INTRODUCTION TO DATA ANALYTICS

Social Network Analysis

(SNA)

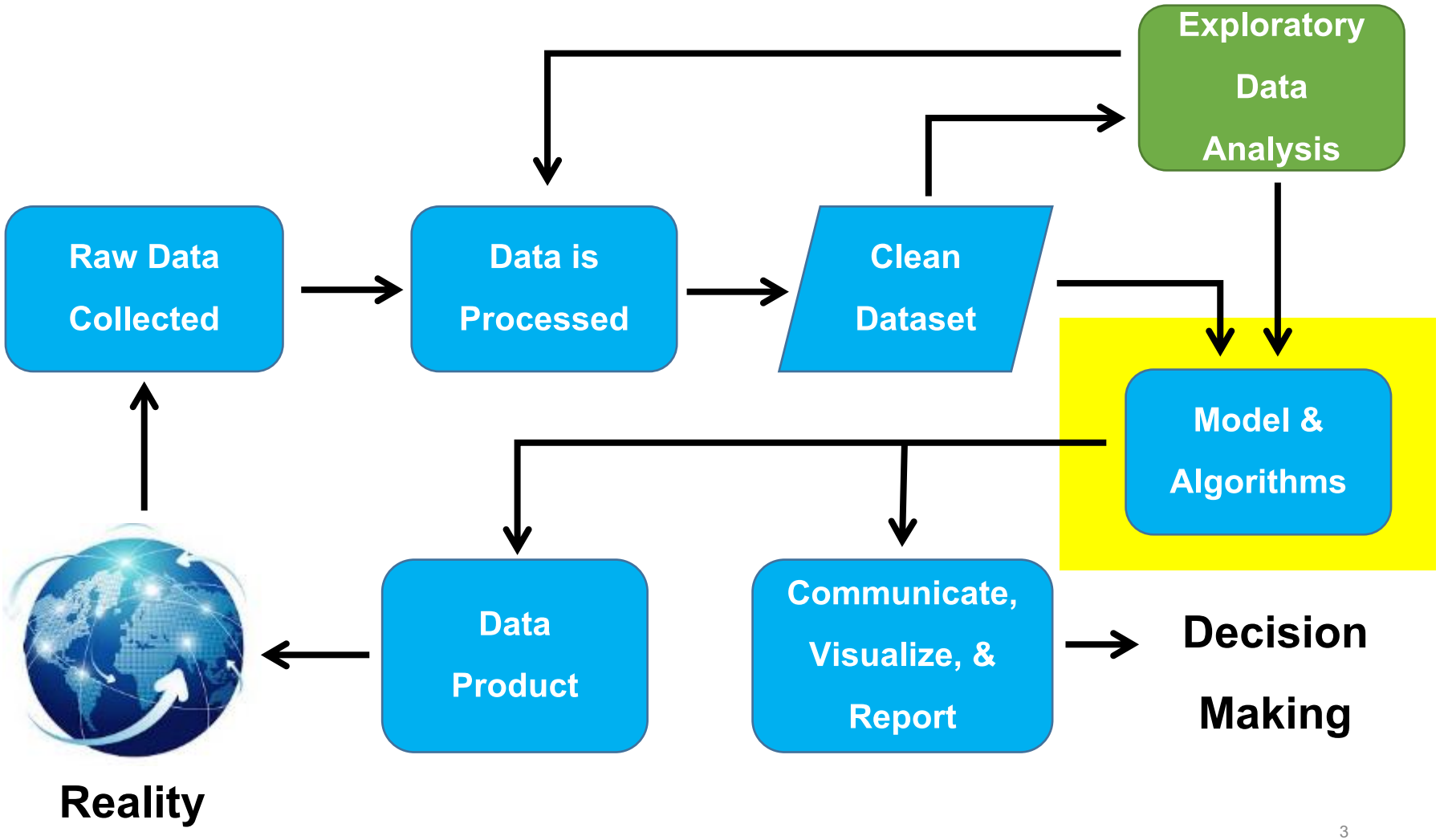
Dr. Rathachai Chawuthai

Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Agenda

- Introduction
- SNA Properties
- Community Detection

Data Science Process



Machine Learning

Supervised Learning

Develop predictive model based on both input and output data

Unsupervised Learning

Develop predictive model based on both input and output data

Regression

- Linear Regression
- Polynomial Regression

Classification

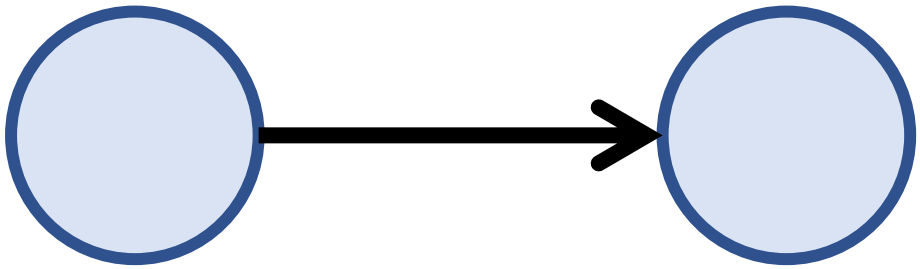
- Decision Tree
- Logistic Regression
- Neural Network
- etc.

Clustering

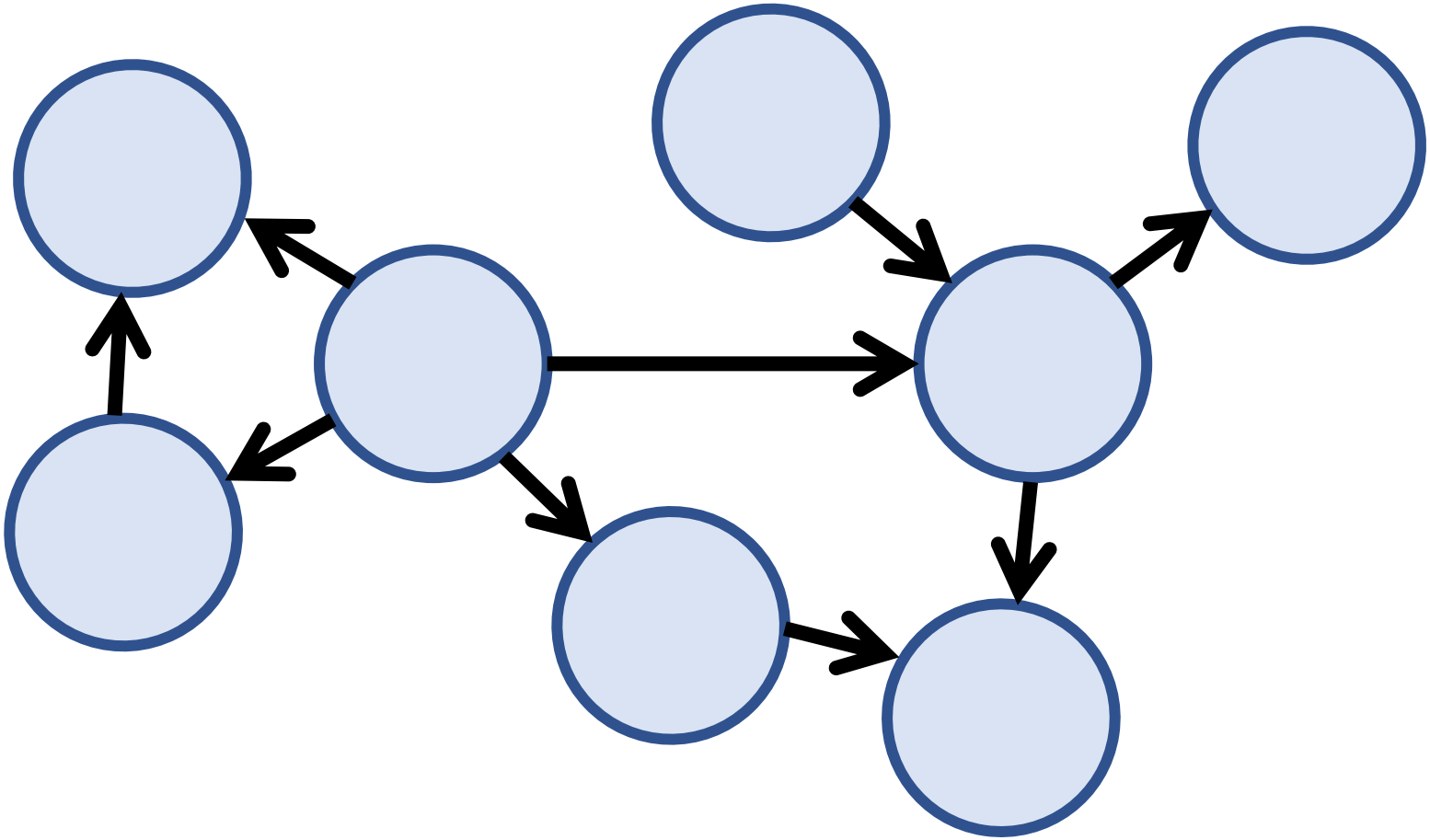
- K-Means
- DB-SCAN
- etc.

