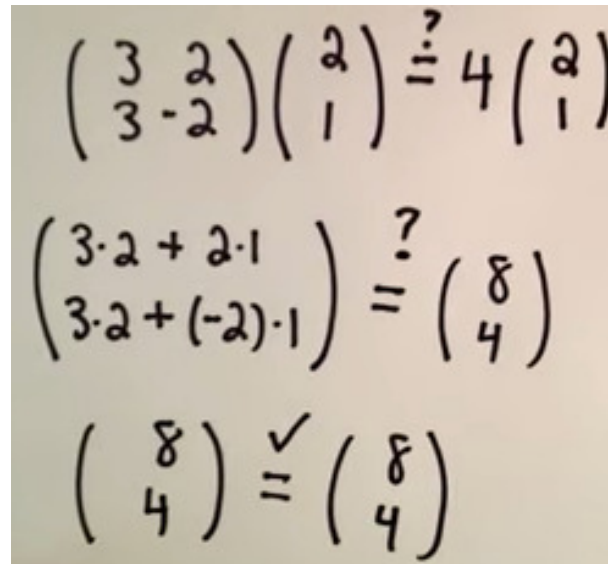# Spectral Analysis

Dr. Natarajan Meghanathan
Professor
Department of Computer Science
Jackson State University, Jackson, MS
E-mail: natarajan.meghanathan@jsums.edu

Eigenvalue and Eigenvector

- Let A be an nxn matrix.
- A scalar λ is called an Eigenvalue of A if there is a non-zero vector X such that $AX = \lambda X$. Such a vector X is called an Eigenvector of A corresponding to λ.
- Example: $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is an Eigenvector of $A = \begin{bmatrix} 3 & 2 \\ 3 & -2 \end{bmatrix}$ for $\lambda = 4$

$$\begin{pmatrix} 3 & 2 \\ 3 & -2 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \overset{?}{=} 4 \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 3 \cdot 2 + 2 \cdot 1 \\ 3 \cdot 2 + (-2) \cdot 1 \end{pmatrix} \overset{?}{=} \begin{pmatrix} 8 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 8 \\ 4 \end{pmatrix} \overset{\checkmark}{=} \begin{pmatrix} 8 \\ 4 \end{pmatrix}$$

Finding Eigenvalues and Eigenvectors

$$A = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$$

① $\lambda I = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$

② $A - \lambda I = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$

$= \begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix}$

③ $\det \begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix}$

$= (7-\lambda)(-1-\lambda) - (3)(3)$

$= -7 - 7\lambda + \lambda + \lambda^2 - 9$

$= \lambda^2 - 6\lambda - 16$

(4) Solving for λ:

$(\lambda - 8)(\lambda + 2) = 0$

λ = 8 and λ = -2 are the Eigen values

(5) Consider A – λ I

⑤ $\begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix}$

$\lambda = 8$:

$\begin{bmatrix} 7-8 & 3 \\ 3 & -1-8 \end{bmatrix} = \begin{bmatrix} -1 & 3 \\ 3 & -9 \end{bmatrix}$ = B

Solve B X = 0

$$\begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix} \begin{pmatrix} X1 \\ X2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

-X1 + 3X2 = 0 ⟶ X1 = 3X2

3X1 – 9X2 = 0 ⟶ 3X1 = 9X2 ➔ X1 = 3X2

If X2 = 1;

X1 = 3

$\begin{bmatrix} 3 \\ 1 \end{bmatrix}$ is an eigenvector for **λ = 8**

Finding Eigenvalues and Eigenvectors

For $\lambda = -2$

$$\begin{bmatrix} 7-(-2) & 3 \\ 3 & -1-(-2) \end{bmatrix} = \begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix}$$

Solve B X = 0

$$\begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

9X1 + 3X2 = 0 $\longrightarrow$ X2 = - 3X1

3X1 + X2 = 0 $\longrightarrow$ X2 = - 3X1

If X1 = 1; $\begin{bmatrix} 1 \\ -3 \end{bmatrix}$ is an eigenvector

X2 = -3 for **λ = -2**

Verification   A = $\begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$

AX = λX

For $\lambda = 8$ and X = $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$$\begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 24 \\ 8 \end{bmatrix} = 8 \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

An *n*x*n* matrix has *n* eigenvalues and the corresponding eigenvectors.

Spectral Analysis

- Spectral decomposition is a method of projecting the characteristics of a network graph in n-dimensions or directions (that are mutually perpendicular) where n is the number of vertices in the graph.

- The projection in each direction is represented in the form of a scalar value (called the eigenvalue) and its corresponding vector with entries for each vertex (called the eigenvector).

- The largest eigenvalue of the projection is called the principal eigenvalue (a.k.a. spectral radius) and the corresponding eigenvector is called the principal eigenvector.

- We will use the adjacency matrix of a network graph to determine its eigenvalues and eigenvectors.

Example: Eigenvalues and Eigenvectors
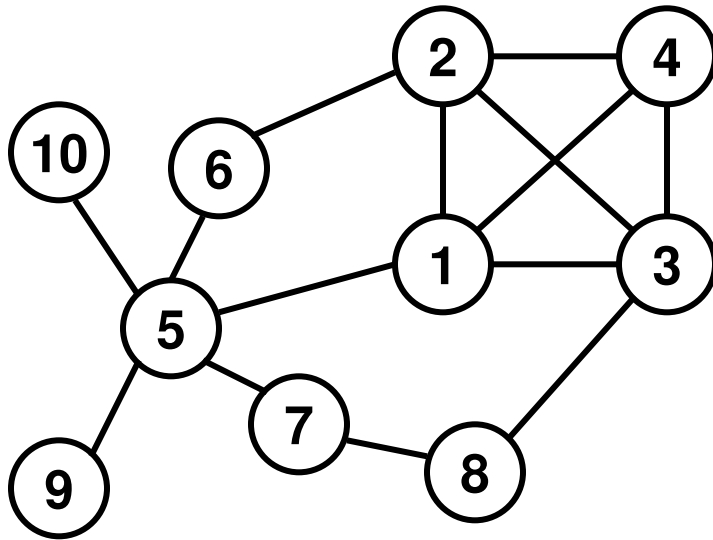
List of edges:
```
1    2
1    3
1    4
1    5
2    3
2    4
2    6
3    4
3    8
5    6
5    7
5    9
5    10
7    8
```

List of edges written in the increasing order of the node IDs from left to right and top to bottom.

<u>**Property:**</u> **For a matrix with all positive entries, the Principal Eigenvalue is always positive & the entries in the Principal Eigenvector are also positive**

Node IDs ← **Eigenvalues** →

|	3.3893	1.888	1.1131	0.3197	0	-0.3741	-0.8563	-1.3098	-1.6828	-2.4871
1	0.4741	0.0641	-0.1261	0.2899	0.0000	-0.3898	0.5219	0.0776	0.3210	-0.3711
2	0.4568	0.2043	-0.2089	-0.3262	0.0000	0.0872	-0.2158	0.6614	0.1384	0.2979
3	0.4453	0.2648	0.2600	0.1054	0.0000	0.3619	0.2312	-0.0638	-0.6797	-0.0477
4	0.4060	0.2824	-0.0674	0.2161	0.0000	-0.1585	-0.6275	-0.5155	0.1309	0.0486
5	0.2986	-0.6305	-0.1241	0.0974	0.0000	-0.1448	0.1651	-0.1838	-0.1298	0.6242
6	0.2229	-0.2257	-0.2991	-0.7157	0.0000	0.1538	0.0591	-0.3646	-0.0051	-0.3708
7	0.1390	-0.3609	0.5100	-0.1521	0.0000	-0.4838	-0.3369	0.2471	-0.2518	-0.3086
8	0.1724	-0.0509	0.6917	-0.1460	0.0000	0.3258	0.1234	-0.1399	0.5536	0.1433
9	0.0881	-0.3339	-0.1114	0.3046	0.7071	0.3870	-0.1928	0.1403	0.0771	-0.2510
10	0.0881	-0.3339	-0.1114	0.3046	-0.7071	0.3870	-0.1928	0.1403	0.0771	-0.2510

Each column corresponds to an Eigenvector with an entry for each node

Spectral Radius Ratio for Node Degree

- Spectral radius ratio for node degree (for an undirected graph) is the ratio of the principal eigenvalue of the adjacency matrix and the average degree of the vertices in the graph.
- If Kmin, Kavg and Kmax are the minimum, average and maximum values for the node degrees, then:
    - Kmin ≤ Kavg ≤ Principal Eigenvalue ≤ Kmax
- So, the spectral radius ratio for node degree of a graph is always greater than or equal to 1.

- The spectral radius ratio for node degree is a measure of the variation in the node degrees. The farther is the value from 1, the larger the variation in node degree.

- The spectral radius ratio for node degree can be uniformly applied across networks of all size and be used to evaluate the relative variation in node degree.

Spectral Radius Ratio for Node Degree
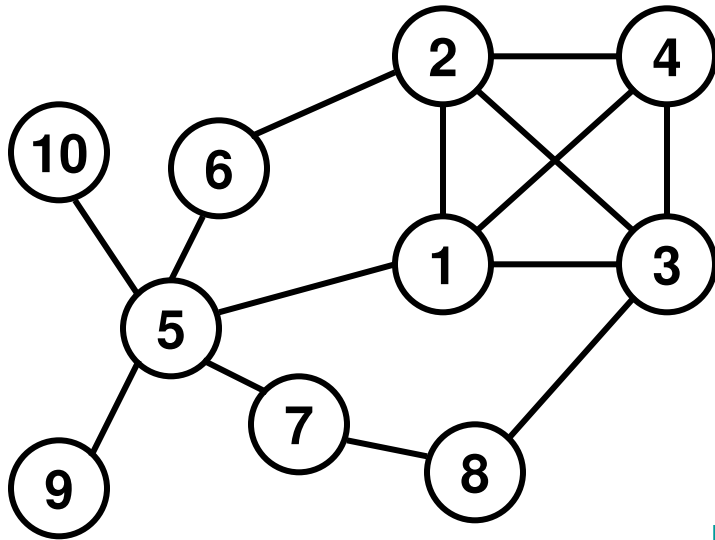
Avg. Degree = Sum of Node Degrees / N
= (4 + 4 + 4 + 3 + 5 + 2 + 2 + 2 + 1 + 1)/10
= 2.8

Principal Eigenvalue = 3.3893
Spectral Radius Ratio for Node Degree
= 3.3893 / 2.8 = 1.21
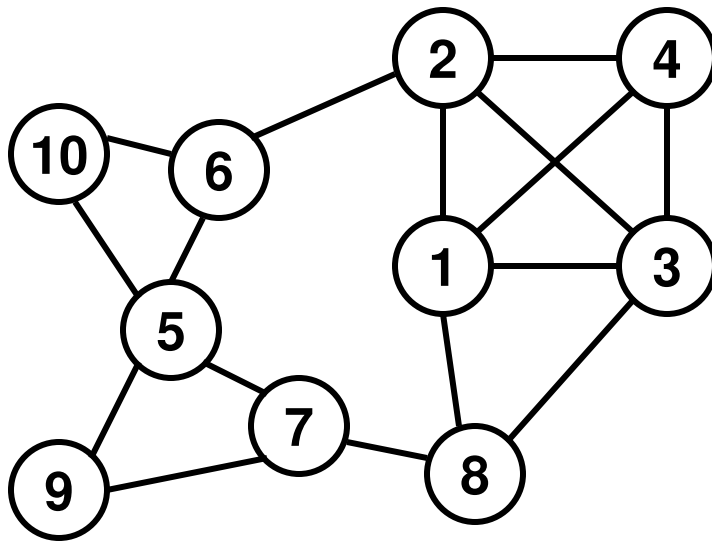
1	2
1	3
1	4
1	8
2	3
2	4
2	6
3	4
3	8
5	6
5	7
5	9
5	10
6	10
7	8
7	9

