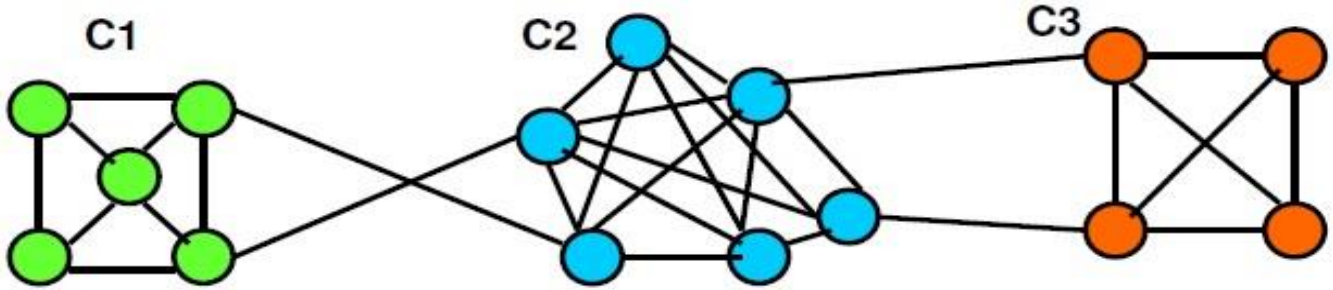
# Community

- **Community:** It is formed by individuals such that those within a group interact with each other more frequently than with those outside the group.
- **Community detection:** discovering groups in a network where individuals' group memberships are not explicitly given.
- Interactions (edges) between nodes can help determine communities
- **Community structures are quite common in real networks**. Social networks include community groups based **on common location, interests, occupation**, etc.
- **Metabolic networks have communities based on functional groupings**.
- Citation networks form communities by research topic.

# Internal and External Community Densities




- Let C be a subset of nodes (V) that form a community.
- For every node i in C, let $k_i^{int}$ and $k_i^{ext}$ be the # links connecting node i to a node in C and outside C respectively.

$$\delta_{\text{int}}(C) = \frac{\sum_i k_i^{\text{int}}}{n_C(n_C - 1)}$$

$$\delta_{ext}(C) = \frac{\sum_i k_i^{ext}}{2n_C(n_C - 1)}$$

The internal density of every cluster is significantly larger than the external density as well as the total density of the network.

**Internal Densities**

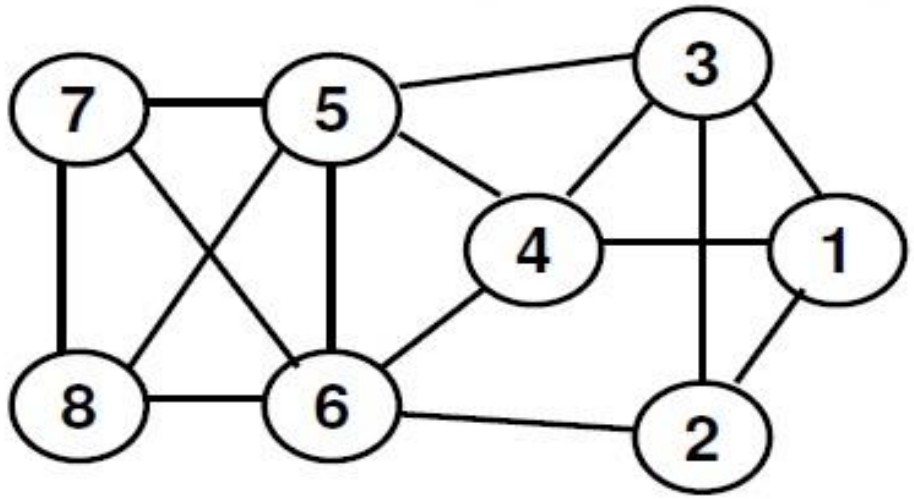
| | |
|---|---|
| C1 | (4*3 + 1*4) / (5*4) = 0.8 |
| C2 | (6*5)/(6*5) = 1.0 |
| C3 | (4*3)/(4*3) = 1.0 |

**External Densities**

| | |
|---|---|
| C1 | (1 + 1) / (2*5*4) = 0.05 |
| C2 | (1 + 1 + 1 + 1)/(2*6*5) = 0.067 |
| C3 | (1 + 1)/(2*4*3) = 0.083 |

# Modularity Maximization

- Modularity measures the strength of a community partition by taking into account the degree distribution.
- Given a network with *m* edges, the expected number of edges between two nodes *i* and *j* with degrees $d_i$ and $d_j$ respectively is $d_i * d_j$ / 2m.