Introduction

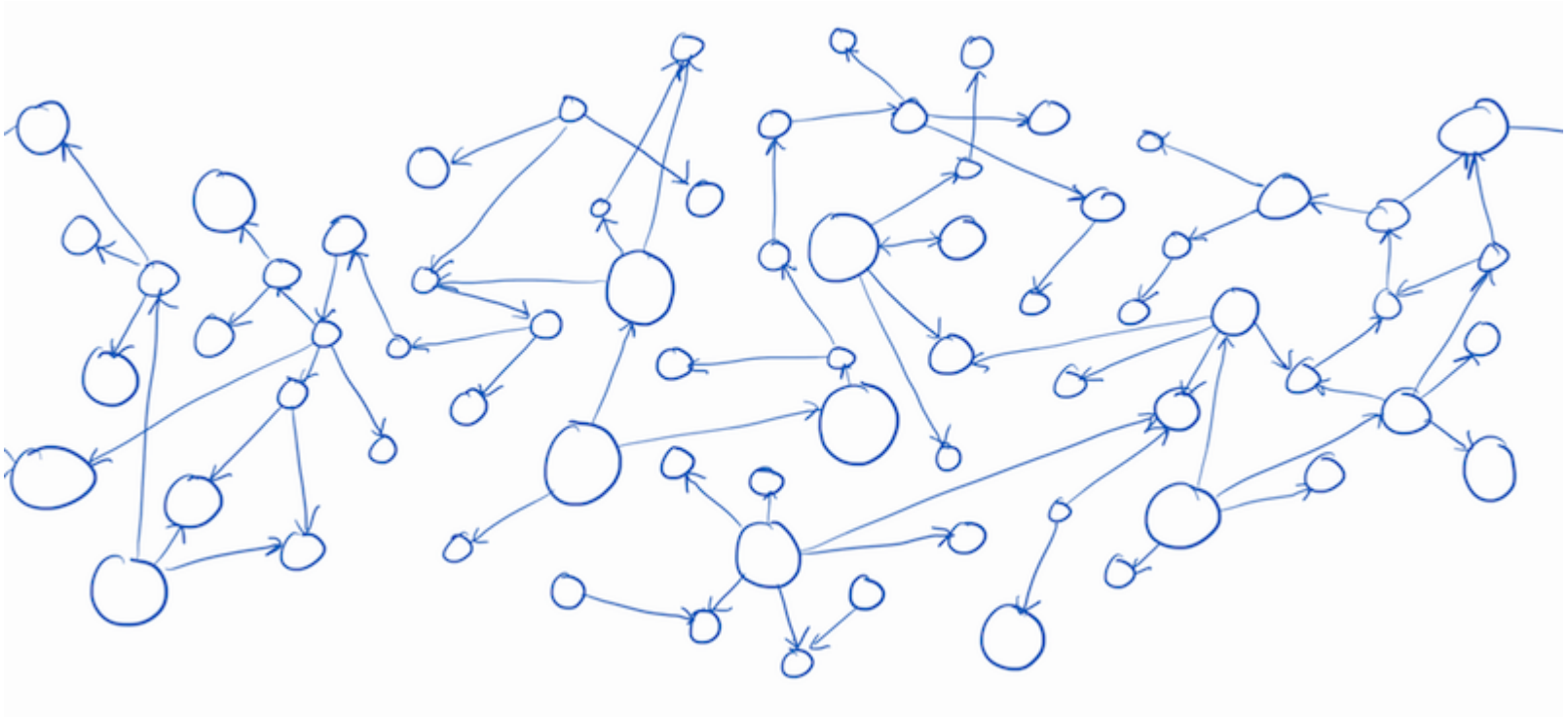
Graph



Graph

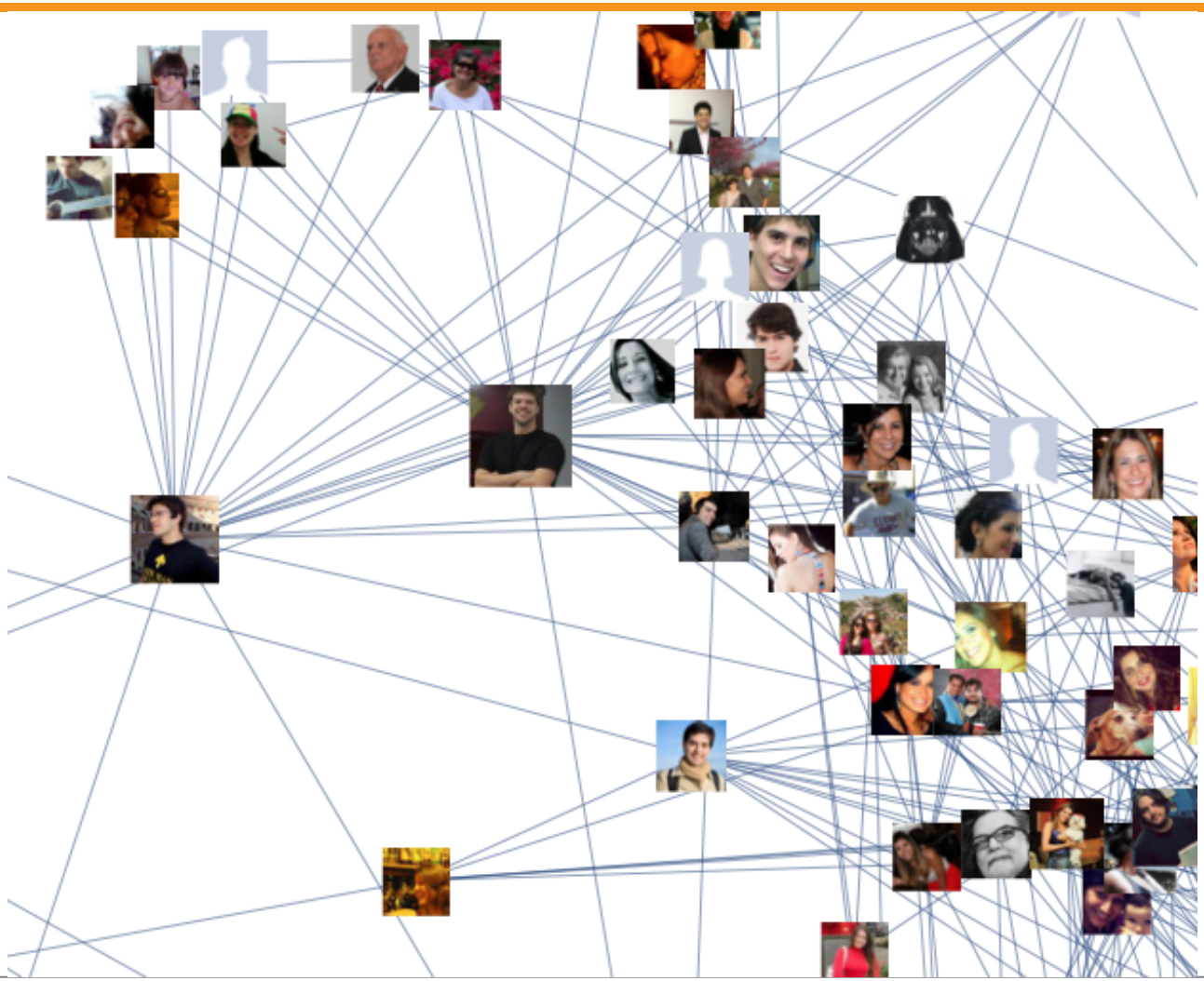


Graph



Ref: • (image) https://code-cartoons.com/a-cartoon-intro-to-facebook-s-relay-part-1-3ec1a127bca5