Avg. Degree = Sum of Node Degrees / N
= (4 + 4 + 4 + 3 + 4 + 3 + 3 + 3 + 2 + 2)/10
= 3.2

Principal Eigenvalue = 3.4735
Spectral Radius Ratio for Node Degree
= 3.4735 / 3.2 = 1.09

Applications

- Bipartivity Index: Detection of bipartivity in graphs
- Estrada Index: Protein folding
- Laplacian Matrix: Determining the number of components, connectivity and number of spanning trees in a graph
- Hierarchical Community Detection
- Prediction of Graph Isomorphism

Eigen Values of a Bipartite Network

- There are even number of vertices.
- Let λ1, λ2, λ3, …, λn be the n Eigenvalues of an n-node network.
- For any $j = 1, 2, …, n/2$, if $\lambda_j = |-\lambda_{n-j+1}|$, then the network is almost bi-partite.
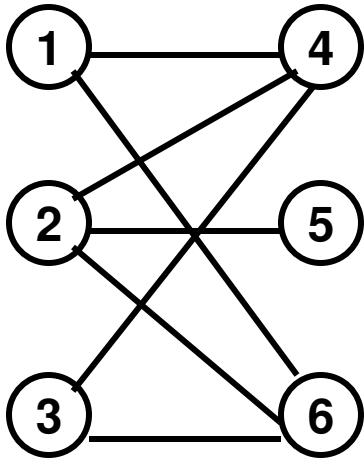
$$\begin{matrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{matrix}$$

Eigenvalues
-2.5243
-0.7923
0
0
0.7923
2.5243

Bipartivity Check

- A bi-partite graph is the one that has two partitions V1 and V2 of its vertices such that
  - V1 U V2 = V; V1 n V2 = Φ.
  - No edges within V1 and within V2.
  - All edges are those connecting V1 and V2.
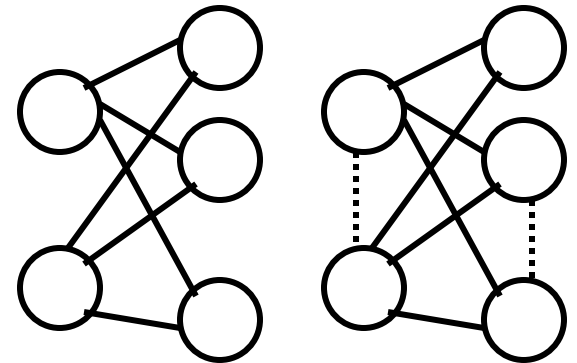
  There should be NO odd length cycles in a truly bi-partite network

- Graphs can be considered close to bi-partite if there are few edges (not a significant number) called the frustrated edges that connect vertices within V1 and/or V2.
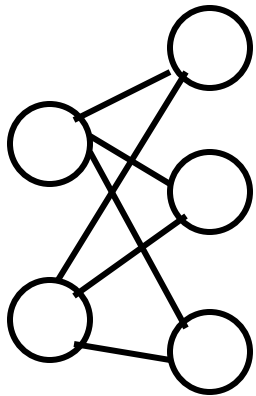
Computing the Bipartite Measure

Compute the Eigenvalues (λ1, λ2, λ3, …, λn) of the nxn adjacency matrix.

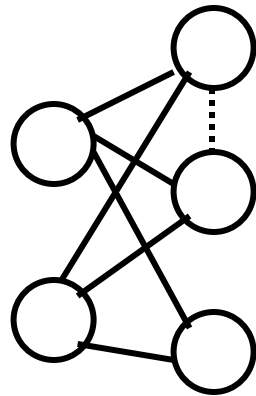$$b_S(G) = \frac{\sum_{j=1}^{n} \cosh(\lambda_j)}{\sum_{j=1}^{n} \cosh(\lambda_j) + \sum_{j=1}^{n} \sinh(\lambda_j)}$$

For a "truly" bi-partite graph, $b_S(G) = 1$; the sinh terms add to 0.

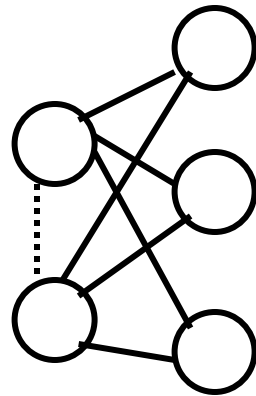**For a "close-to" bi-partite graph, $b_S(G) < 1$; the sinh terms add up to some small positive value.**

Bipartivity Measure: Examples

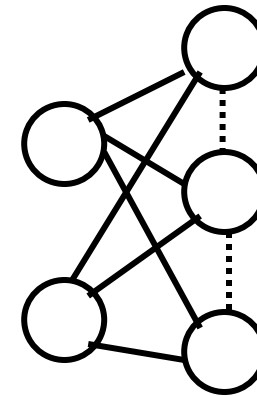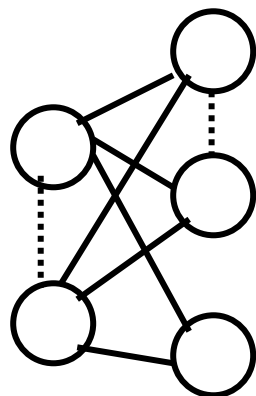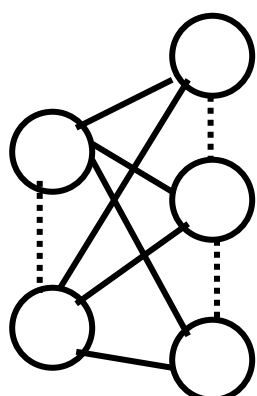

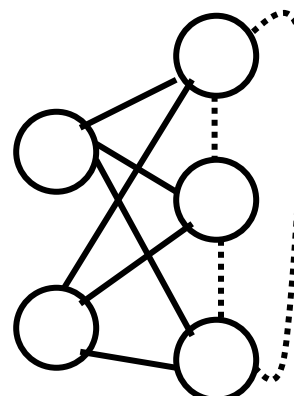$b_S(G) = 1.0$ $b_S(G) = 0.829$ $b_S(G) = 0.769$ $b_S(G) = 0.731$
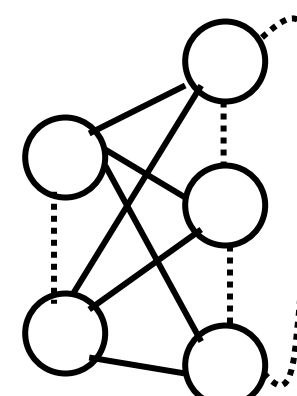
$b_S(G) = 0.692$ $b_S(G) = 0.645$ $b_S(G) = 0.645$ $b_S(G) = 0.597$

For a given number of frustrated links, a larger bipartivity measure is observed if more of the frustrated links are present in the network with the larger subset.

Bipartivity of Real Networks

Type: Information

Network	Bipartivity Measure
SciMet	0.500
Roget	0.529

Type: Social

Network	Bipartivity Measure
Drugs	0.500
Corporate Elite	0.500
Karate Club	0.597
Saw Mill	0.749

Type: Food webs

Network	Bipartivity Measure
Coachella	0.500
El Verde	0.500
Grassland	0.743
Stony stream	0.815

Type: PPIs

Network	Bipartivity Measure
Yeast	0.500
Human	0.576
H. Pylori	0.711
A. Fulgidus	0.976

Type: Transcription

Network	Bipartivity Measure
Urchin	0.618
E. Coli	0.831
Yeast	0.960

Type: Technological

Network	Bipartivity Measure
USAir97	0.500
Internet	0.502
Electronic3	0.952

Identifying Bipartite Subsets using Eigenvalue and Eigenvector

- We identify the smallest Eigenvalue (most likely a negative value), hereafter called the bi-partite Eigenvalue, and its corresponding Eigenvector, hereafter called the bi-partite Eigenvector.

- The values in the bi-partite Eigenvector will be positive and negative.
  - The node IDs whose entries are of the same sign in the bi-partite Eigenvector form the two subsets.
    - The vertices that are of the same sign are more likely not to have links between them, and are more likely to have links with vertices of the other sign.
  - Each of the two subsets will have the minimum (or zero, if possible) number of frustrated links. Most of the links are likely to be between the vertices in the two subsets.

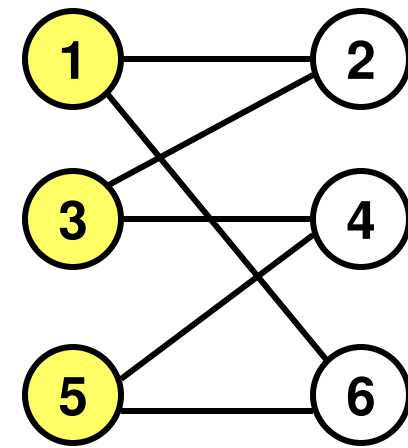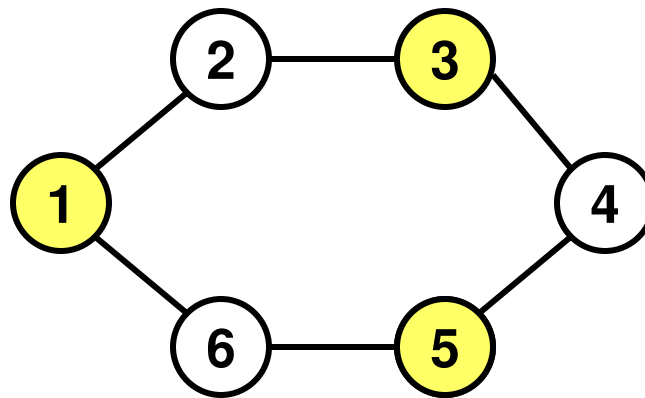# Identifying Bipartite Subsets using Eigenvalue and Eigenvector: Ex. 1 (1)
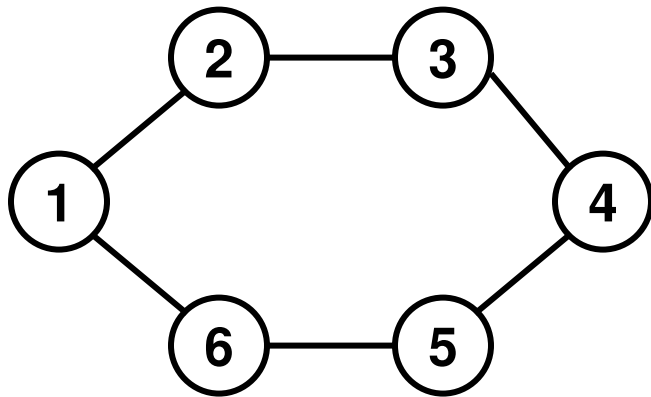
	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	1	0	0
4	0	0	1	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

1	2
1	6
2	3
3	4
4	5
5	6

Based on the lowest Negative Eigenvalue -2:

1	-0.4082
2	0.4082
3	-0.4082
4	0.4082
5	-0.4082
6	0.4082

Identifying Bipartite Subsets using Eigenvalue and Eigenvector: Ex. 1 (2)

Eigenvalue, λ	cosh(λ)	sinh(λ)
-2	3.7622	-3.6269
-1	1.5431	-1.1752
-1	1.5431	-1.1752
1	1.5431	1.1752
1	1.5431	1.1752
2	3.7622	3.6269

