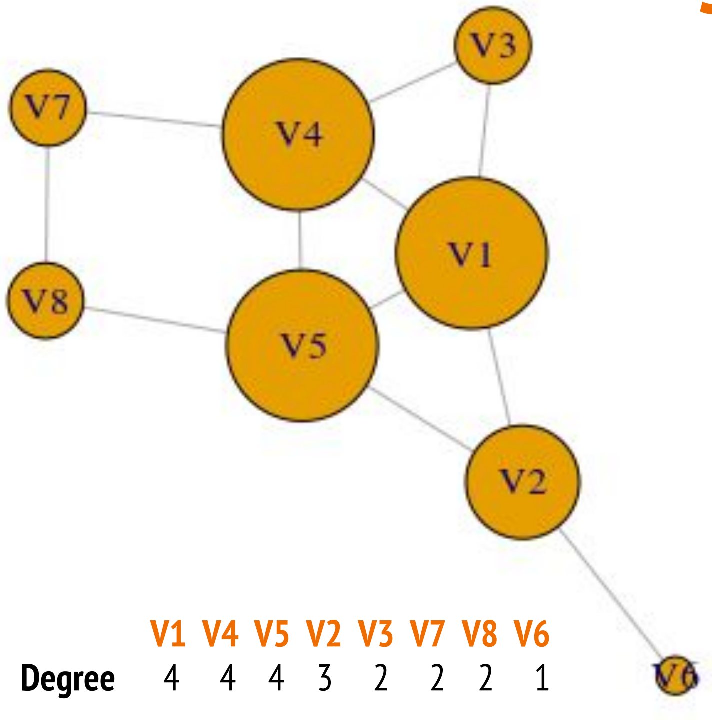
cumulative degree distribution

```
> deg.dist <- degree_distribution(network, cumulative=FALSE, mode="all")
```

OUTPUT : 0 0.125 0.375 0.125 0.375

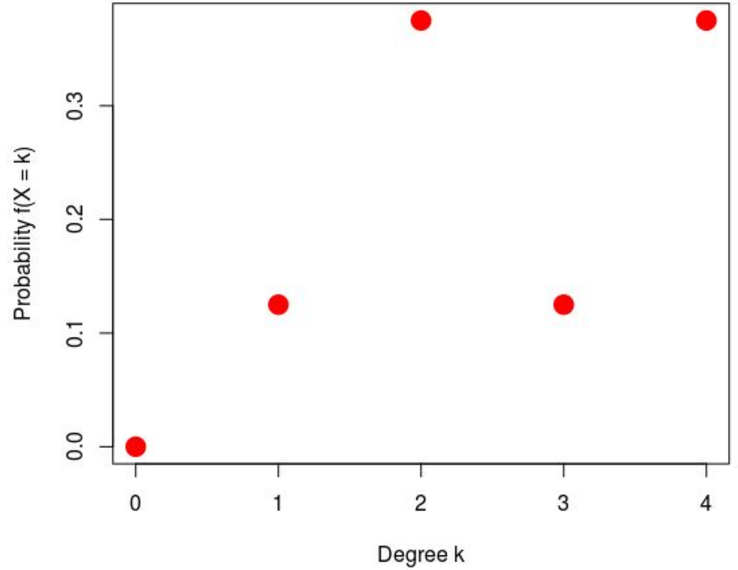
```
> plot ( x=0 : max(degree), y= deg.dist, col="red",  xlab = "Degree k",
        ylab = "Probability f(X = k)")
```

OUTPUT :



	V1	V4	V5	V2	V3	V7	V8	V6
Degree	4	4	4	3	2	2	2	1

Degree k	0	1	2	3	4	
N_k	0	1	3	1	4	#nodes of degree k
$f(X=k)=N_k /N$	0	0.125	0.375	0.125	0.375	degree distribution