Graph

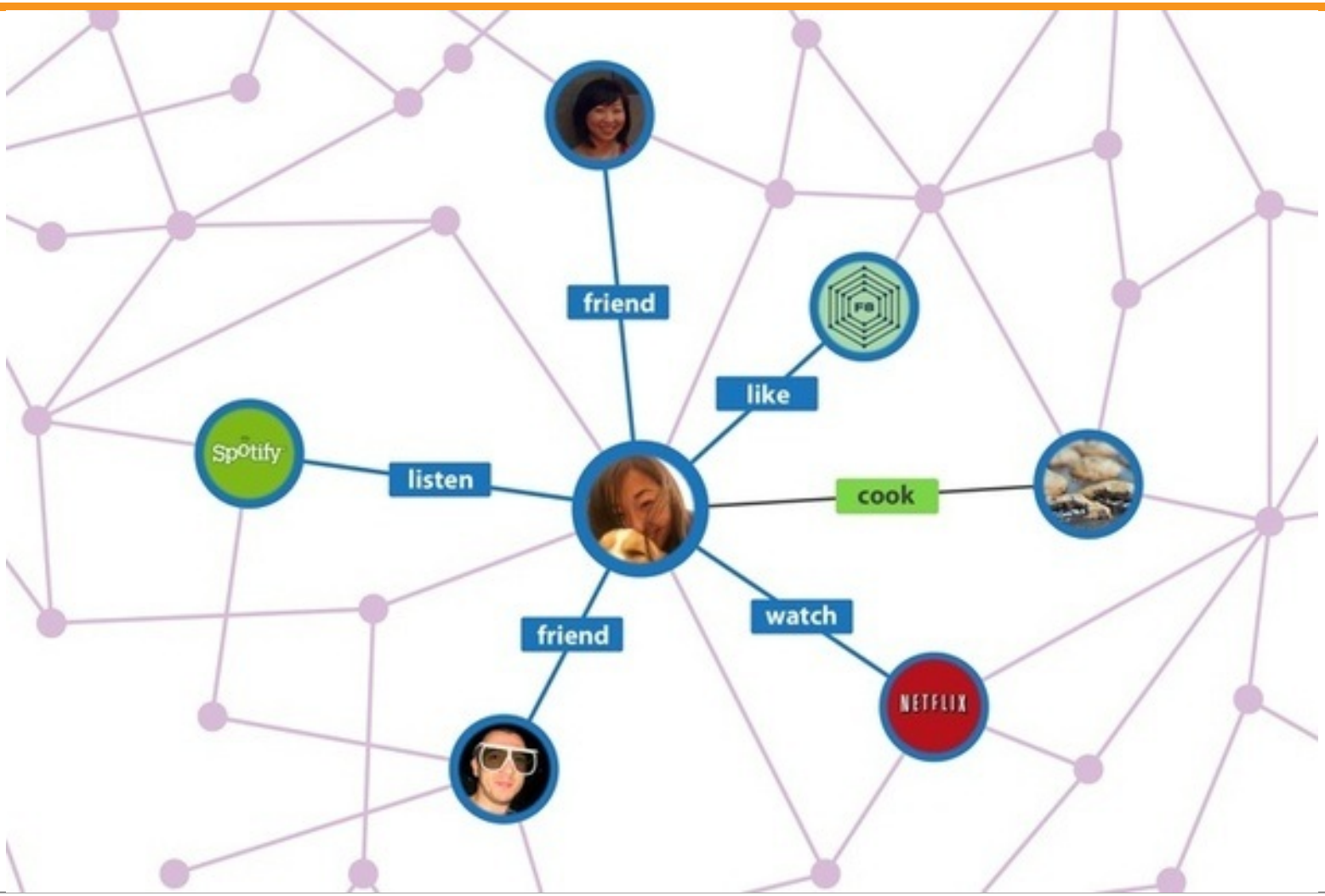


Ref: • (image) https://mathematica.stackexchange.com/questions/11673/how-to-play-with-facebook-data-inside-mathematica

Graph

Ref: • (image) https://mathematica.stackexchange.com/questions/11673/how-to-play-with-facebook-data-inside-mathematica

Graph



Ref: • (image) https://www.quora.com/How-does-graph-theory-apply-to-Facebook

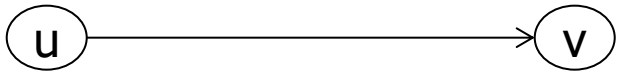
Definitions - Graph

A generalization of the simple concept of a set of dots, links, edges or arcs.

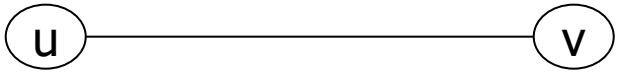
Representation: Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)

Edge Type

Directed: Ordered pair of vertices. Represented as (u, v) directed from vertex u to v.



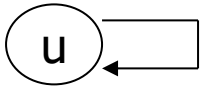
Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably.



Ref: • B. Prabhakaran: "Graph Theory"

Edge Type

- **Loop:** A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}

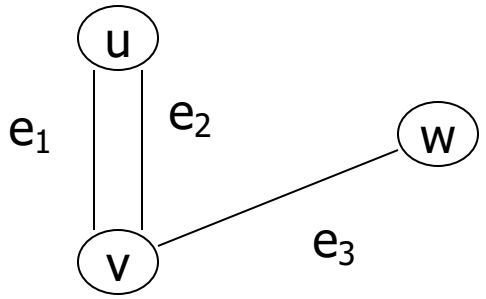


- **Multiple Edges:** Two or more edges joining the same pair of vertices.

Graph Type

Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V, u ≠ v}. The edges e1 and e2 are called multiple or parallel edges if f (e1) = f (e2).

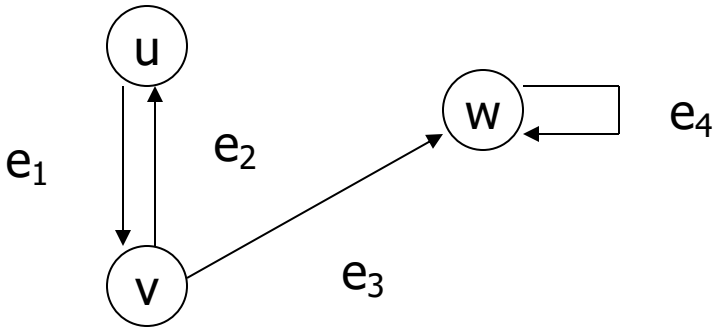
Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3}



Graph Type

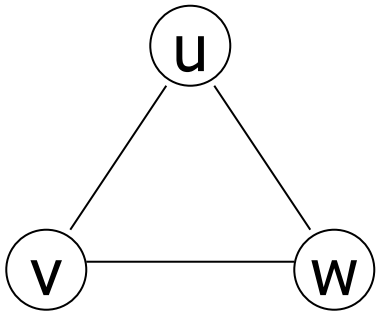
Directed Multigraph: G(V,E), consists of set of vertices V, set of Edges E and a function f from E to {{u, v}| u, v V}. The edges e1 and e2 are multiple edges if f(e1) = f(e2)

Representation Example: V = {u, v, w}, E = {e_1, e_2, e_3, e_4}



Representation- Adjacency Matrix

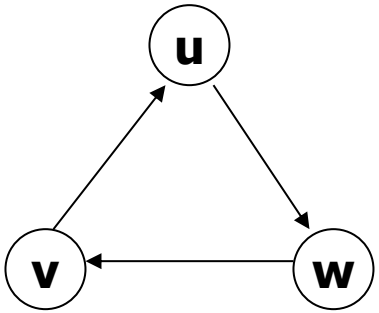
- Example: Undirected Graph G (V, E)



	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Representation- Adjacency Matrix

- Example: Directed Graph G (V, E)

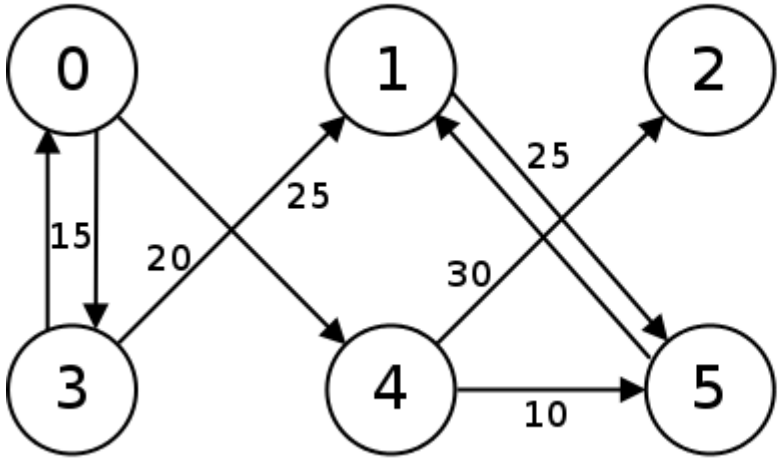


	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Representation- Adjacency Matrix

- Example: Weighted Graph G (V, E, w)

แบบมีน้ำหนักก็คล้ายๆกัน



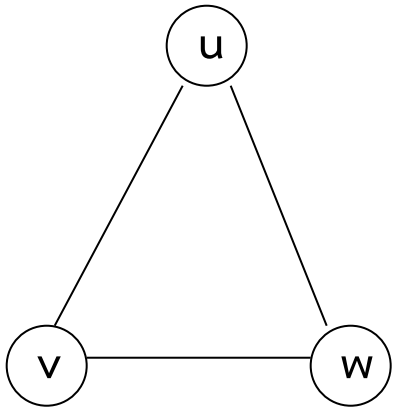
	0	1	2	3	4	5
0				15	20	
1						25
2						
3	15	25				
4			30			10
5		25				