Total	13.6968	0

$$b_S(G) = \frac{\displaystyle\sum_{j=1}^{n} \cosh(\lambda_j)}{\displaystyle\sum_{j=1}^{n} \cosh(\lambda_j) + \sum_{j=1}^{n} \sinh(\lambda_j)}$$

$$b_S(G) = \frac{13.6968}{(13.6968 + 0)}$$

$$= 1.0$$

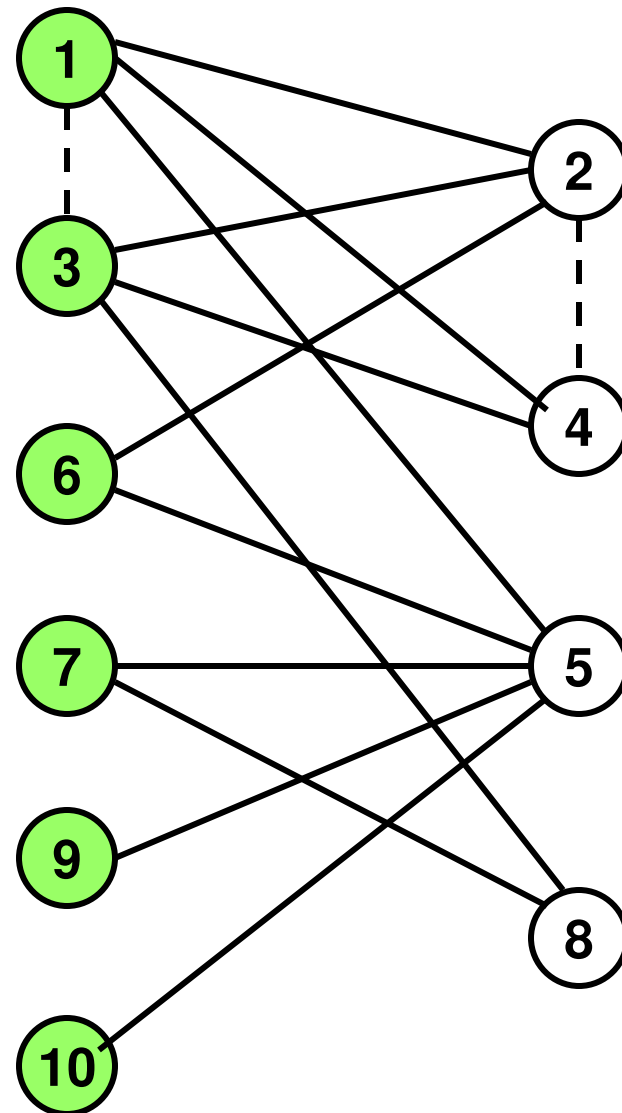# Identifying Bipartite Subsets using Eigenvalue and Eigenvector: Ex. 2 (1)
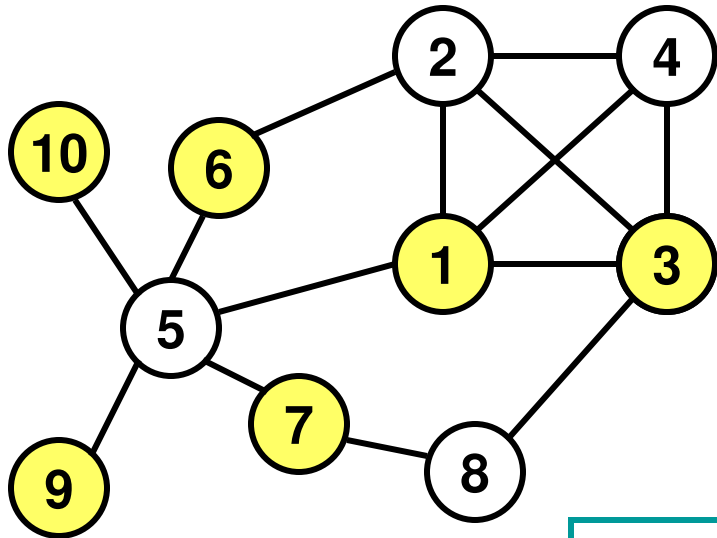
	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	0	0	0	0	0
2	1	0	1	1	0	1	0	0	0	0
3	1	1	0	1	0	0	0	1	0	0
4	1	1	1	0	0	0	0	0	0	0
5	1	0	0	0	0	1	1	0	1	1
6	0	1	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0
8	0	0	1	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0

1	2
1	3
1	4
1	5
2	3
2	4
2	6
3	4
3	8
5	6
5	7
5	9
5	10
7	8

Identifying Bipartite Subsets using Eigenvalue and Eigenvector: Ex. 2 (2)

Based on the lowest Negative Eigenvalue -2.4870836366555165:

1 0.3711379191456978
2 -0.2979483534591263
3 0.04772178019913417
4 -0.04861571364294786
5 -0.6242087587467021
6 0.3707784887547619
7 0.3085825736313854
8 -0.14326191068896146
9 0.25098020410206323
10 0.250980204102063

Identifying Bipartite Subsets using Eigenvalue and Eigenvector: Ex. 2 (3)

Eigenvalue, λ	cosh(λ)	sinh(λ)
-2.4871	6.0547	-5.9716
-1.6828	2.7832	-2.5974
-1.3098	1.9877	-1.7178
-0.8564	1.3897	-0.9650
-0.3741	1.0708	-0.3829
0	1.0	0
0.3197	1.0515	0.3252
1.1131	1.6862	1.3576
1.8880	3.3788	3.2274
3.3893	14.839	14.8057
--		
Total	35.2416	8.0812

$$b_S(G) = \frac{\sum_{j=1}^{n} \cosh(\lambda_j)}{\sum_{j=1}^{n} \cosh(\lambda_j) + \sum_{j=1}^{n} \sinh(\lambda_j)}$$

$$b_S(G) = \frac{35.2416}{(35.2416 + 8.0812)}$$

$$= 0.8135$$

Bipartite Partition Detection: Digraph

- When confronted with a directed graph, first transform the directed graph to an undirected graph and determine the two partitions as explained previously using the Eigenvector approach.

- After identifying the partitions, restore the directions of the edges.

- In a directed graph, the edges typically point from one set of vertices to the other set of vertices.

  – There could be scenarios where the edges could point in the reverse direction; as long as we know the direction of the edges, we could restore them after determining the two partitions.
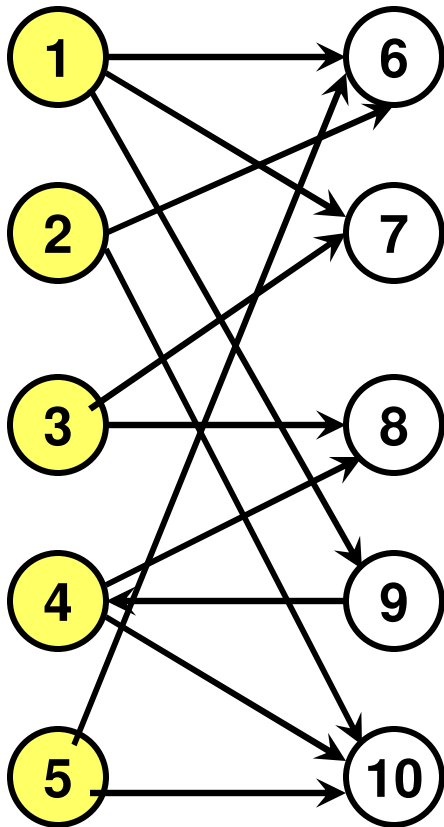
Digraph Bipartivity Detection: Example (1)

Digraph Bipartivity Detection: Example (2)

Based on the lowest Negative Eigenvalue -2.4998:

1	-0.362595131299419
2	-0.32242994615264337
3	-0.17066751642864314
4	-0.362595131299418
5	-0.32242994615264337
6	0.40300972011359426
7	0.21331972680920852
8	0.21331972680920852
9	0.29009605184492576
10	0.4030097201135942

Protein Folding

- Protein folding is the process by which a protein transforms from a random coil (sequence of amino acids: linear polypeptide chain) to its characteristic 3-dimensional structure that is essential to its expected function.

- The correct three-dimensional structure is essential to function, although some parts of functional proteins may remain unfolded.

- Failure to fold into native structure generally produces inactive proteins, but in some instances misfolded proteins have modified or toxic functionality.

- When modeled as a graph, the more flat (linear chain) is the graph, the less the folding and vice-versa.
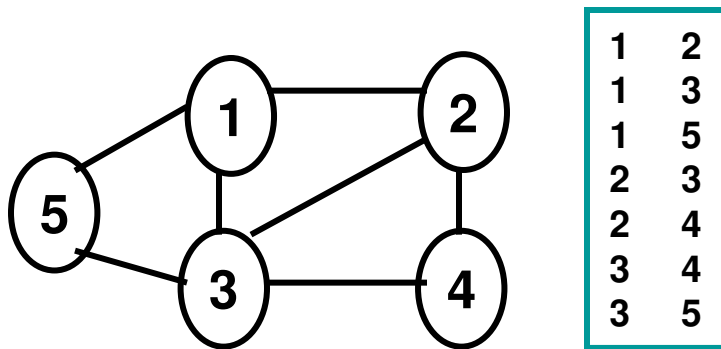
Estrada, E. (2000). "Characterization of 3D molecular structure". *Chem. Phys. Lett.* **(319): 713**

Source: Wikipedia
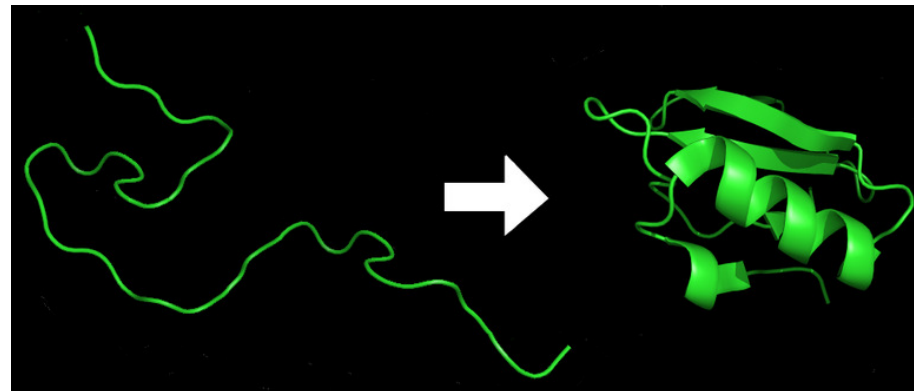
Estrada Index of Graphs

- The Estrada Index can be used to determine the degree of folding of a protein.
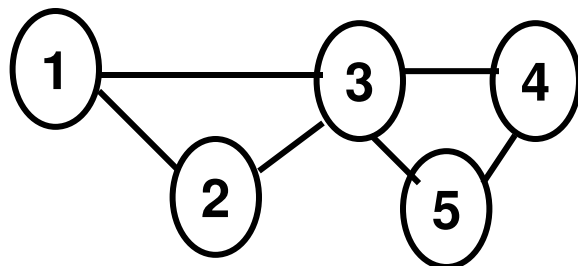 - Larger the Estrada Index, the larger the folding.

$$EE(G) = \sum_{j=1}^{n} e^{\lambda_j}$$

EE(G) = 21.56
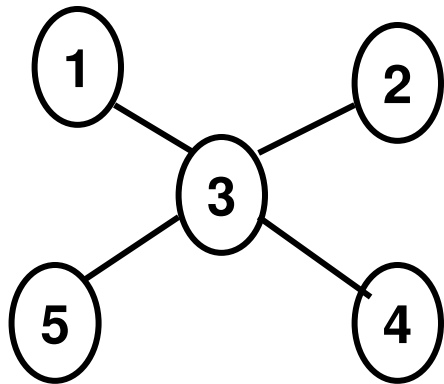
1	2
1	3
1	5
2	3
2	4
3	4
3	5

$$A = \begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{matrix}$$

e^{λ_j}

$\lambda 1 = -1.618$ 0.2
$\lambda 2 = -1.473$ 0.23
$\lambda 3 = -0.463$ 0.63
$\lambda 4 = 0.618$ 1.852
$\lambda 5 = 2.935$ 18.654