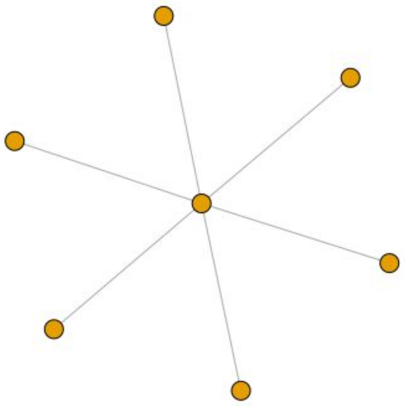


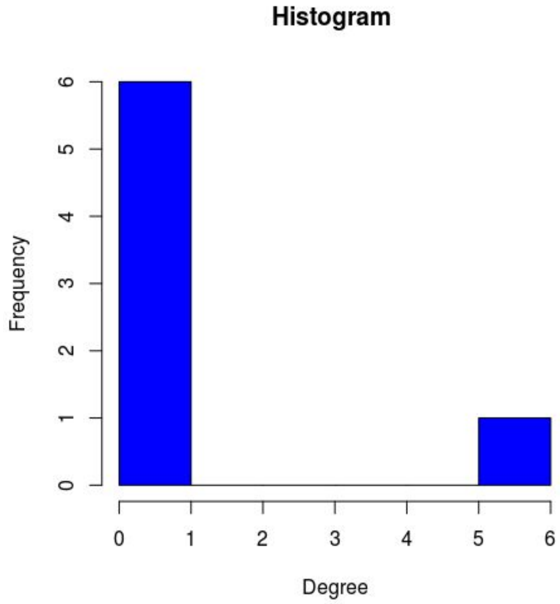
Degree Distribution of Star Graph

Degree Sequence

X-axis = 1: max degree

```
> hist (degree, breaks=1: vcount (network)-1, main="Histogram", xlab ="Degree", col="blue")
```

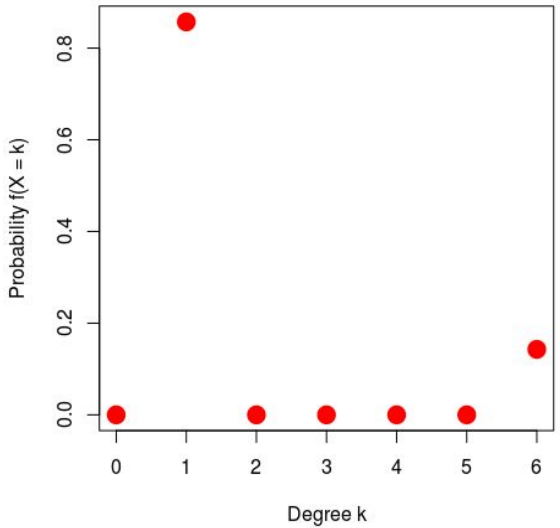




k	0	1	2	3	4	5	6
N_k	0	6	0	0	0	0	1
$F_k = N_k / N$	0	0.85	0	0	0	0	0.14

```
> deg.dist <- degree_distribution (g, cumulative=FALSE, mode="all")
```

```
> plot ( x=0 : max(degree), y=deg.dist, col="red", xlab="Degree k", ylab="Probability f(X = k)")
```

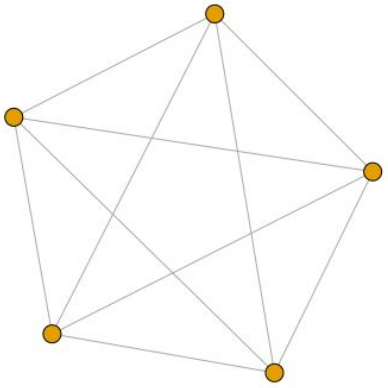


Degree Distribution of Complete Graph

Degree Sequence

X-axis = 1: max degree

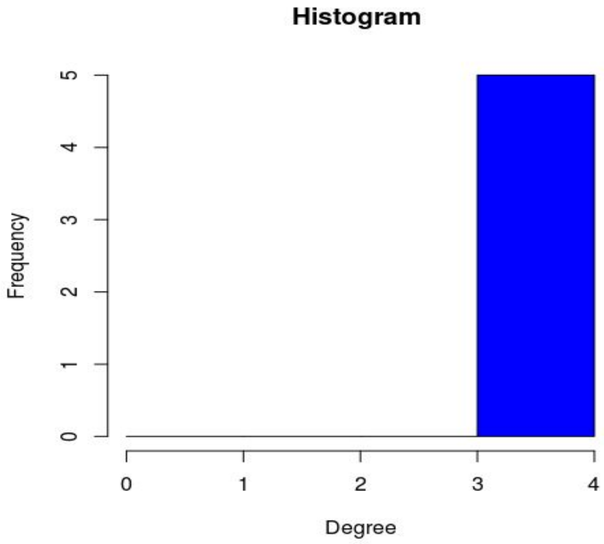
```
> hist (degree, breaks=1: vcount (network)-1,
main="Histogram", xlab ="Degree", col="blue")
```