Ref: • http://cs-si-ucr.github.io/CS14-Spring15/lab_week_09/

Representation- Adjacency List

- Each node (vertex) has a list of which nodes (vertex) it is adjacent

  Example: undirected graph G (V, E)



node	Adjacency List
u	v , w
v	w, u
w	u , v

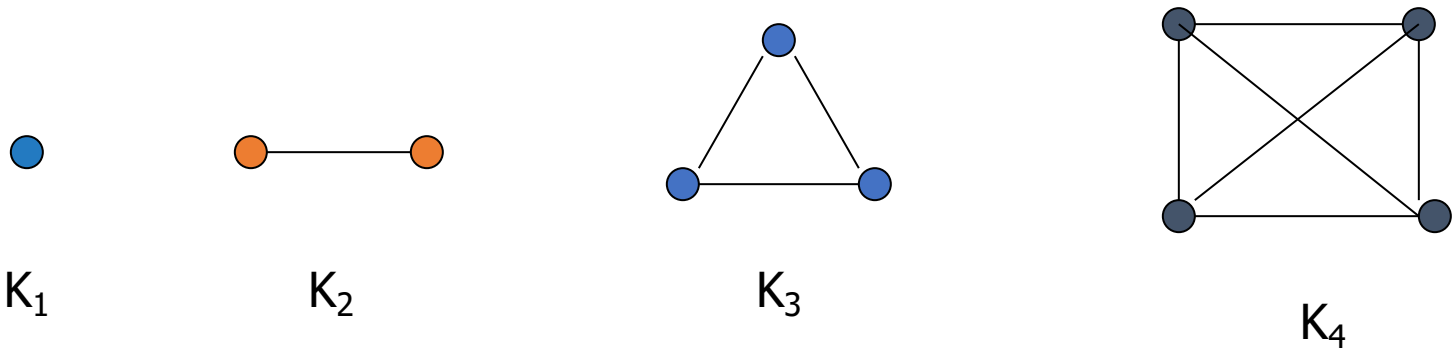
ลักษณะของกราฟ

Complete Graph

ทุกๆ node เชื่อมกันหมด

- **Complete graph:** K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

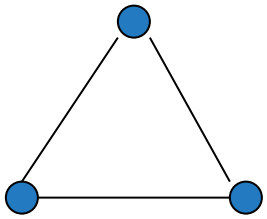
Representation Example: K_1, K_2, K_3, K_4



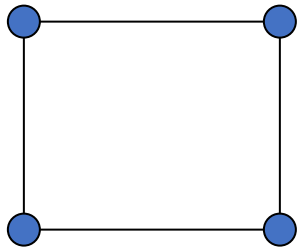
Cycle

- **Cycle:** C_n, $n \geq 3$ consists of n vertices v_1, v_2, v_3 … v_n and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$ … $\{v_{n-1}, v_n\}$, $\{v_n, v_1\}$

  Representation Example: C_3, C_4



C_3



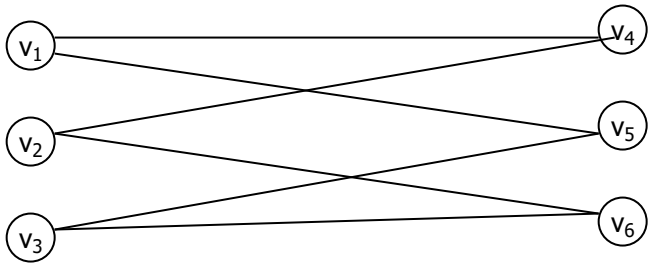
C_4

Bipartite Graphs

- In a simple graph G, if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2)

  Application example: Representing Relations

  Representation example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



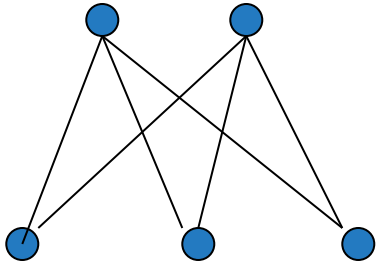
Ref: • B. Prabhakaran: "Graph Theory"

Complete Bipartite Graph

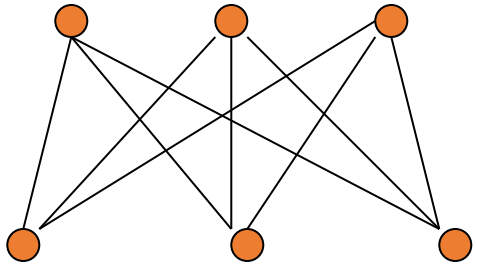
เชื่อมทุกจุดแบบทั้งหมด

- $K_{m,n}$ is the graph that has its vertex set portioned into two subsets of m and n vertices, respectively There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

  Representation example: $K_{2,3}$, $K_{3,3}$



$K_{2,3}$



$K_{3,3}$

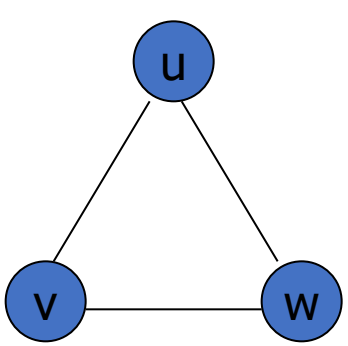
Ref: • B. Prabhakaran: "Graph Theory"

Subgraphs เป็นยังไงบ้างเอ่ย...

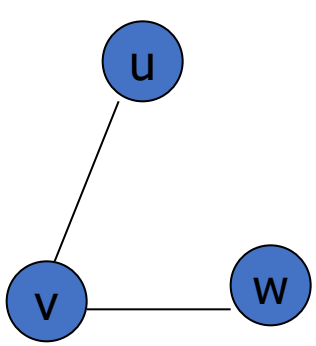
- A subgraph of a graph G = (V, E) is a graph H =(V', E') where V' is a subset of V and E' is a subset of E

  Application example: solving sub-problems within a graph

  Representation example: V = {u, v, w}, E = ({u, v}, {v, w}, {w, u}}, H_1 , H_2



G



H_1

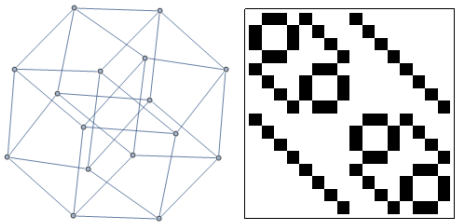
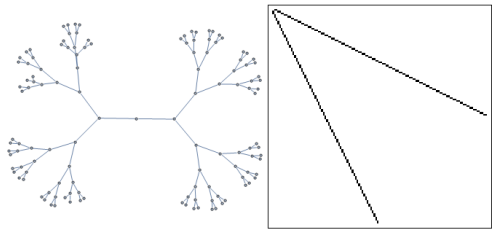
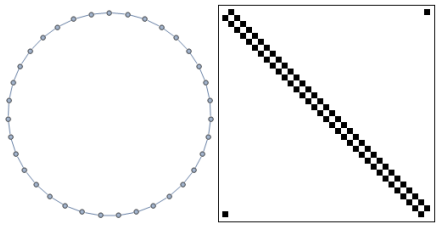
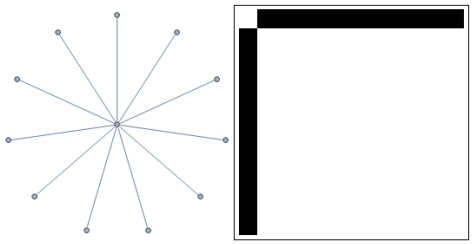
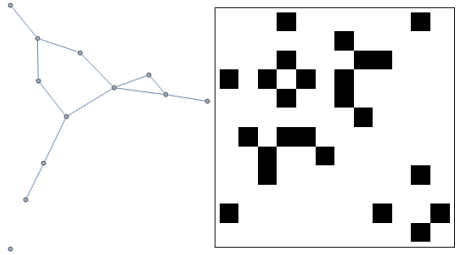
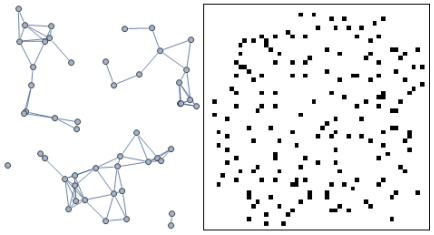
H_2

Ref: • B. Prabhakaran: "Graph Theory"