1	2
1	3
2	3
3	4
3	5
4	5

$$\begin{matrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{matrix}$$

e^{λ_j}

$\lambda 1 = -1.5616$ 0.211
$\lambda 2 = -1$ 0.369
$\lambda 3 = -1$ 0.369
$\lambda 4 = 1$ 2.71
$\lambda 5 = 2.5616$ 12.856

EE(G) = 16.51

Estrada Index of Star and Chain

Star graph (top):

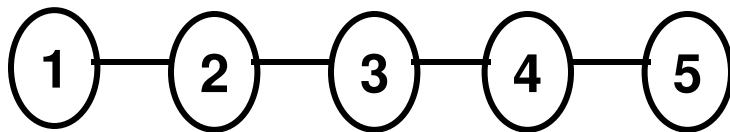Edge list:
1	3
2	3
3	4
3	5

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

	e^{λ_j}
$\lambda 1 = -2$	0.136
$\lambda 2 = 0$	1
$\lambda 3 = 0$	1
$\lambda 4 = 0$	1
$\lambda 5 = 2$	7.344

EE(G) = 10.48

$$EE(G) = \sum_{j=1}^{n} e^{\lambda_j}$$

Chain graph (bottom):
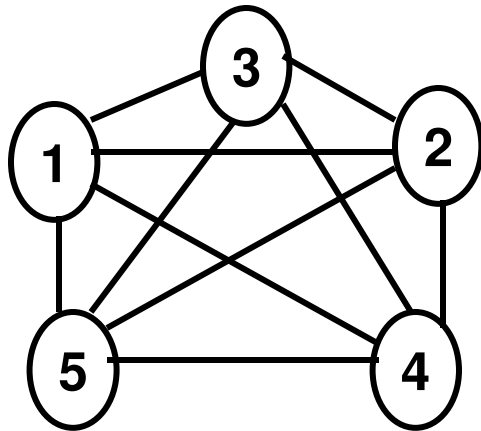
Edge list:
1	3
2	3
3	4
4	5

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

	e^{λ_j}
$\lambda 1 = -1.7321$	0.178
$\lambda 2 = -1$	0.369
$\lambda 3 = 0$	1
$\lambda 4 = 1$	2.71
$\lambda 5 = 1.7321$	5.622

EE(G) = 9.88

Estrada Index of Complete Graph

1	2
1	3
1	4
1	5
2	3
2	4
2	5
3	4
3	5
4	5

$$A = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$$

$e^{\lambda j}$

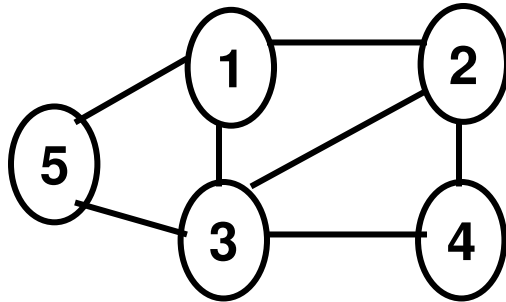|	$e^{\lambda j}$
$\lambda 1 = -1$	0.368
$\lambda 2 = -1$	0.368
$\lambda 3 = -1$	0.368
$\lambda 4 = -1$	0.368
$\lambda 5 = 4$	54.576

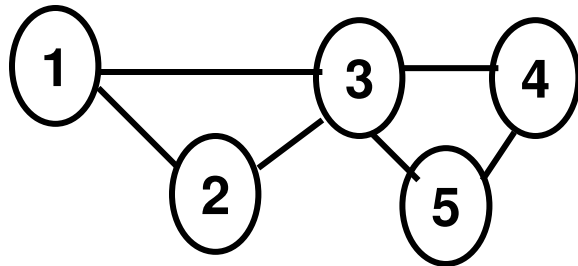**EE(G) = 56.048**

$$EE(G) = \sum_{j=1}^{n} e^{\lambda_j}$$

Foldina Effectiveness

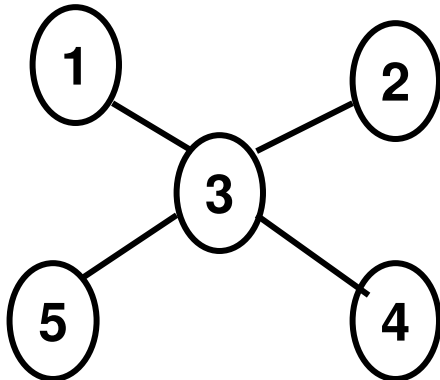$$\text{Folding Effectiveness}(G) = \frac{Estrada\ Index\ (G)}{Estrada\ Index\ Complete\ Graph\ (G)}$$

Closer is the value to 1, the more folded is the protein

Folding Effectiveness (G) = 21.56 / 56.048 = 0.385

**Folding Effectiveness (G) = 16.51 / 56.048 = 0.295**

Folding Effectiveness (G) = 10.48 / 56.048 = 0.187

Components (Clusters)

- The vertices of a graph are said to be in a single component if there is a path between the vertices.
- A graph is said to be connected if all its vertices are in one single component; otherwise, the graph is said to be disconnected and consists of multiple components.
  - Adding one or more links (bridges) can connect the different components
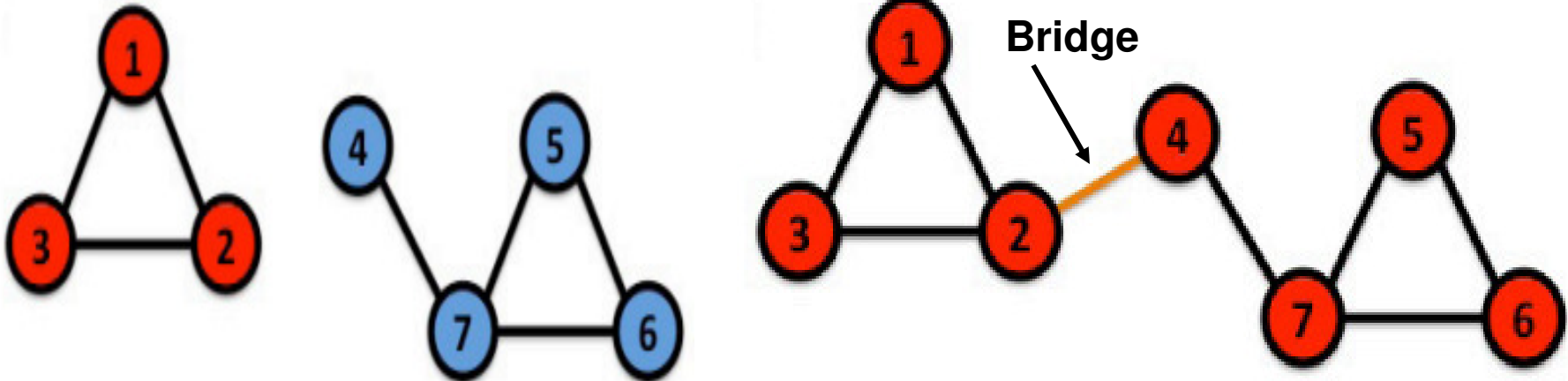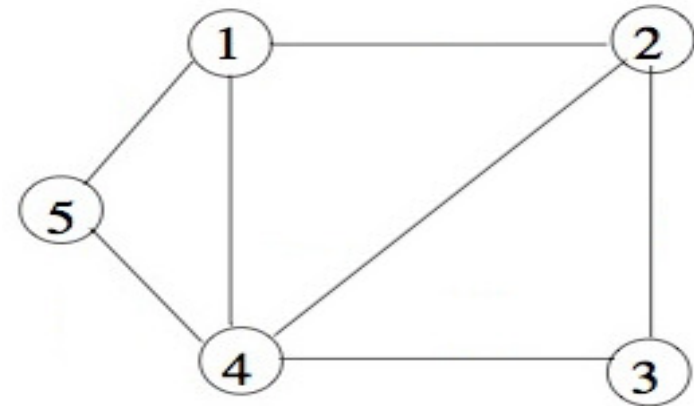
Laplacian Matrix

- Laplacian Matrix L = D – A

$$L_{ij} = \begin{cases} deg(i) & \text{for} \quad i = j \\ -a_{ij} & \text{for} \quad i \neq j \end{cases}$$

$$\begin{pmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{pmatrix}$$

The second smallest eigenvalue of the Laplacian matrix is a measure of the Connectivity of the graph and is called (**Algebraic Connectivity**).

The eigenvalues of the above Laplacian matrix are:
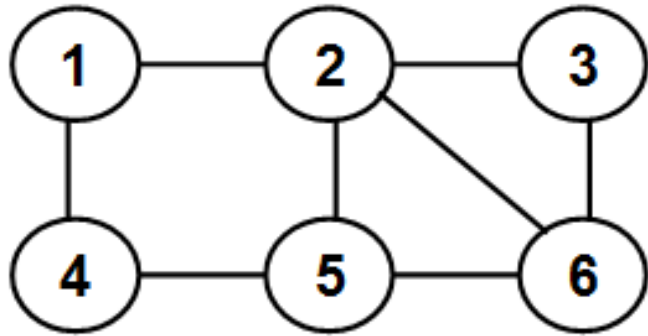0   1.586   3.0   4.414   5.0

The **n** eigenvalues of an **n** x **n** Laplacian matrix are all positive.

The first eigenvalue is always 0.

The number of 0s among the Eigenvalues of the Laplacian Matrix indicates the number of Connected components of a graph.

Max. Alg. Conn. for a graph of **n** vertices is **n** (for a complete graph).

Algebraic Connectivity

Graph 1

Algebraic Connectivity: 1.108

Traditional Connectivity: 2

Laplacian Matrix

	1	2	3	4	5	6
1	2	-1	0	-1	0	0
2	-1	4	-1	0	-1	-1
3	0	-1	2	0	0	-1
4	-1	0	0	2	-1	0
5	0	-1	0	-1	3	-1
6	0	-1	-1	0	-1	3

Eigenvalues

0

1.108

2.295

3.0

4.317

5.278

If we remove vertices 1 and 5, Graph 1 will get disconnected, but not Graph 2

Larger the algebraic connectivity of a graph, lower the chances of a network to get disconnected due to node removal.
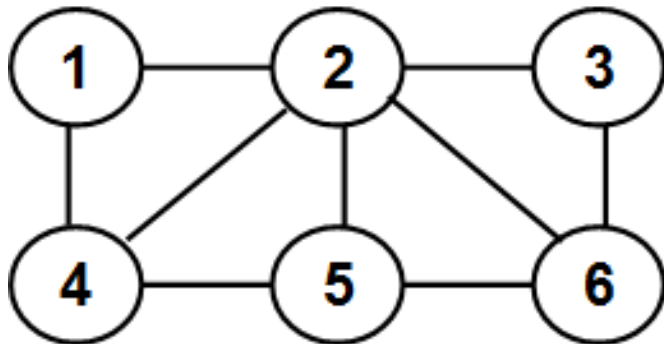
Graph 2

Algebraic Connectivity: 1.382

Traditional Connectivity: 2

Laplacian Matrix

	1	2	3	4	5	6
1	2	-1	0	-1	0	0
2	-1	4	-1	-1	-1	-1
3	0	-1	2	0	0	-1
4	-1	-1	0	2	-1	0
5	0	-1	0	-1	3	-1
6	0	-1	-1	0	-1	3

Eigenvalues

0

1.382

2.382

3.618

4.618

5.999

Laplacian Matrix
Components

The 7 eigenvalues of the Laplacian Matrix are:

0
0
1.0
3.0
3.0
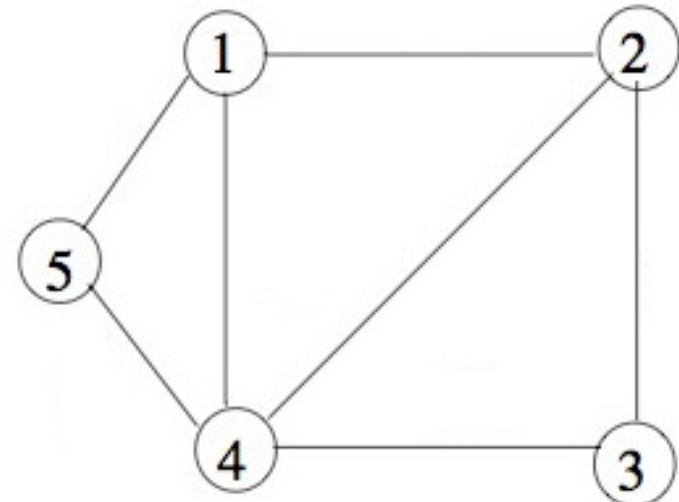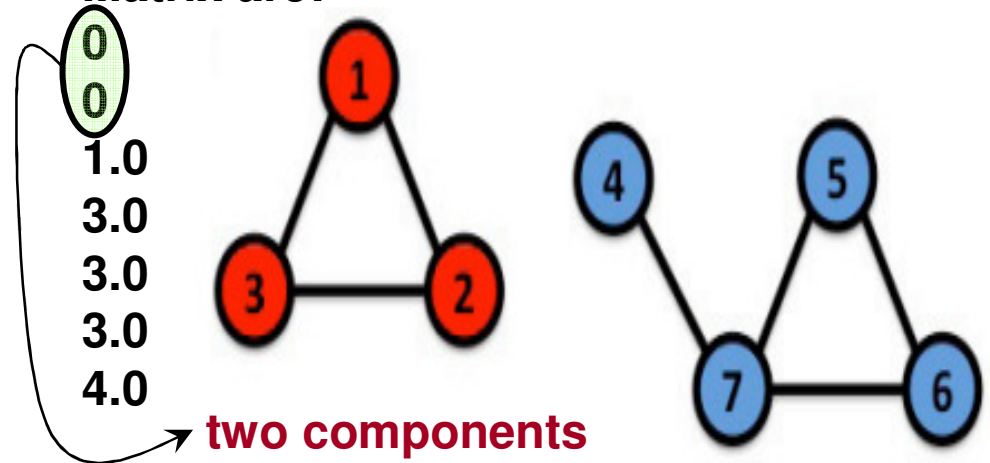3.0
4.0

two components

Laplacian Matrix
Spanning Trees

If $\mu_1 = 0 < \mu_2 \leq \mu_3 \leq \ldots \leq \mu_n$ are the n Eigenvalues of the Laplacian matrix of a connected graph of n vertices

The # spanning trees of the graph is then

$$\frac{1}{n}\prod_{i=2}^{n}\mu_i$$

**The five eigenvalues of the Laplacian matrix of the above graph are: 0, 1.586, 3.0, 4.414, 5.0**

**# spanning trees = (1/5)(1.586*3*4.414*5) = 21.**

Community

- Community: It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group.
- The paths among vertices within a community is most likely to involve only the other vertices within the community.
- For a highly modular community of vertices, the number of edges connecting them within the community is significantly greater than the number of edges connecting them to vertices outside the community.
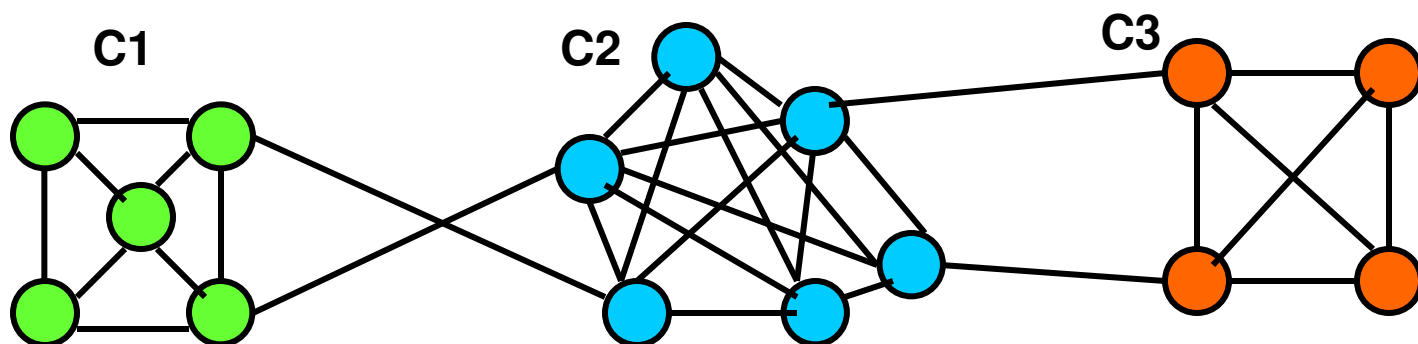
Modularity Maximization

- Modularity measures the strength of a community partition by taking into account the degree distribution.
- Given a network with m edges, the probability of an edge between two nodes i and j with degrees d_i and d_j respectively is $d_i d_j / 2m$.
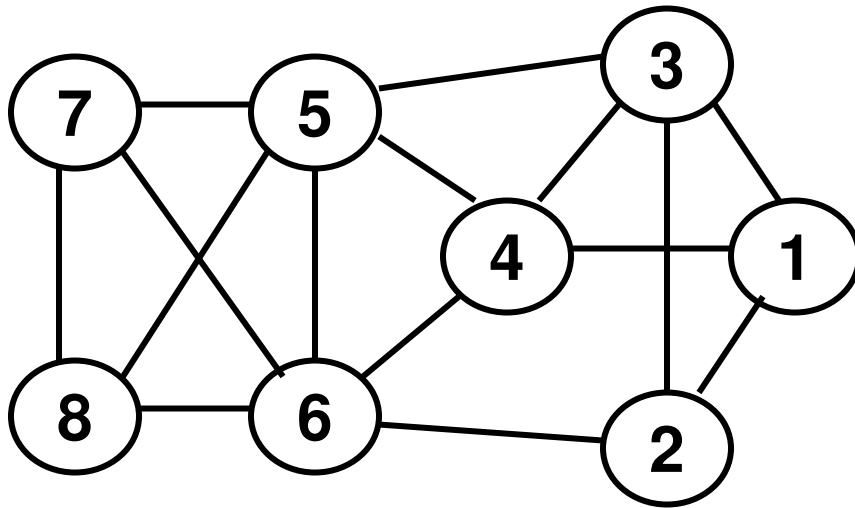
Probability of an edge between nodes 1 and 2 is $(3)(3) / (2*15) = 0.30$

Strength of a Community, C

$$\sum_{i \in C, j \in C} A_{i,j} - \frac{d_i d_j}{2m}$$

For a network with k communities and a total of m edges
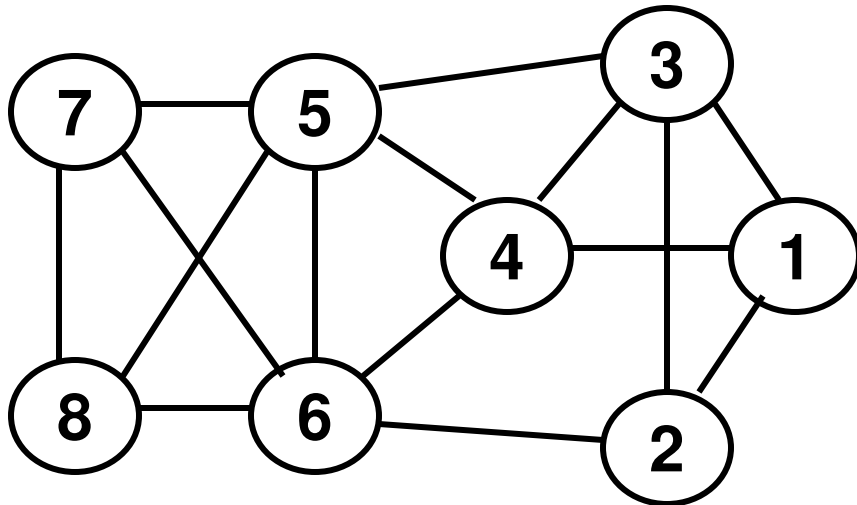
Modularity:
$$Q = \sum_{l=1}^{k} \sum_{i \in C_l, j \in C_l} A_{i,j} - \frac{d_i d_j}{2m}$$

A larger value for Q indicates a good community structure

Modularity Maximization

- The intuition behind the idea of modularity is that a community is a structural element of a network that has been formed in a manner far from a random process.

- If we consider the actual density of links in a community, it should be significantly larger than the density we would expect if the links in the network were formed by a random process.

  - In other words, if two nodes i and j are end up being in the same community, there should be more likely a link between them (i.e., $A_{ij} = 1$, leading to an overall high value for Q).

  - If i and j end up being in a community such that the chances of having a link between them is just as the same as between any two nodes in the network (i.e., a random network), then the value of Q is more likely to be low (because there could be some $A_{ij} = 0$ that will bring down the value of Q).

Evaluating Modularity (Example 1)

Community [1, 4, 5, 7]

Edges with Aij = 1	Modularity
1 – 4	1 – (3)(4)/(2*15) = 0.60
4 – 5	1 – (4)(5)/(2*15) = 0.33
5 – 7	1 – (3)(5)/(2*15) = 0.50

Edges with Aij = 0	
1 – 5	0 – (3)(5)/(2*15) = -0.50
1 – 7	0 – (3)(3)/(2*15) = -0.30
4 – 7	0 – (4)(3)/(2*15) = -0.40

Total Modularity Score for Community [1, 4, 5, 7]   0.23

Total Modularity for the two Communities: 0.23 + 0.23 = 0.46
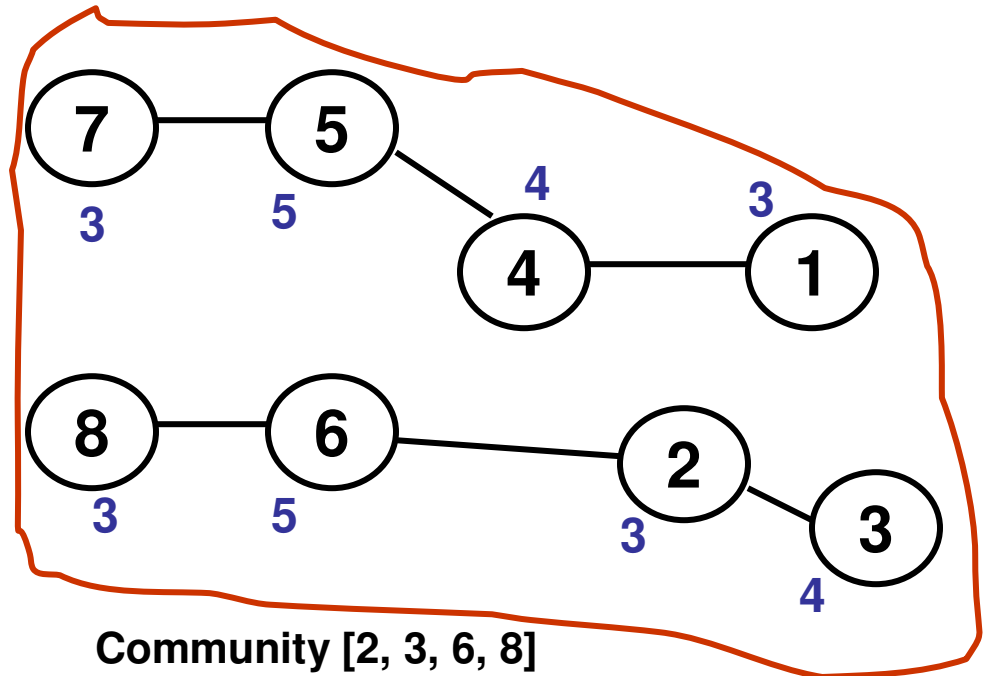
Community [2, 3, 6, 8]