Expected number of edges between nodes 1 and 2 is (3)(2) / (2*15) = 0.20

**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:**

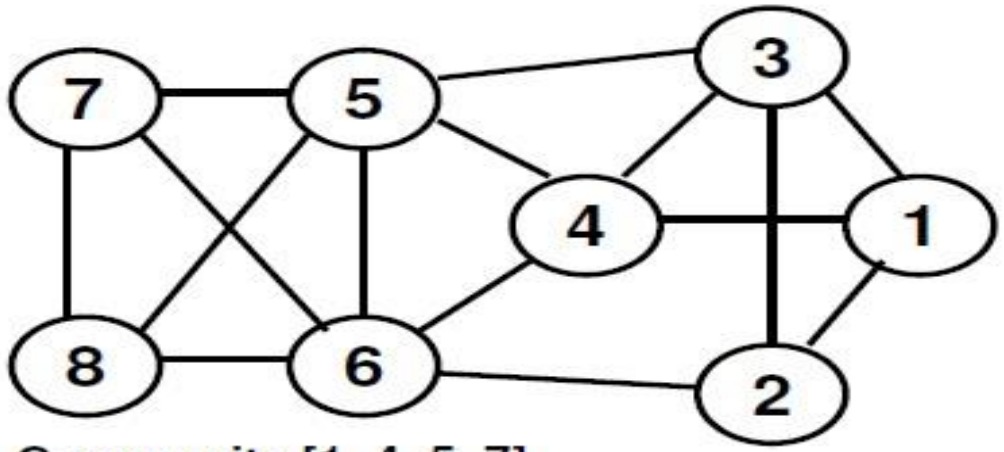
$$Q = \sum_{l=1}^{k} \sum_{i \in C_l,\, j \in C_l} A_{i,j} - \frac{d_i d_j}{2m}$$

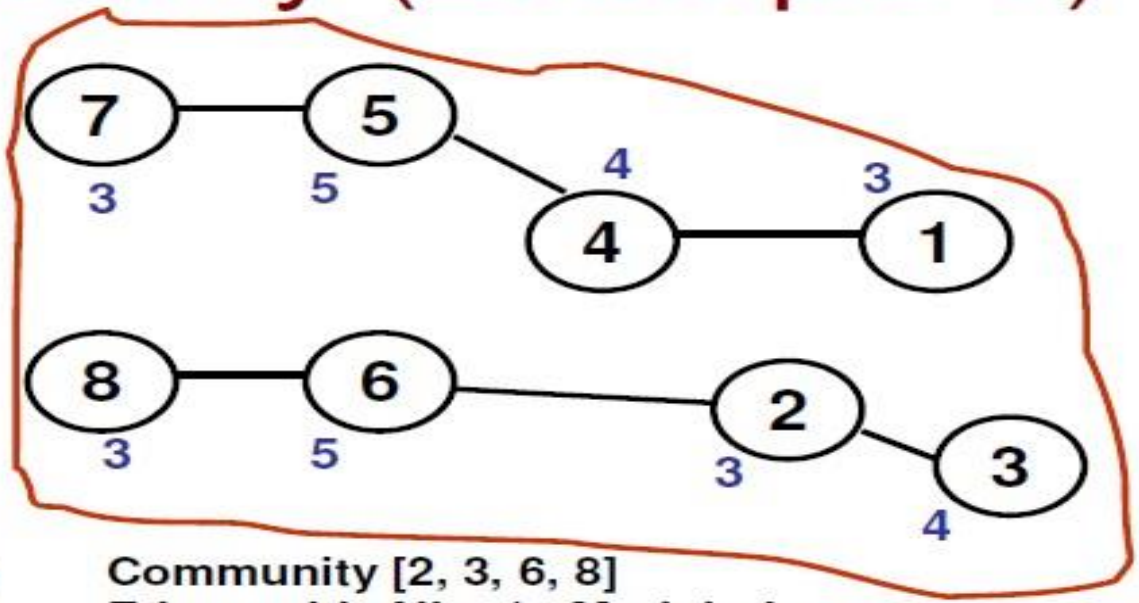
**A larger value for Q indicates a good community structure**

# 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., Aij = 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 Aij = 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