More Graphs ตัวอย่างรูปภาพ

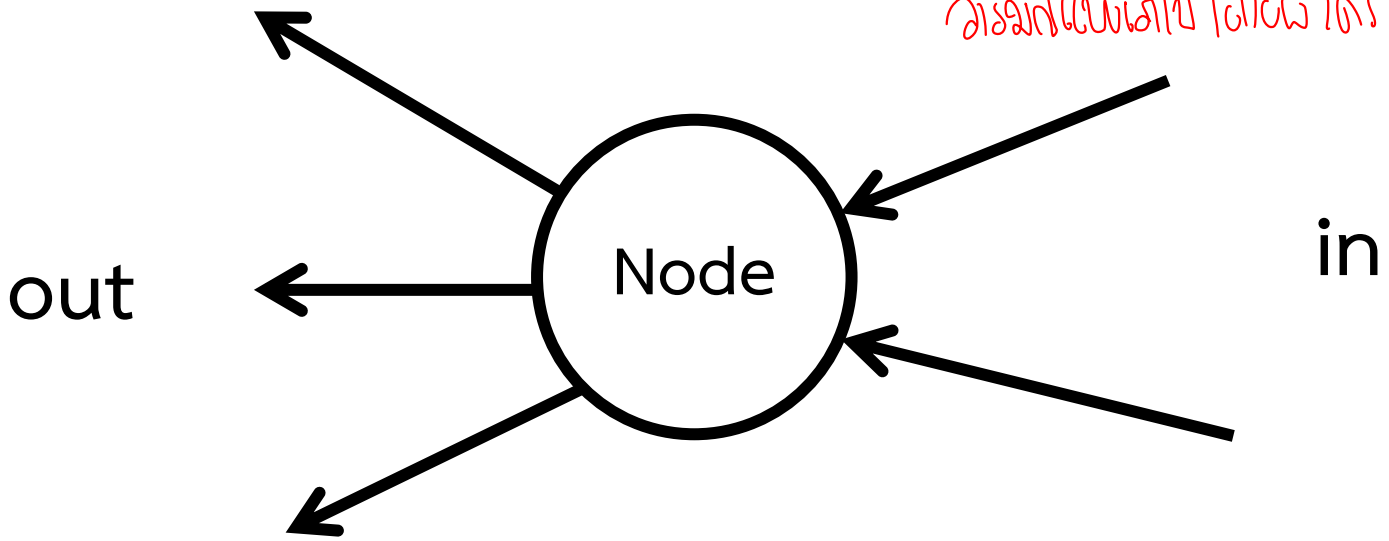
- Go to

http://www.orbifold.net/Mathematica/Networks/

Properties

Degree



degree = 5

indegree = 2

outdegree = 3

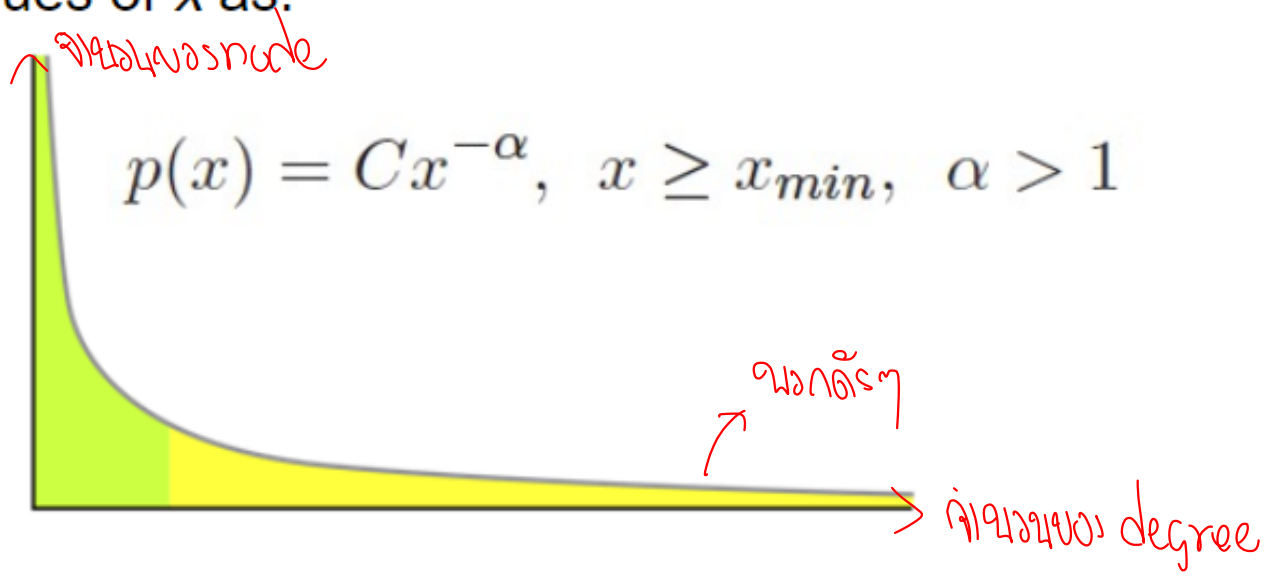
Degree

- Min Degree
- Max Degree
- Average Degree

Ref:
- (content) URL
- (image) URL

Degree Distribution

- Degree distribution in large-scale networks often follows a power law, that is, the fraction $p(x)$ of nodes in the network having x connections to other nodes goes for large values of x as:



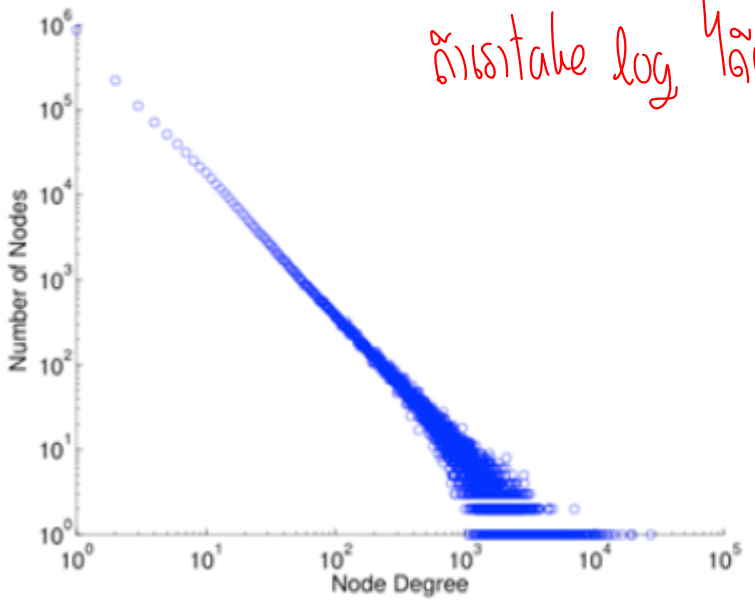
- A.k.a. long tail distribution, scale-free distribution

Ref: • http://slideplayer.com/slide/8897150/

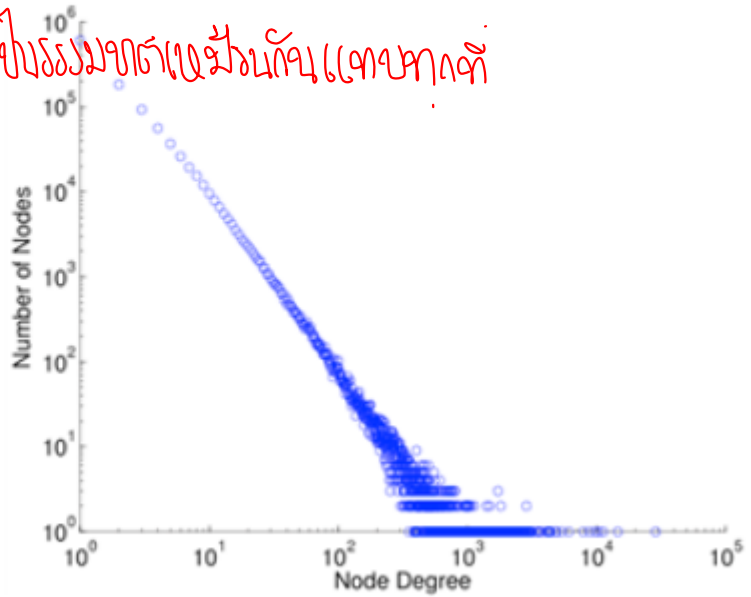
Log-Plot

- Power law distribution becomes a straight line if plotted in a log-log scale

ถ้าเราtake log ได้เป็นเส้นตรงมาตามสัดส่วนกัน



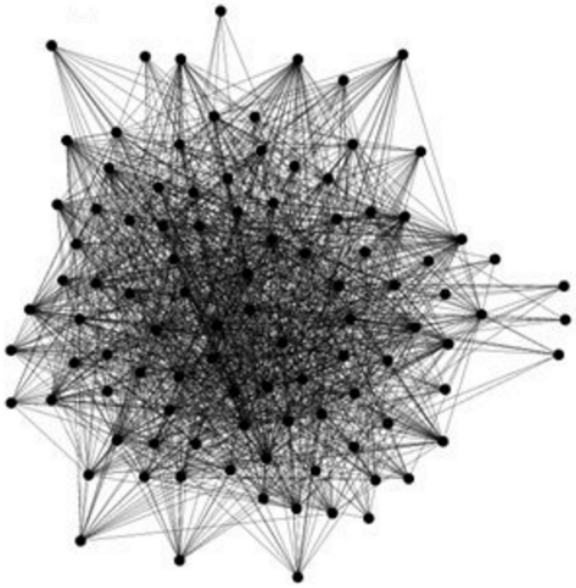
Friendship Network in Flickr



Friendship Network in YouTube

Ref: • http://slideplayer.com/slide/8897150/

Dense/Sparse



Dense Graph

Sparse Graph

Ref: • (image) https://blog.metaflow.fr/sparse-coding-a-simple-exploration-152a3c900a7c?gi=b8cb243083b1

Density

- In mathematics, a dense graph is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a sparse graph. The distinction between sparse and dense graphs is rather vague, and depends on the context.