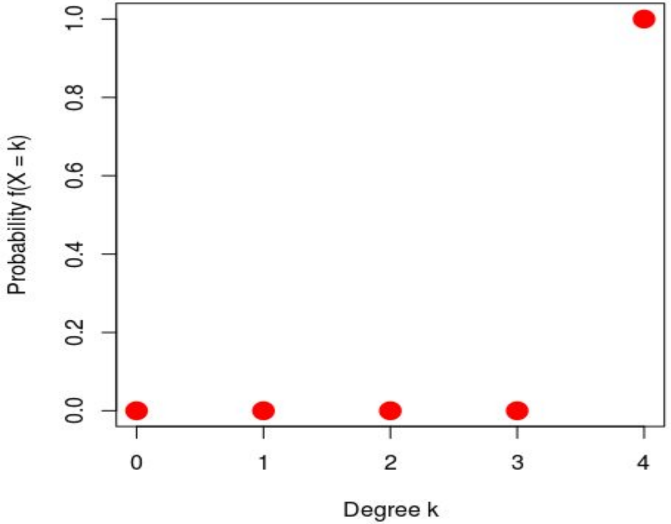


k	0	1	2	3	4
N_k	0	0	0	0	5
$F_k = N_k / N$	0	0	0	0	1

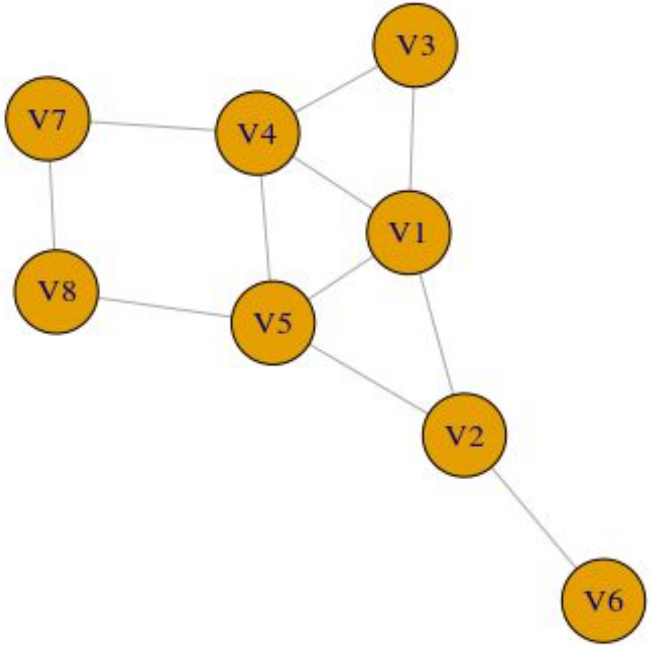
```
> deg.dist <- degree_distribution (g, cumulative=FALSE,
mode="all")
```

```
> plot ( x=0 : max(degree), y=deg.dist, pch=19, cex=2.0,
col="red",xlab="Degree k", ylab="Probability f(X = k)")
```





Density of Network (existing edges/possible edges)




```
> graph.density ( network)
```