Edges with Aij = 1	Modularity
2 – 3	1 – (3)(4)/(2*15) = 0.60
2 – 6	1 – (3)(5)/(2*15) = 0.50
6 – 8	1 – (3)(5)/(2*15) = 0.50

Edges with Aij = 0	
2 – 8	0 – (3)(3)/(2*15) = -0.30
3 – 6	0 – (4)(5)/(2*15) = -0.67
3 – 8	0 – (4)(3)/(2*15) = -0.40

Total Modularity Score for Community [2, 3, 6, 8]   0. 23

Evaluating Modularity (Example 2)
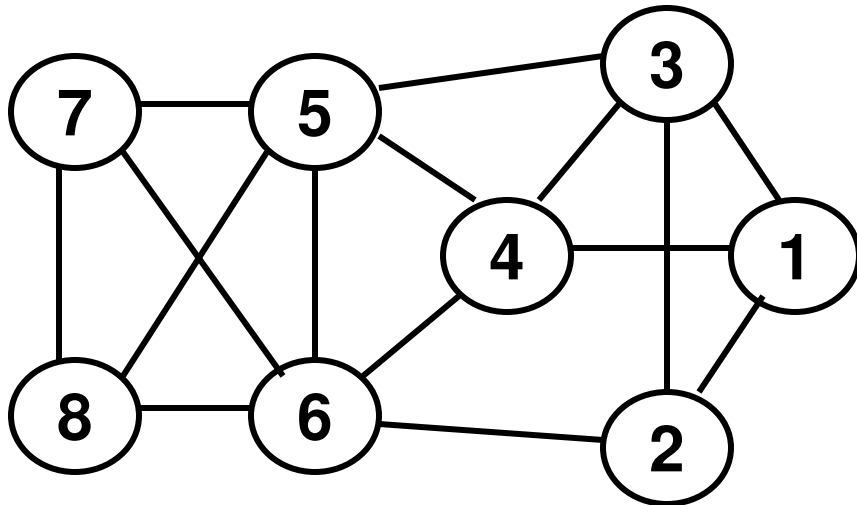
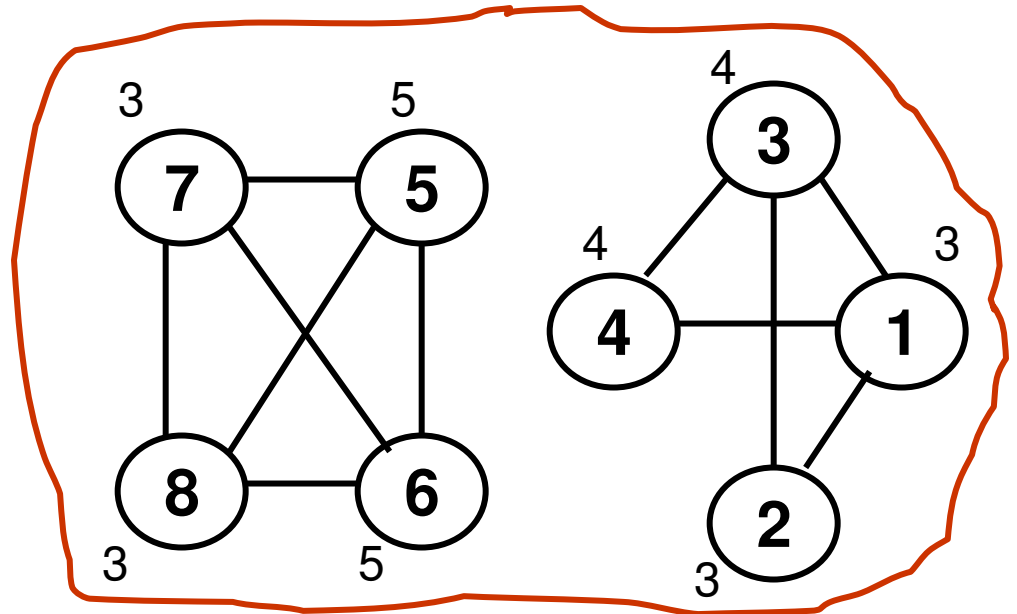

Community [1, 2, 3, 4]

Edges with Aij = 1	Modularity
1 – 2	1 – (3)(3)/(2*15) = 0.70
1 – 3	1 – (3)(4)/(2*15) = 0.60
1 – 4	1 – (3)(4)/(2*15) = 0.60
2 – 3	1 – (3)(3)/(2*15) = 0.70
3 – 4	1 – (4)(4)/(2*15) = 0.47

Edges with Aij = 0	
2 – 4	0 – (3)(4)/(2*15) = -0.40

Total Modularity Score for Community [1, 2, 3, 4]   2.67

Community [5, 6, 7, 8]

Edges with Aij = 1	Modularity
5 – 6	1 – (5)(5)/(2*15) = 0.17
5 – 7	1 – (3)(5)/(2*15) = 0.50
5 – 8	1 – (3)(5)/(2*15) = 0.50
6 – 7	1 – (3)(5)/(2*15) = 0.50
6 – 8	1 – (3)(5)/(2*15) = 0.50
7 – 8	1 – (3)(3)/(2*15) = 0.70

Total Modularity Score for Community [2, 3, 6, 8]   2.87

Total Modularity for the two Communities: 2.67 + 2.87 = 5.54

Using Eigenvectors to Identify Components

- Compute the Eigenvalues and Eigenvectors of the Adjacency matrix A
- The principal Eigenvector is the one that corresponds to the largest Eigenvalue.
- If all the entries in the "principal Eigenvector" are positive, then it implies that all the nodes are in one component.
  - Else, the vertices with the positive entries are in one component and those with the negative entries are in another component. (Note: 0 is considered positive).
- We apply the above interpretation to all the subsequent Eigenvectors (in the decreasing order of the corresponding Eigenvalues) and identify the smaller communities within the larger components.

Example 1: Eigenvectors to Identify Communities

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

1	2
1	3
2	3
3	4
4	5
4	6
5	6
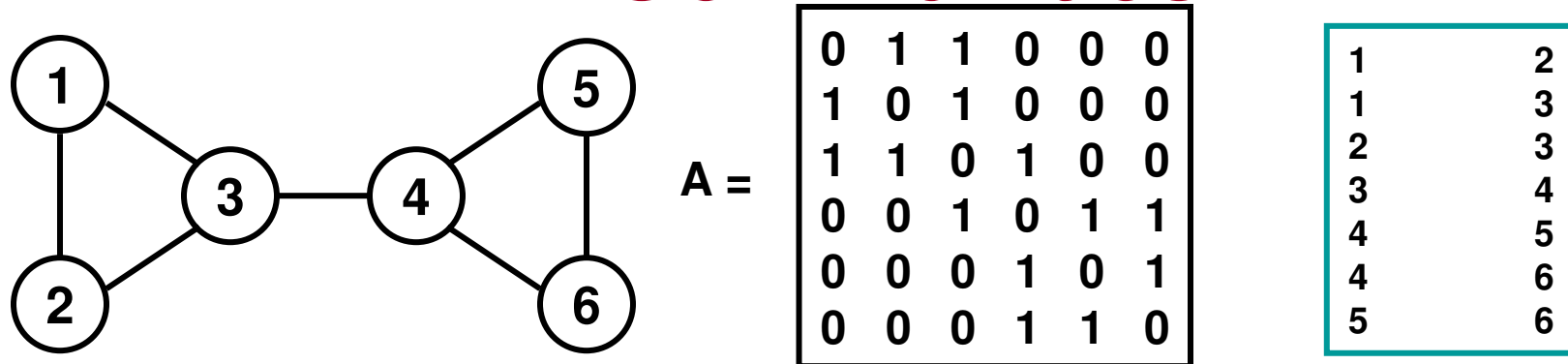
The Eigenvalues in the decreasing order and their corresponding Eigenvectors are:

Eigenvalue   Eigenvector

2.4142   [0.35; 0.35; 0.5; 0.5; 0.35; 0.35]

All entries are +ve; hence all vertices are in one single component community

1.7321   [-0.44; -0.44; -0.33; 0.33; 0.44; 0.44]

Vertices 1, 2, 3 form one community
Vertices 4, 5, 6 form another comm.

-0.4142   [0.35; 0.35; -0.5; -0.5; 0.35; 0.35]

Within 1-2-3; 3 is in one comm.
Within 4-5-6; 4 is in one comm.

-1   [-0.71; 0.71; 0; 0; 0; 0]
-1   [0; 0; 0; 0; -0.71; 0.71]
-1.7321   [0.23; 0.23; -0.63; 0.63; -0.23; -0.23]

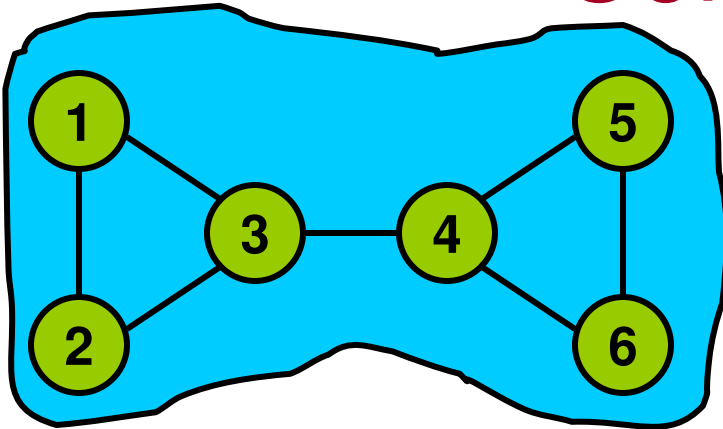# Example 1: Eigenvectors to Identify Communities (1)
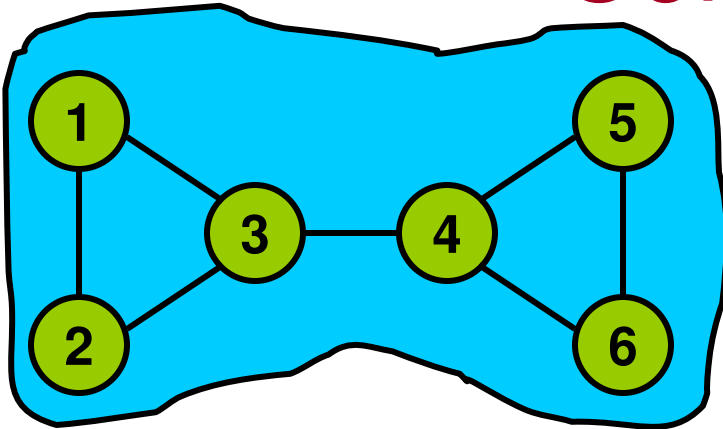
2.4142 [0.35; 0.35; 0.5; 0.5; 0.35; 0.35]

Edge	Modularity
3 – 4	$1 - (3*3/2*7) = 0.36$
3 – 5	$0 - (2*3/2*7) = -0.43$
3 – 6	$0 - (2*3/2*7) = -0.43$
4 – 5	$1 - (2*3/2*7) = 0.57$
4 – 6	$1 - (2*3/2*7) = 0.57$

Total Modularity = 0.47

Edge	Modularity
1 – 2	$1 - (2*2/2*7) = 0.71$
1 – 3	$1 - (2*3/2*7) = 0.57$
1 – 4	$0 - (2*3/2*7) = -0.43$
1 – 5	$0 - (2*2/2*7) = -0.29$
1 – 6	$0 - (2*2/2*7) = -0.29$
2 – 3	$1 - (2*3/2*7) = 0.57$
2 – 4	$0 - (2*3/2*7) = -0.43$
2 – 5	$0 - (2*2/2*7) = -0.29$
2 – 6	$0 - (2*2/2*7) = -0.29$

Example 1: Eigenvectors to Identify Communities (2)