ในการวัดการหนาแน่น

Undirected Graph

$$D = \frac{2|E|}{|V|\,(\,|V| - 1\,)}$$

ใช้สำหรับกรณีกลับ

Directed Graph

$$D = \frac{|E|}{|V|\,(\,|V| - 1\,)}$$

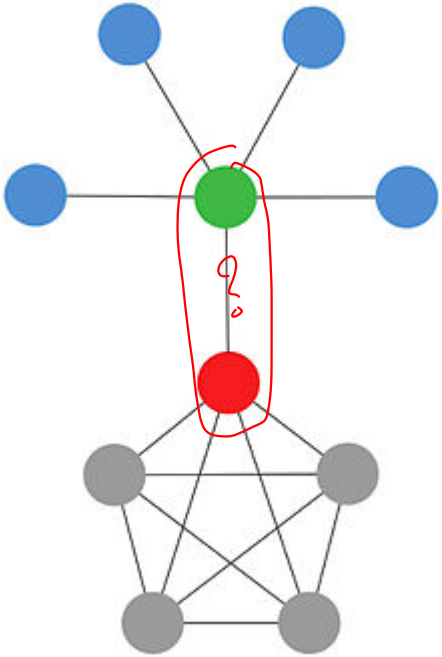
|E| จำนวน edge ของกราฟที่สนใจ

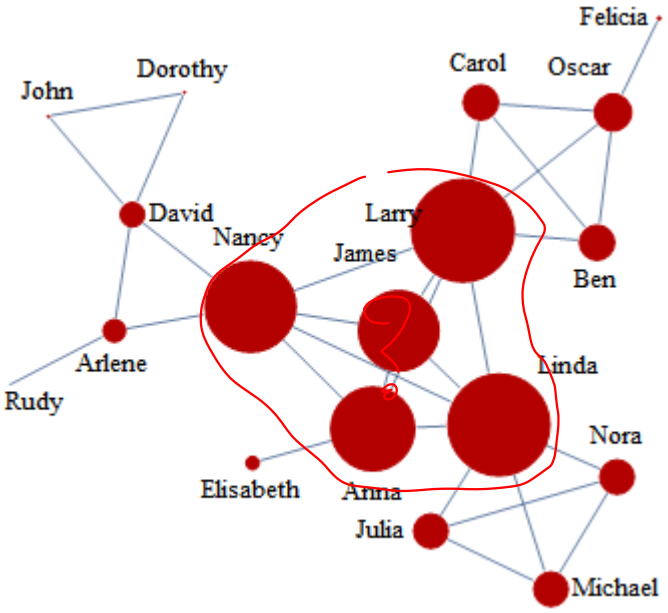
|V|(|V|−1) → จำนวนของ edge ที่มากสุดที่เป็นไปได้

= n(node) x (n(node) − 1)

Ref: • https://en.wikipedia.org/wiki/Dense_graph

Centrality





Ref:
- (image) https://en.wikipedia.org/wiki/Centrality
- (image) http://www.orbifold.net/Mathematica/Networks/

Closeness Centrality ระยะทาง

- In a connected graph, the normalized closeness centrality (or closeness) of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes. Closeness was defined by Bavelas (1950) as the reciprocal of the farness, that is:

ของ node ที่ x

$$C(x) = \frac{N-1}{\sum_y d(y,x)}$$

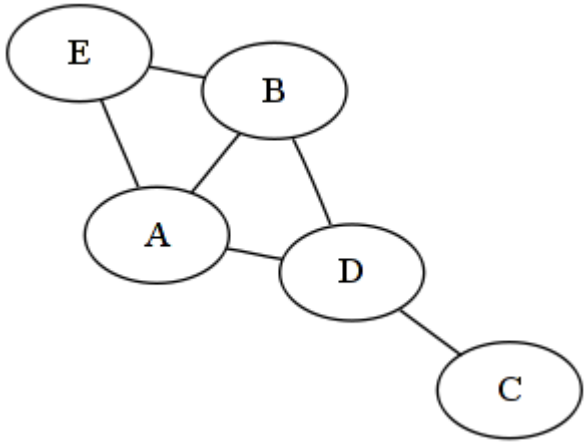
→ เส้นทางที่ใกล้ที่สุดจาก node x ไป node อื่นๆ

- where **d(x,y)** is the distance between vertices **x** and **y**. However, when speaking of closeness centrality, people usually refer to its normalized form, generally given by the previous formula multiplied by **N-1**, where **N** is the number of nodes in the graph. This adjustment allows comparisons between nodes of graphs of different sizes.
- Taking distances from or to all other nodes is irrelevant in undirected graphs, whereas it can produce totally different results in directed graphs (e.g. a website can have a high closeness centrality from outgoing link, but low closeness centrality from incoming links).