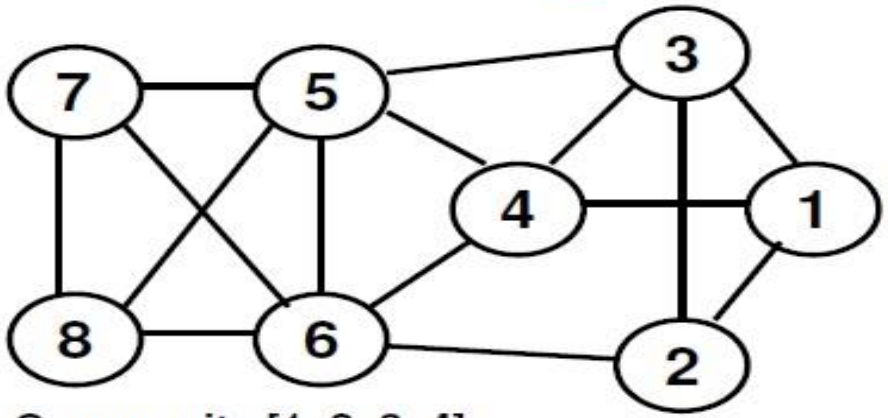
**Total Modularity for the two Communities: 0.23 + 0.23 = 0.46**

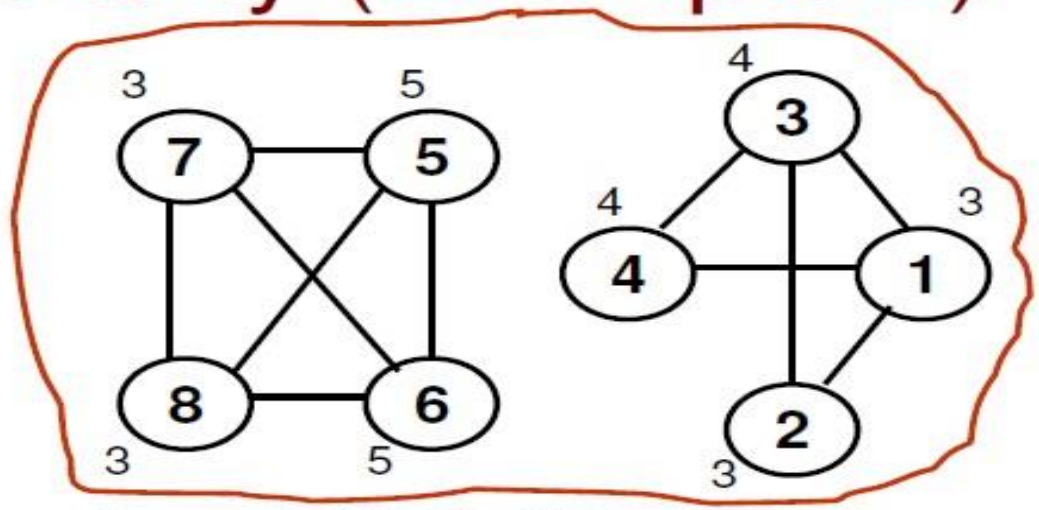
**Community [2, 3, 6, 8]**

| 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

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

**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

# Girvan-Newman Algorithm for Community Detection

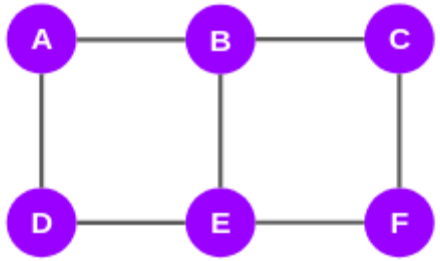
- Under the Girvan-Newman algorithm, **the communities in a graph are discovered by iteratively removing the edges of the graph, based on the edge betweenness centrality value.**
- The edge with the highest edge betweenness is removed first. first, let's understand the concept of "edge betweenness centrality".

**Edge Betweenness Centrality (EBC)**

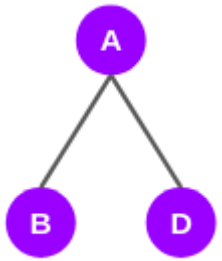
- The **edge betweenness centrality (EBC) can be defined as the number of shortest paths that pass through an edge in a network. Each and every edge is given an EBC score based on the shortest paths among all the nodes in the graph**.