2.4142   [0.35; 0.35; 0.5; 0.5; 0.35; 0.35]
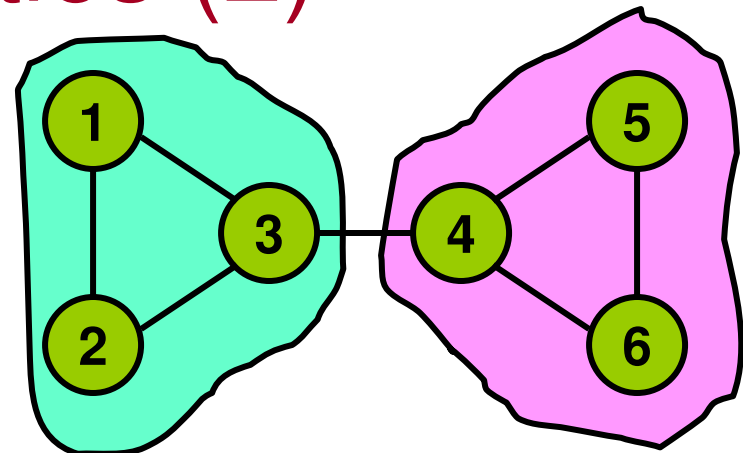
1.7321   [-0.44; -0.44; -0.33; 0.33; 0.44; 0.44]
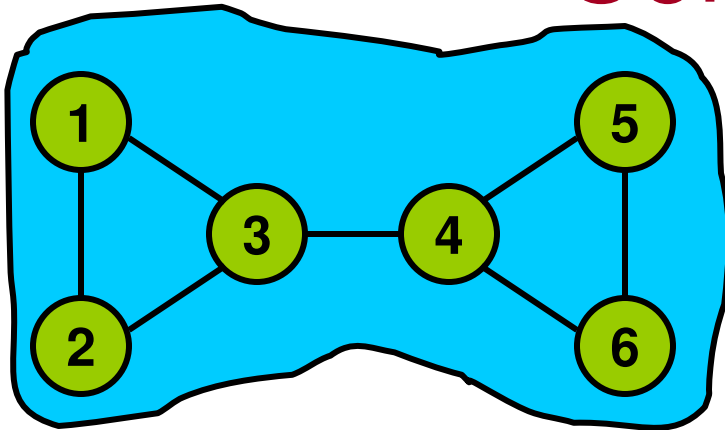
Modularity of [1, 2, 3]

Edge	Modularity
1 – 2	1 − {(2*2)/(2*7)} = 0.71
1 – 3	1 − {2*3)/(2*7)} = 0.57
2 – 3	1 − {(2*3)/(2*7)} = 0.57
Total Modularity = **1.85**	

Modularity of [4, 5, 6]

Edge	Modularity
4 – 5	1 − {(2*3)/(2*7)} = 0.57
4 – 6	1 − {2*3)/(2*7)} = 0.57
5 – 6	1 − {(2*2)/(2*7)} = 0.71
Total Modularity = **1.85**	

Total Modularity of [1, 2, 3] and [4, 5, 6] = <u>3.7</u>

Example 1: Eigenvectors to Identify Communities (3)
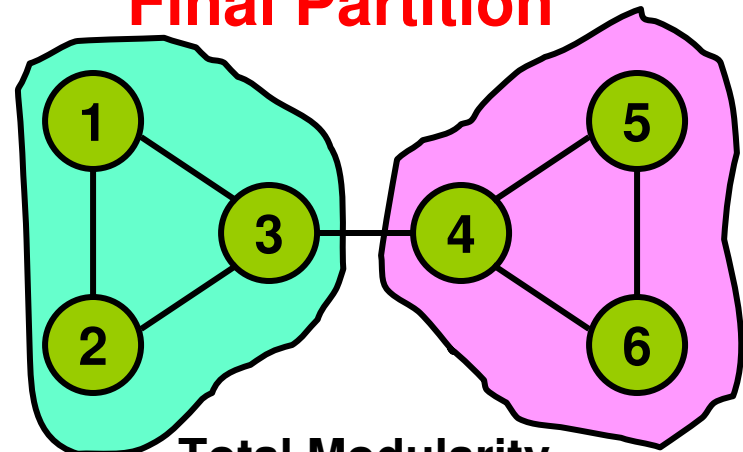

2.4142   [0.35; 0.35; 0.5; 0.5; 0.35; 0.35]
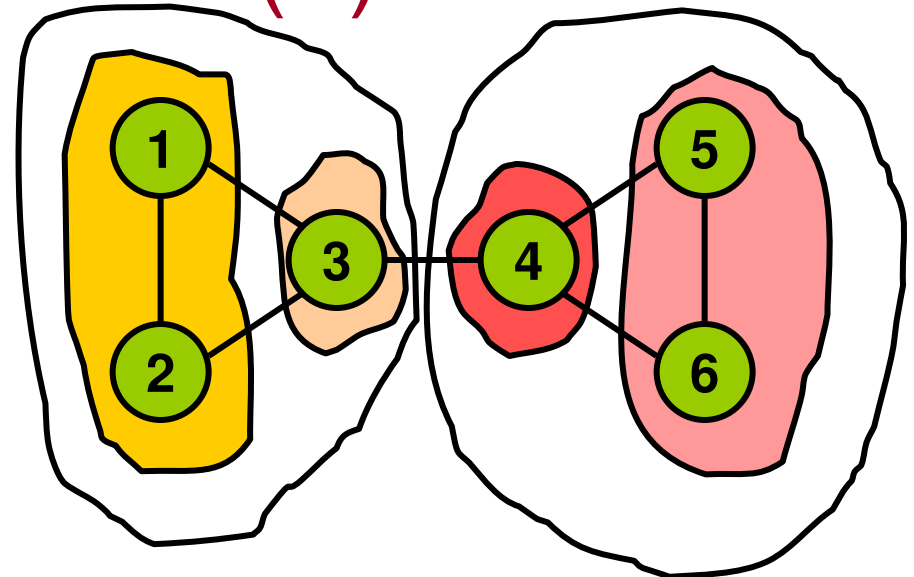
Modularity of [3] and [4] are 0 each
Modularity of [1, 2] and [5, 6] are 0.71 each

Total Modularity of [1, 2] and [3] is 0.71
is less than the modularity of [1, 2, 3]. Hence,
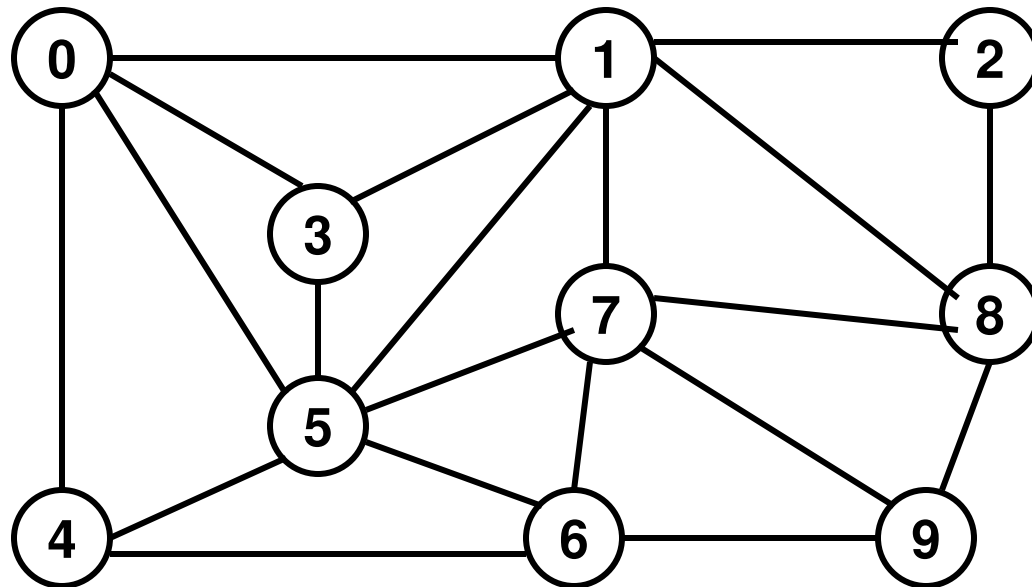We stay with [1, 2, 3] as a community.

Total Modularity of [5, 6[and [4] is 0.71
is less than the modularity of [4, 5, 6]. Hence,
We stay with [4, 5, 6] as a community.

Final Partition

Total Modularity Score = 3.7

Example 2

Eigenvalue # 10 (4.3515)

```
0  0.319710270928299
1  0.426420098753271
2  0.160703528303939
3  0.274067807962952
4  0.244256778523465
5  0.446482529766949
6  0.296966608945022
7  0.381728242984595
8  0.272885311344879
9  0.218615358909448
```
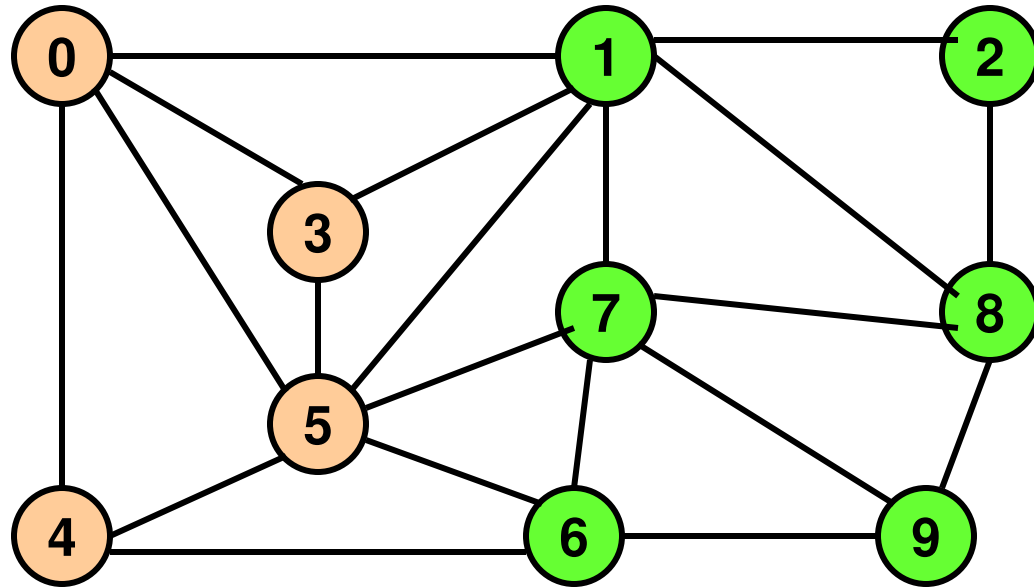
Eigenvalue # 9 (2.1175)

```
0  -0.414785230820025
1   0.003448307134804
2   0.221308082070367
3  -0.324654900415762
4  -0.280979155915067
5  -0.276112087029933
6   0.095930654776867
7   0.336379564509008
8   0.465166279849068
9   0.423842552587755
```

Eigenvalue # 8 (1.6723)

```
0   0.116484376791579
1   0.430514608467768
2   0.407558193770855
3   0.242122540021507
4  -0.335742360646069
5  -0.142098283222588
6  -0.535846699237412
7  -0.155162707963287
8   0.251043484439173
9  -0.263090938207366
```

Example 2 (1)