Ref: • (content) https://en.wikipedia.org/wiki/Centrality

Closeness Centrality

$$C(x) = \frac{N-1}{\sum_y d(y,x)}$$



$$C(A) = \frac{5-1}{d(B,A) + d(C,A) + d(D,A) + d(E,A)}$$

เสียเวลาที่จะไปทุก node จาก A

$$C(A) = \frac{4}{1+2+1+1} = \frac{4}{5} = 0.8$$

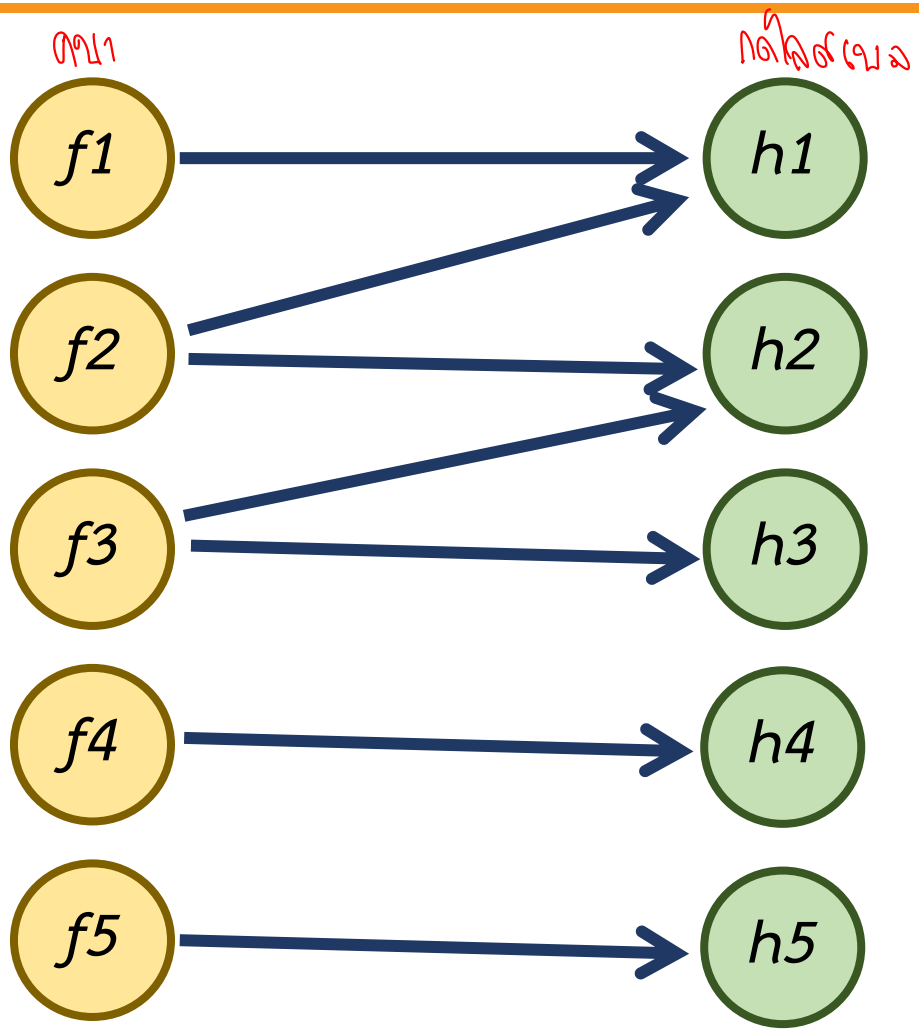
ยิ่งใกล้ 🎯 เท่ากับ

ยิ่งเวลาน้อยแสดงเป็น center เยอะ

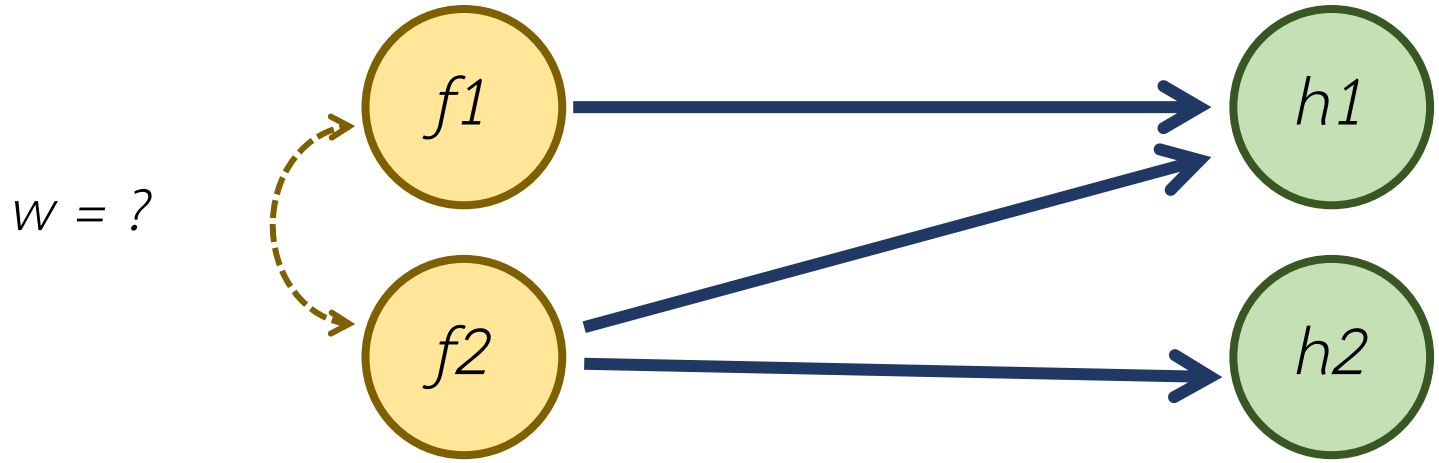
Other Centrality

- Degree centrality
- Closeness Centrality
- Harmonic centrality
- Betweenness centrality
- Eigenvector centrality
- Katz centrality
- PageRank centrality
- Percolation centrality
- Cross-clique centrality
- Freeman Centralization

Similarity



Similarity



Jaccard Index → ที่เอาที่ซ้ำ ทั้งหมด เหมือนเดิม

$$w(f1, f2) = \frac{|\Gamma(f1) \cap \Gamma(f2)|}{|\Gamma(f1) \cup \Gamma(f2)|} = \frac{|\{h1\} \cap \{h1, h2\}|}{|\{h1\} \cup \{h1, h2\}|} = \frac{|\{h1\}|}{|\{h1, h2\}|}$$

$$= \frac{1}{2} \qquad = 0.5$$

return set of f_n ใดๆ ที่เชื่อมต่ออยู่

Similarity Indices

Γ(n) is a function that returns a set of nodes that interact with the node *n*.
Example: *Γ(f2) = { h1, h2 }*

- Common Neighbors (CN) : $|\Gamma(x) \cap \Gamma(y)|$

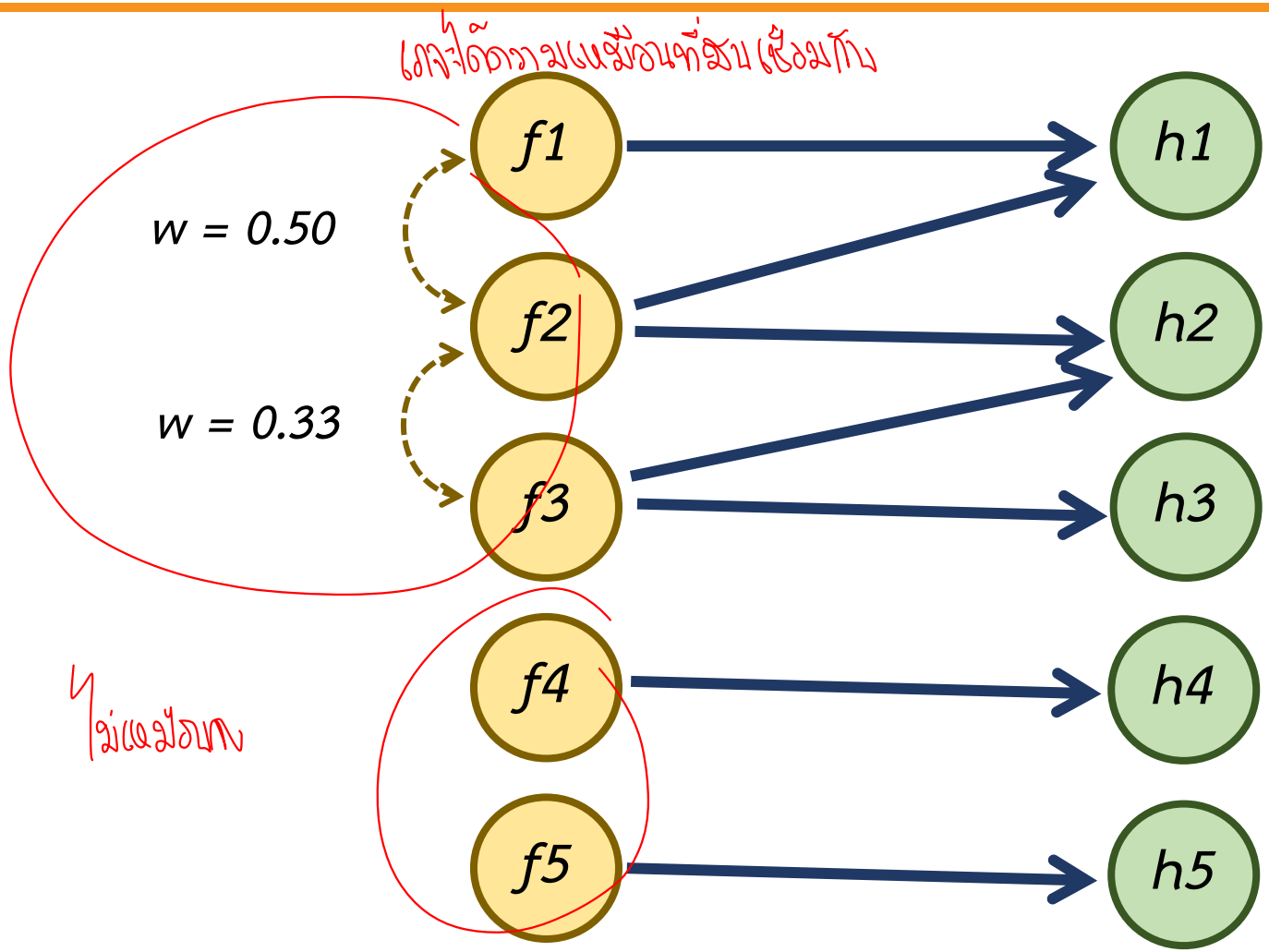
- Jaccard Index: $\frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$

- Sørensen index: $\frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|}$

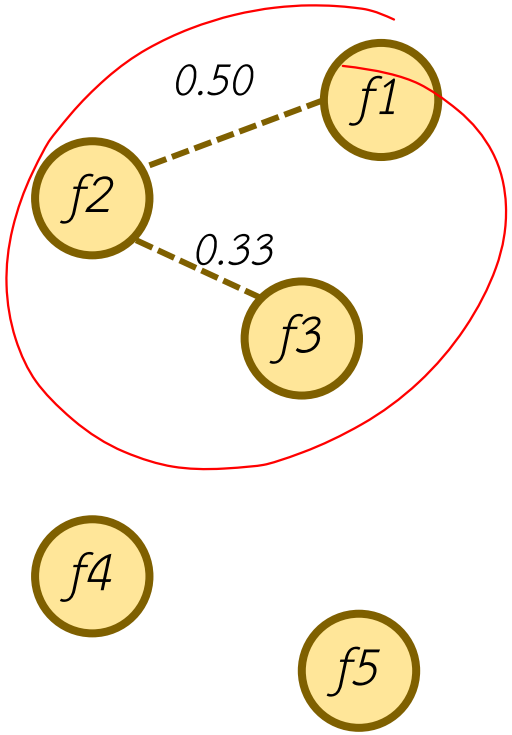
- Hub Depressed Index (HDI): $\frac{|\Gamma(x) \cap \Gamma(y)|}{\max(\Gamma(x), \Gamma(y))}$

- Resource Allocation Index (RA) : $\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|z|}$