# Girvan-Newman Algorithm for Community Detection

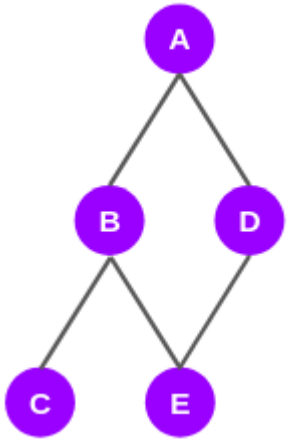
With respect to graphs and networks, the shortest path means the path between any two nodes covering the least amount of distance. Let’s take an example to find how EBC scores are calculated. Consider this graph below:



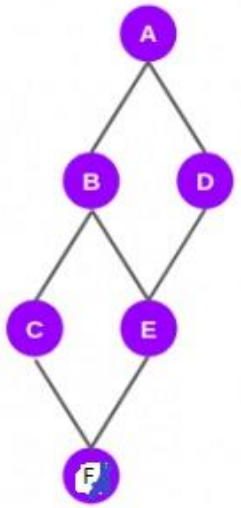

Now, let’s start with node A. The directly connected nodes to node A are nodes B and D. So, the shortest paths to B and D from A are AB and AD respectively:



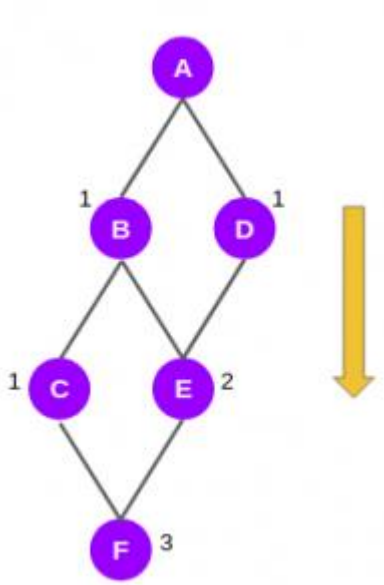

It turns out that the shortest paths to nodes C and E from A go through B and D:




The shortest paths to the last node F from node A, pass through nodes B, D, C, and E:



## Assigning Scores to Nodes



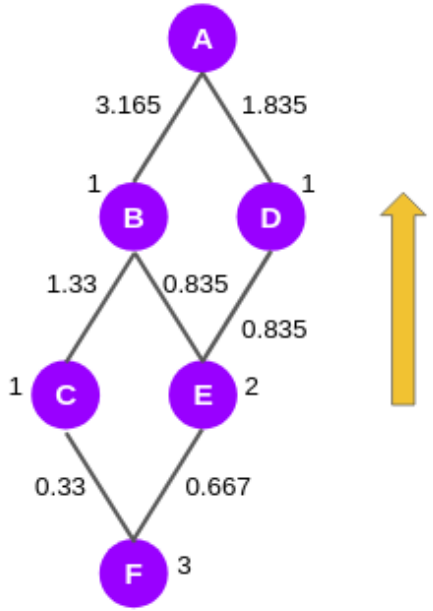

As you can see in the graph above, nodes B and D have been given a score of 1 each. This is because the shortest path to either node from node A is only one. For the very same reason, node C has been given a score of 1 as there is only one shortest path from node A to node C.

Moving on to node E. It is connected to node A through two shortest paths, ABE and ADE. Hence, it gets a score of 2.

The last node F is connected to A through three shortest paths — ABCF, ABEF, and ADEF. So, it gets a score of 3.

## Computing Scores for Edges

Next, we will proceed with computing scores for the edges. Here we will move in the backward direction, from node F to node A:




FC = 1/3 = 0.33
FE = 2/3 = 0.667

CB = 1 + 0.33 = 1.33
EB = (1 + 0.667)/2 = 0.835
ED = (1 + 0.667)/2 = 0.835

BA = (1 + 1.33 + 0.835)/1 = 3.165
DA = (1 + 0.835)/1 = 1.835

We first compute the score for the edges FC and FE. As you can see, the edge score for edge FC is the ratio of the node scores of C and F, i.e. 1/3 or 0.33. Similarly, for FE the edge score is 2/3.

Now we have to calculate the edge score for the edges CB, EB, and ED. According to the Girvan-Newman algorithm, from this level onwards, every node will have a default value of 1 and the edge scores computed in the previous step will be added to this value.

So, the edge score of CB is (1 + 0.33)/1. Similarly, edge score EB or ED is (1 + 0.667)/2. Then we move to the next level to calculate the edge scores for BA and DA.

So far, we have computed the edge scores of the shortest paths with respect to node A. We will have to repeat the same steps again from the other remaining five nodes.