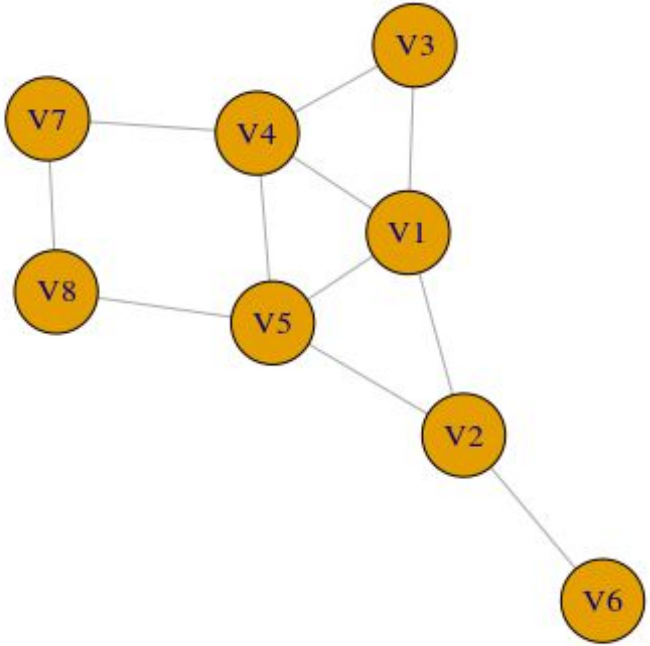
OUTPUT :

0.3928571

- Possible edges in undirected graphs = N(N - 1)/2
- Existing edges in graph = 11

Density = 11 / [8(8-1)/2]
= 0.3928571

Connected Components




```
> components ( network)
```

OUTPUT :

```
$membership
V1 V2 V3 V4 V5 V6 V7 V8
 1  1  1  1  1  1  1  1
```

Gives the component id to which each vertex belongs.

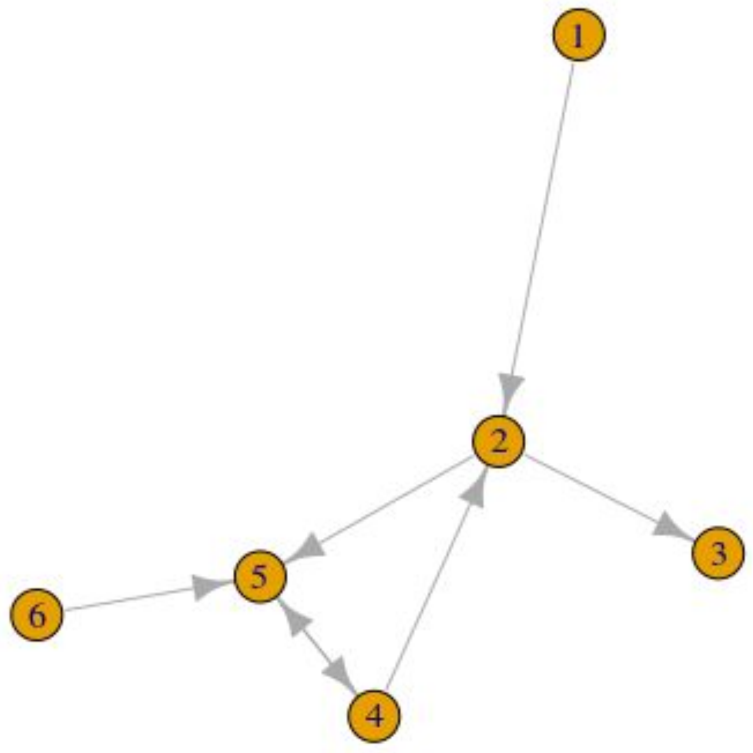
```
$csize
[1] 8
```

Gives the sizes of the component.

```
$no
[1] 1
```

Number of components.

Connected Components in Directed Graph



> ***components*** (network , mode="**strong**")

OUTPUT :

$membership
[1] 2 3 4 3 3 1

Gives the component id to which each vertex belongs.

$csize
[1] 1 1 3 1

Gives the sizes of the component.

$no
[1] 4

Number of components.

> ***components*** (network , mode="**weak**")

OUTPUT :

$membership
[1] 1 1 1 1 1 1

Gives the component id to which each vertex belongs.