Eigenvalue # 9 (2.1175)

```
0 -0.41478523082002594
1 0.0034483071344804475
2 0.22130808207036753
3 -0.32465490044157624
4 -0.28097915591550676
5 -0.27611208702993
6 0.09593065477768678
7 0.33637956450900897
8 0.465166279849068
9 0.42384255258775577
```
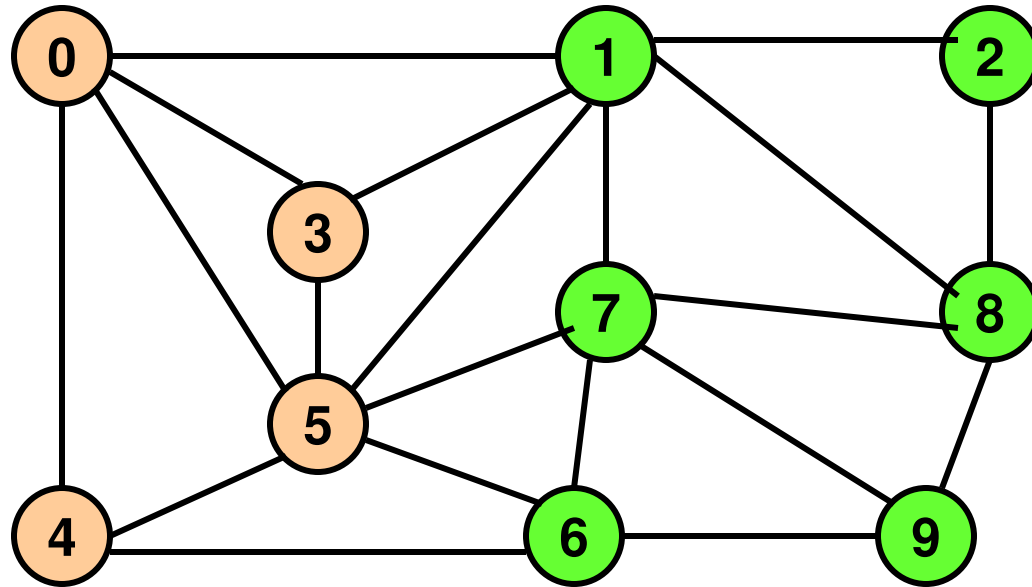
Modularity for Community [0, 3, 4, 5]

Edge	Modularity
0 – 3	$1 - \{(4*3)/(2*20)\} = 0.7$
0 – 4	$1 - \{(4*3)/(2*20)\} = 0.7$
0 – 5	$1 - \{(4*6)/(2*20)\} = 0.4$
3 – 4	$0 - \{(3*3)/(2*20)\} = -0.23$
3 – 5	$1 - \{(3*6)/(2*20)\} = 0.55$
4 – 5	$1 - \{(3*6)/(2*20)\} = 0.55$

Total Modularity Score for
[0, 3, 4, 5] = **2.67**

Modularity for Community [1, 2, 6, 7, 8, 9]

| Edge	Modularity
1 – 2	$1 - \{(6*2)/(2*20)\} = 0.7$
1 – 6	$0 - \{(6*4)/(2*20)\} = -0.6$
1 – 7	$1 - \{(6*5)/(2*20)\} = 0.25$
1 – 8	$1 - \{(6*4)/(2*20)\} = 0.4$
1 – 9	$0 - \{(6*3)/(2*20)\} = -0.45$
2 – 6	$0 - \{(2*4)/(2*20)\} = -0.2$
2 – 7	$0 - \{(2*5)/(2*20)\} = -0.25$
2 – 8	$1 - \{(2*4)/(2*20)\} = 0.8$

Example 2 (2)

Modularity for Community [1, 2, 6, 7, 8, 9]

Edge	Modularity
1 − 2	1 − {(6*2)/(2*20)} = 0.7
1 − 6	0 − {(6*4)/(2*20)} = -0.6
1 − 7	1 − {(6*5)/(2*20)} = 0.25
1 − 8	1 − {(6*4)/(2*20)} = 0.4
1 − 9	0 − {(6*3)/(2*20)} = -0.45
2 − 6	0 − {(2*4)/(2*20)} = -0.2
2 − 7	0 − {(2*5)/(2*20)} = -0.25
2 − 8	1 − {(2*4)/(2*20)} = 0.8
2 − 9	0 − {(2*3)/(2*20)} = -0.15
6 − 7	1 − {(4*5)/(2*20)} = 0.5
6 − 8	0 − {(4*4)/(2*20)} = -0.4
6 − 9	1 − {(4*3)/(2*20)} = 0.7
7 − 8	1 − {(5*4)/(2*20)} = 0.5
7 − 9	1 − {(5*3)/(2*20)} = 0.63
8 − 9	1 − {(4*3)/(2*20)} = 0.7

Total Modularity Score for [1, 2, 6, 7, 8, 9] = **3.13**

Example 2 (3)
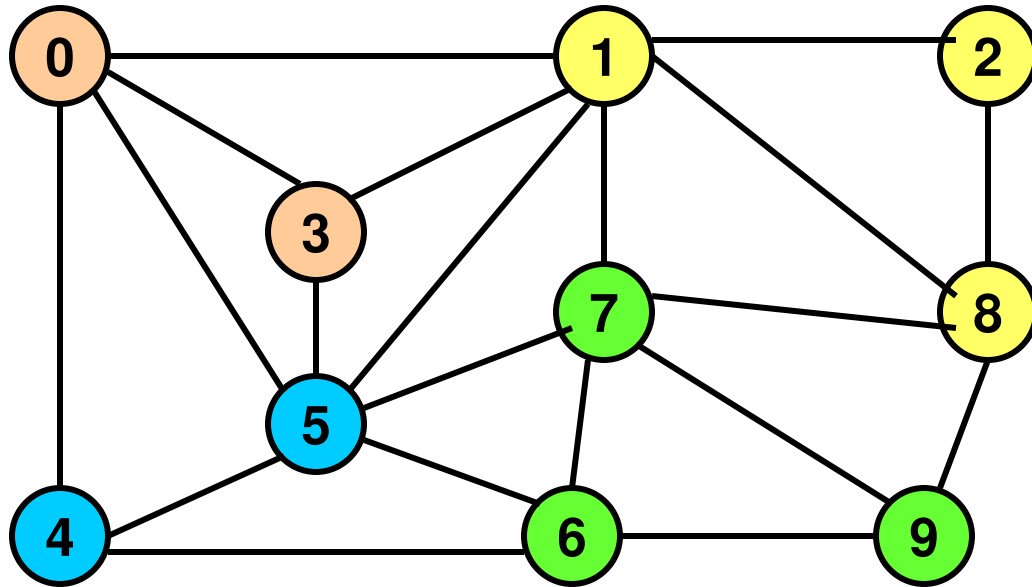
Eigenvalue # 8 (1.6723)

0 0.1164843767915 7984
1 0.4305144608467 7686
2 0.4075581993770 8554
3 0.2421225400215074
4 -0.3357423606460691
5 -0.1420982832225889
6 -0.5358466992374123
7 -0.1551627079632 8716
8 0.2510434844391 7357
9 -0.26309093820736645

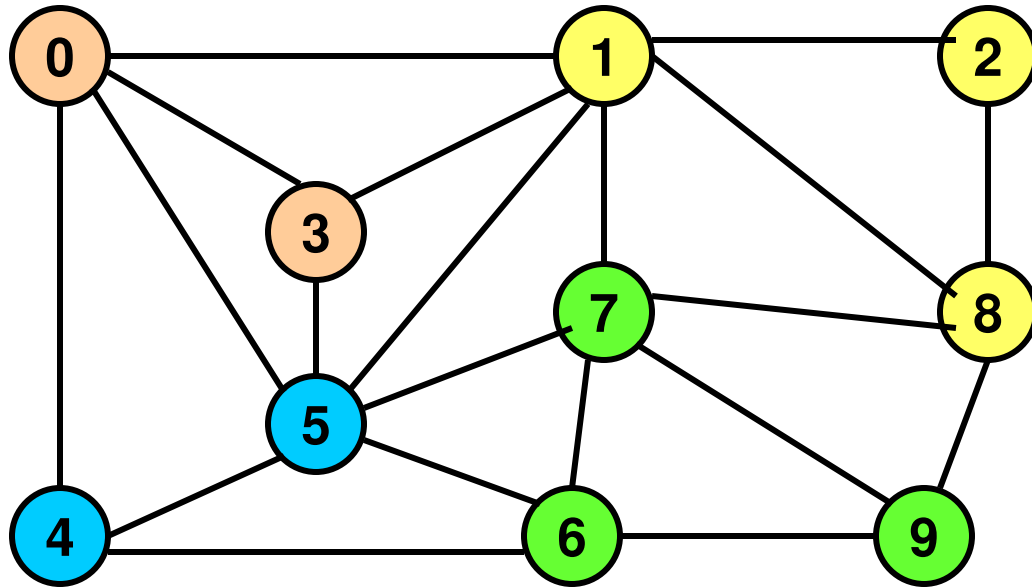**Modularity of Community [0, 3] = 0.7**
Modularity of Community [4, 5] = 0.55
Total Modularity of [0, 3] and [4, 5] = 1.25
is less than the Modularity of [0, 3, 4, 5]
= 2.67
Hence, we stay with community [0, 3, 4, 5]
without further partitioning it.

Example 2 (4)

Modularity of Community [1, 2, 8]
Edge   Modularity
1 − 2   1 − {(6*2)/(2*20)} = 0.7
1 − 8   1 − {(6*4)/(2*20)} = 0.4
2 − 8   1 − {(2*4)/(2*20)} = 0.8
Total Modularity of [1, 2, 8] = **1.9**

Modularity of Community [6, 7, 9]
Edge   Modularity
6 − 7   1 − {(4*5)/(2*20)} = 0.5
6 − 9   1 − {(4*3)/(2*20)} = 0.7
7 − 9   1 − {(5*3)/(2*20)} = 0.63
Total Modularity of [6, 7, 9] = **1.83**

Modularity of [1, 2, 8] = 1.9
Modularity of [6, 7, 9] = 1.83
Total Modularity of [1, 2, 8] and [6, 7, 9]
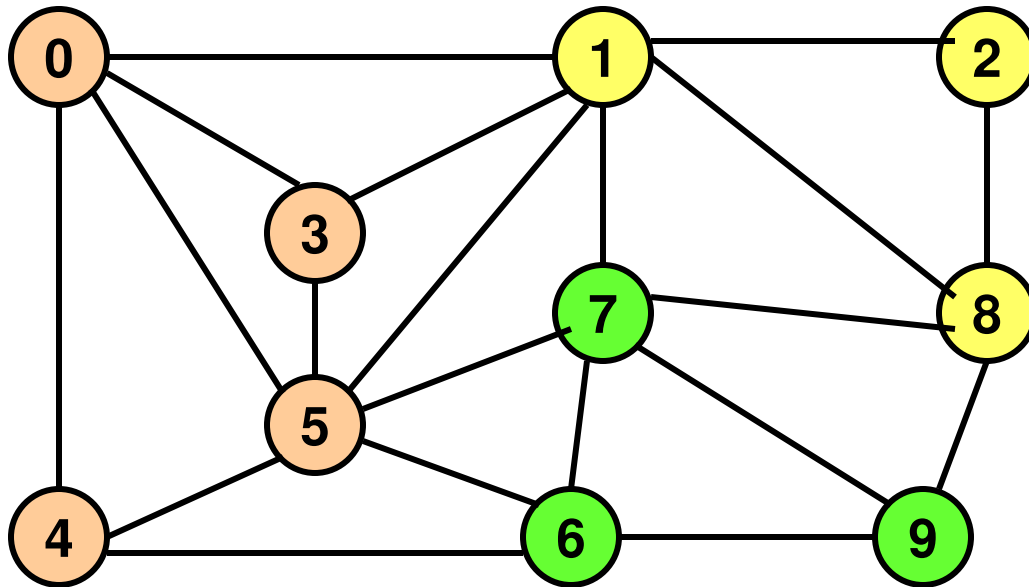= 3.73
is larger than the modularity of
[1, 2, 6, 7, 8, 9] = 3.13
Hence, we allow the partitioning of
[1, 2, 6, 7, 8, 9] into [1, 2, 8] and [6, 7, 9]

Example 2 (5)

Final Partition

Modularity of [0, 3, 4, 5] = 2.67
Modularity of [1, 2, 8] = 1.9
Modularity of [6, 7, 9] = 1.83
Total Modularity = 6.4