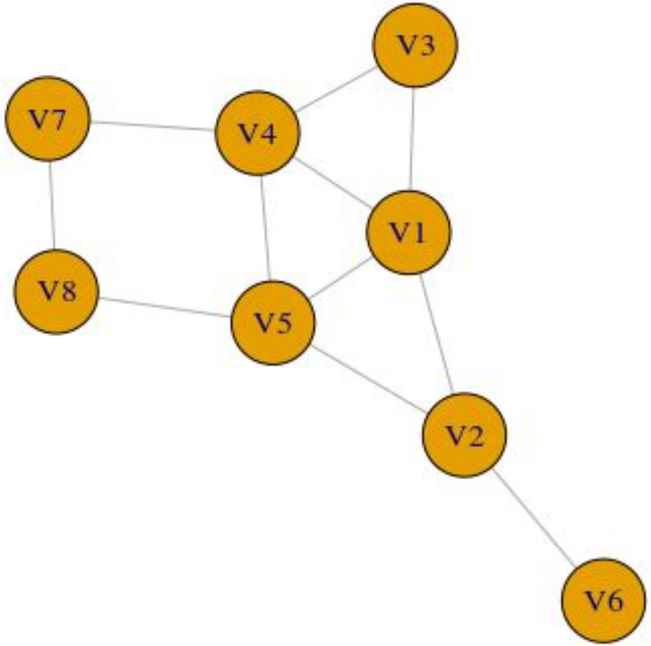
$csize
[1] 6

Gives the sizes of the component.

$no
[1] 1

Number of components.

Eccentricity (shortest distance from the farthest other node in the graph)




```
> eccentricity ( network )
```

OUTPUT :

	V1	V2	V3	V4	V5	V6	V7	V8
Eccentricity:	2	3	3	3	2	4	4	3

Diameter of graph

```
> diameter ( network )  #max eccentricity
```

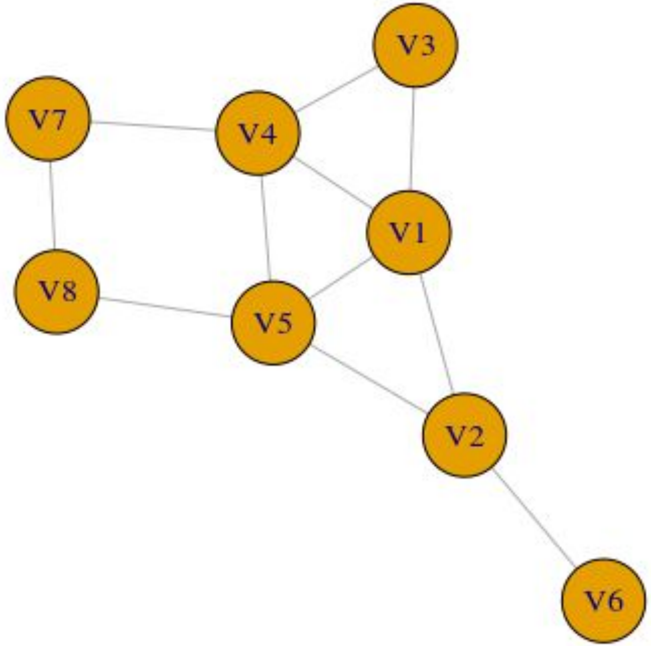
OUTPUT : 4

Radius of graph

```
> radius ( network )  #min eccentricity
```

OUTPUT : 2

Global Clustering Coefficient *("how many of my friends are friends")*

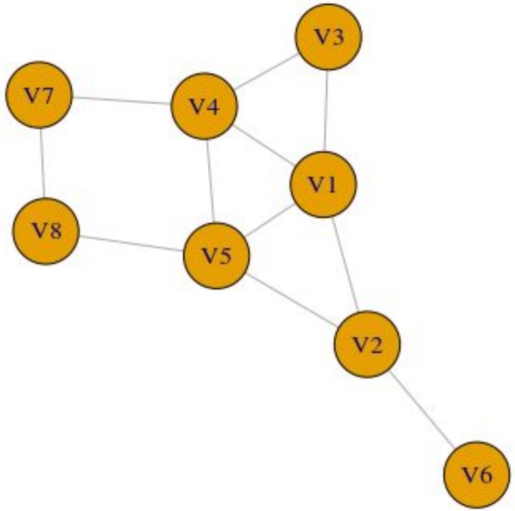



```
> transitivity ( network)   #Clustering coefficient of network
```

OUTPUT :

0.375

Average Path Length



> *distances* (network) or shortest.paths (network)

OUTPUT :

Path Length :

	V1	V2	V3	V4	V5	V6	V7	V8
V1	**0**	1	1	1	1	2	2	2
V2	1	**0**	2	2	1	1	3	2
V3	1	2	**0**	1	2	3	2	3
V4	1	2	1	**0**	1	3	1	2
V5	1	1	2	1	**0**	2	2	1
V6	2	1	3	3	2	**0**	4	3
V7	2	3	2	1	2	4	**0**	1
V8	2	2	3	2	1	3	1	**0**

Average path distance:

Let N = | V | be the number of nodes.

$$< D > = \frac{\sum_i \sum_{j>i} dist(Vi,Vj)}{\binom{N}{2}}$$

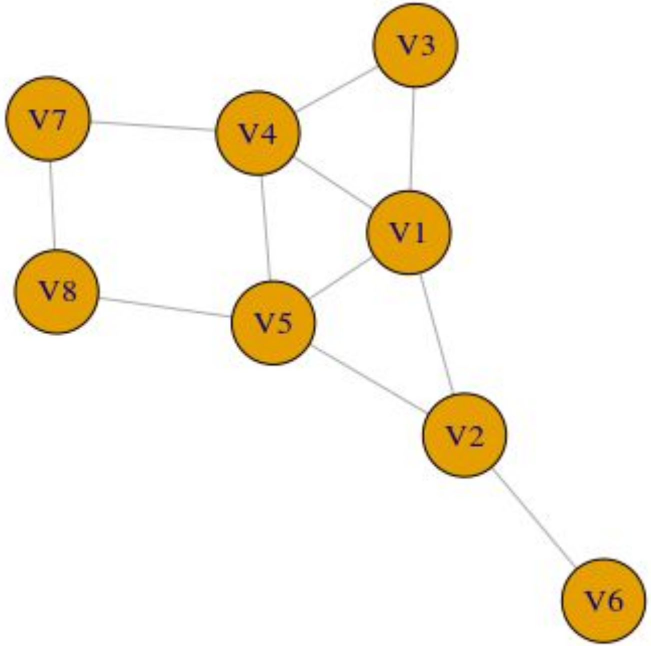
> *average.path.length* (network)

OUTPUT : 1.857

Average Path Length <D> = 52 / 8C2

= 1.857

Efficiency of graph



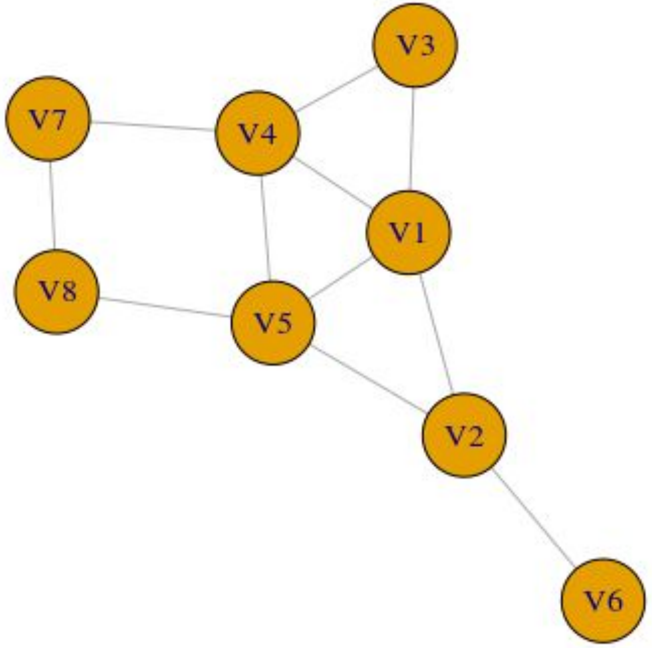

```
> efficiency ( network, , type = c("global"))   #efficiency of network
```

OUTPUT :

0.6577381

PART 3: More graph functions

Neighbors of nodes




```
> neighbors (network, "V1")  # neighbours of vertex V1
```

OUTPUT :

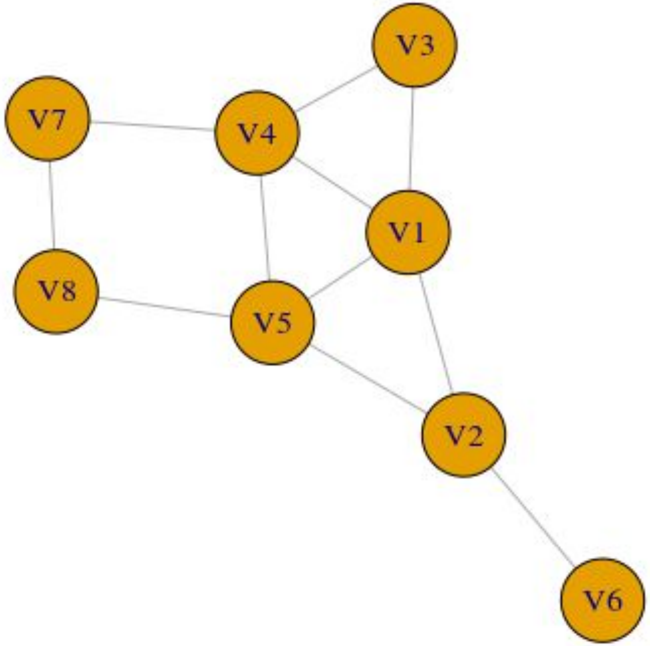
+ 4/8 vertices, named :
[1] V2 V3 V4 V5

```
> incident ( network, "V1", mode=c("all"))   #incident edges of vertex V1
```

OUTPUT :

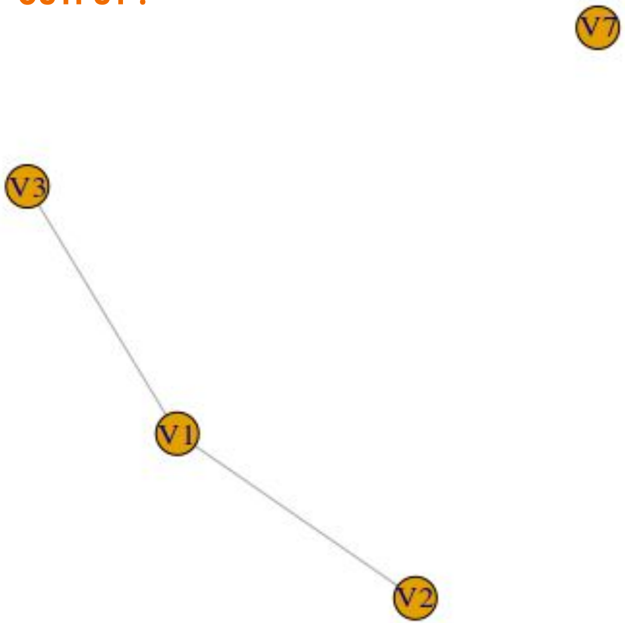
+ 4/11 edges (vertex names):
[1] V1--V2 V1--V3 V1--V4 V1--V5

Induced Subgraph

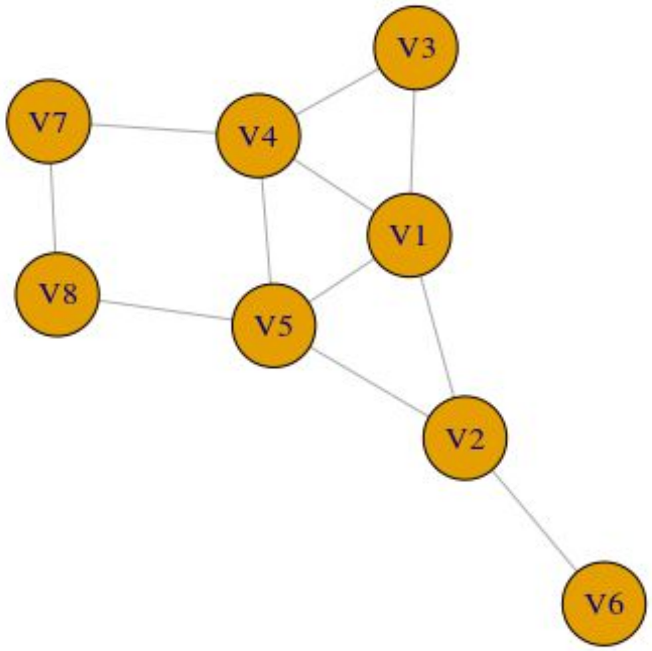



```
> sub = induced_subgraph ( network, c(1,2,3,7))
> plot  (sub)
```

OUTPUT :



Cliques (complete subgraphs of an undirected graph)



> ***cliques*** (network, min = 3, max = NULL)

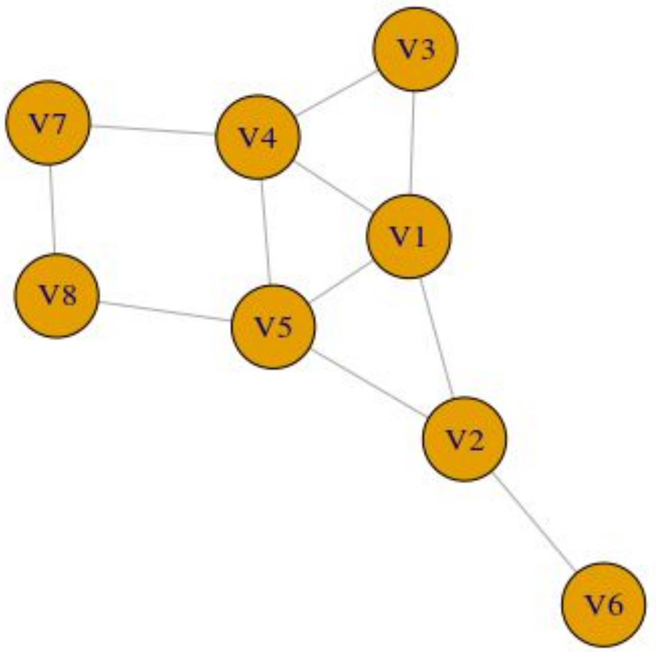
OUTPUT :

[[1]]
+ 3/8 vertices, named :
[1] V1 V3 V4

[[2]]
+ 3/8 vertices, named :
[1] V1 V4 V5

[[3]]
+ 3/8 vertices, named :
[1] V1 V2 V5

Find the shortest path between specific nodes.

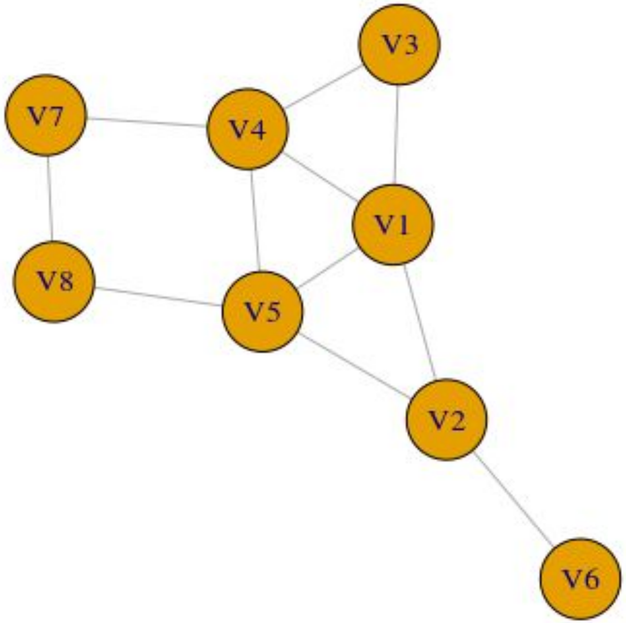



```
> get.shortest.paths ( network, "V1", "V8")
```

OUTPUT :

```
$vpath[[1]]
+ 3/8 vertices, named :
[1] V1 V5 V8
```

All Simple paths




```
> all_simple_paths(g, "V1", "V5")

# lists all simple paths between V1 and V5.
```

OUTPUT :

[[1]]
+ 3/8 vertices, named :
[1] V1 V2 V5

[[2]]
+ 4/8 vertices, named :
[1] V1 V3 V4 V5

[[3]]
+ 6/8 vertices, named :
[1] V1 V3 V4 V7 V8 V5

[[4]]
+ 3/8 vertices, named :
[1] V1 V4 V5

[[5]]
+ 5/8 vertices, named :
[1] V1 V4 V7 V8 V5

[[6]]
+ 2/8 vertices, named :
[1] V1 V5

Thankyou !