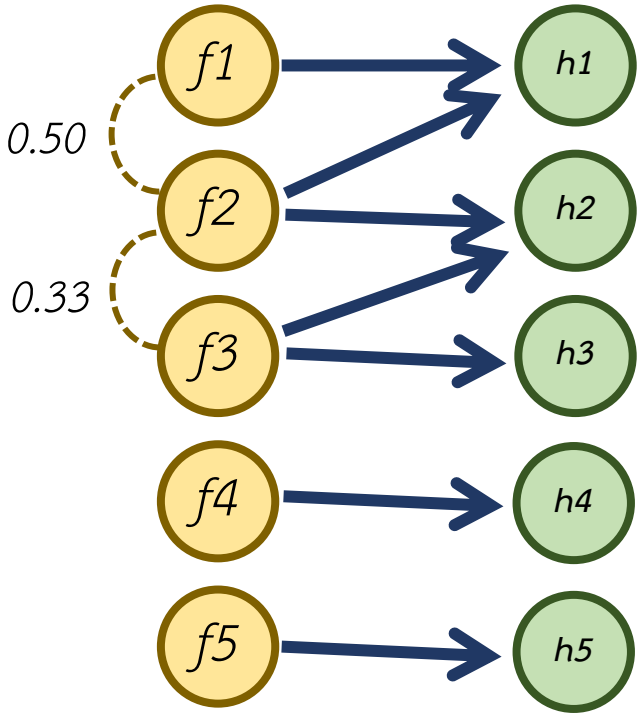
Similarity



Projection

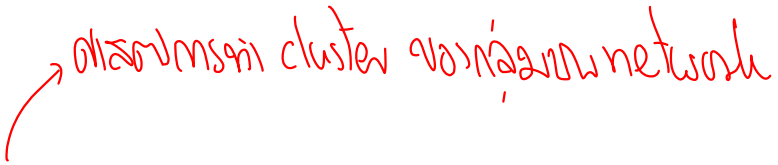


ได้กลุ่มของจุด
ที่มีความเหมือนกัน



Projection of F

Bipartite Graph



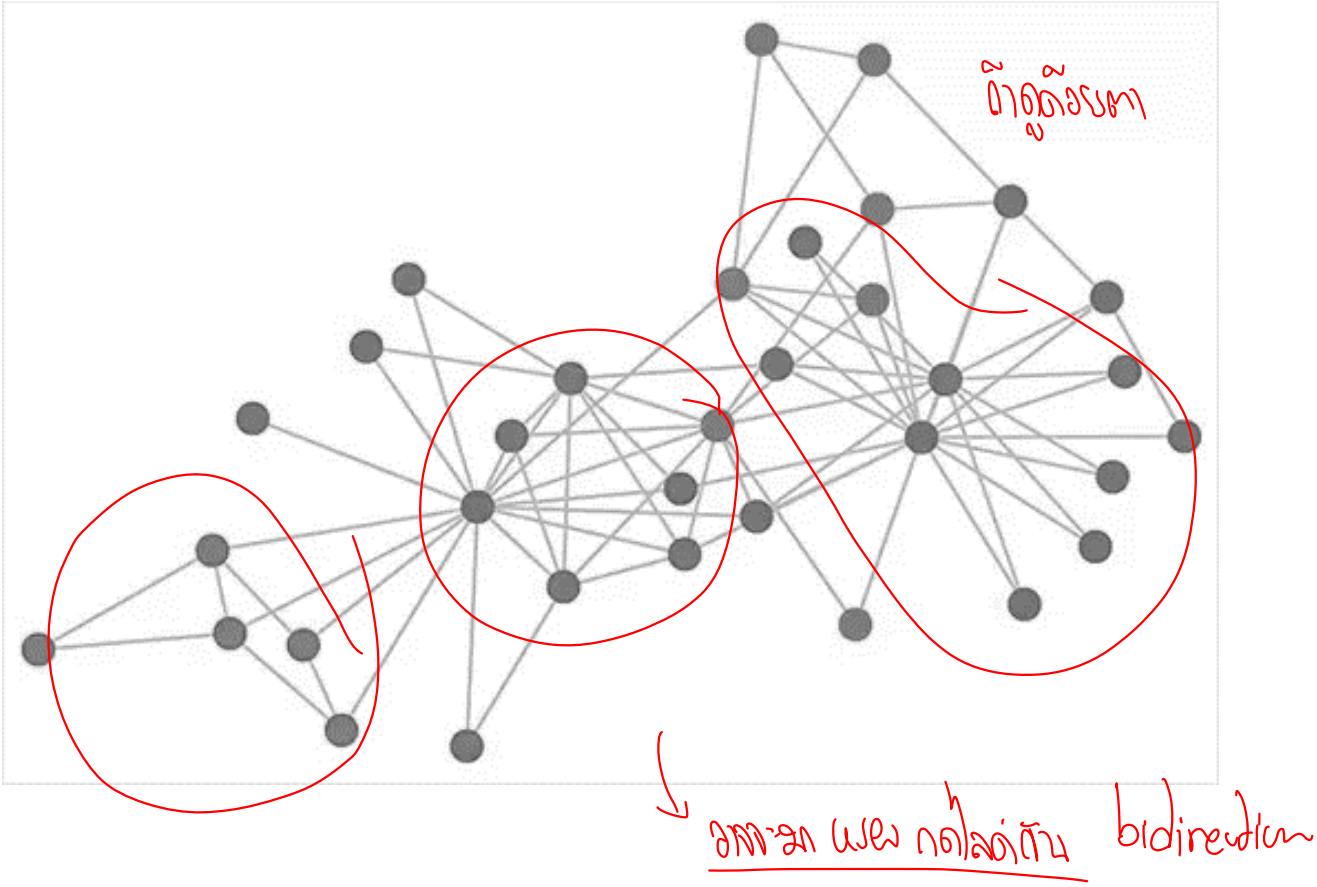
Community Detection

Modularity Maximization

- Modularity is a **measure** of the network structure.
- It was designed to **evaluate the strength of division of a network** into modules (also called groups, clusters or communities).
- Networks with high modularity have **dense connections** between the nodes within modules but sparse connections between nodes in different modules.
- Modularity is often used in **optimization methods** for detecting community structure in networks
- A simple calculation is good for unweighted and undirected graphs

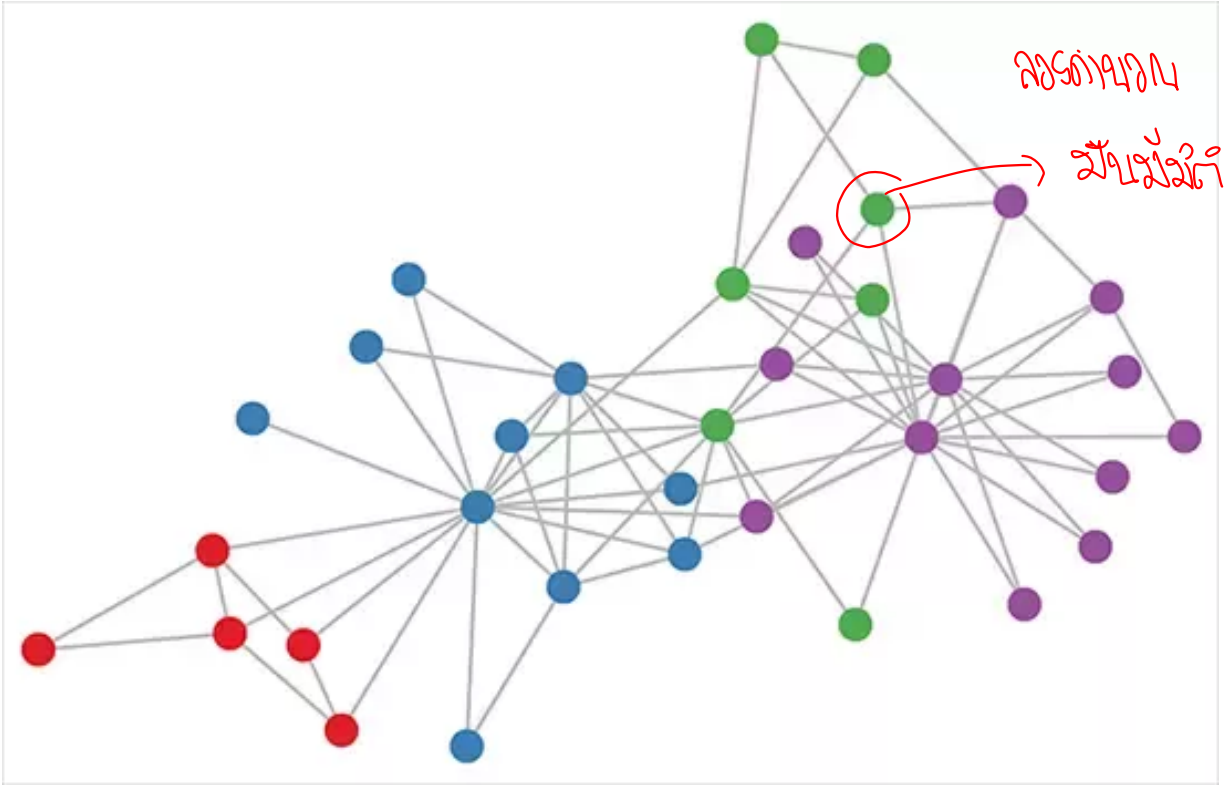
Ref: • (content) URL
• (image) URL

Community Structure in SNA



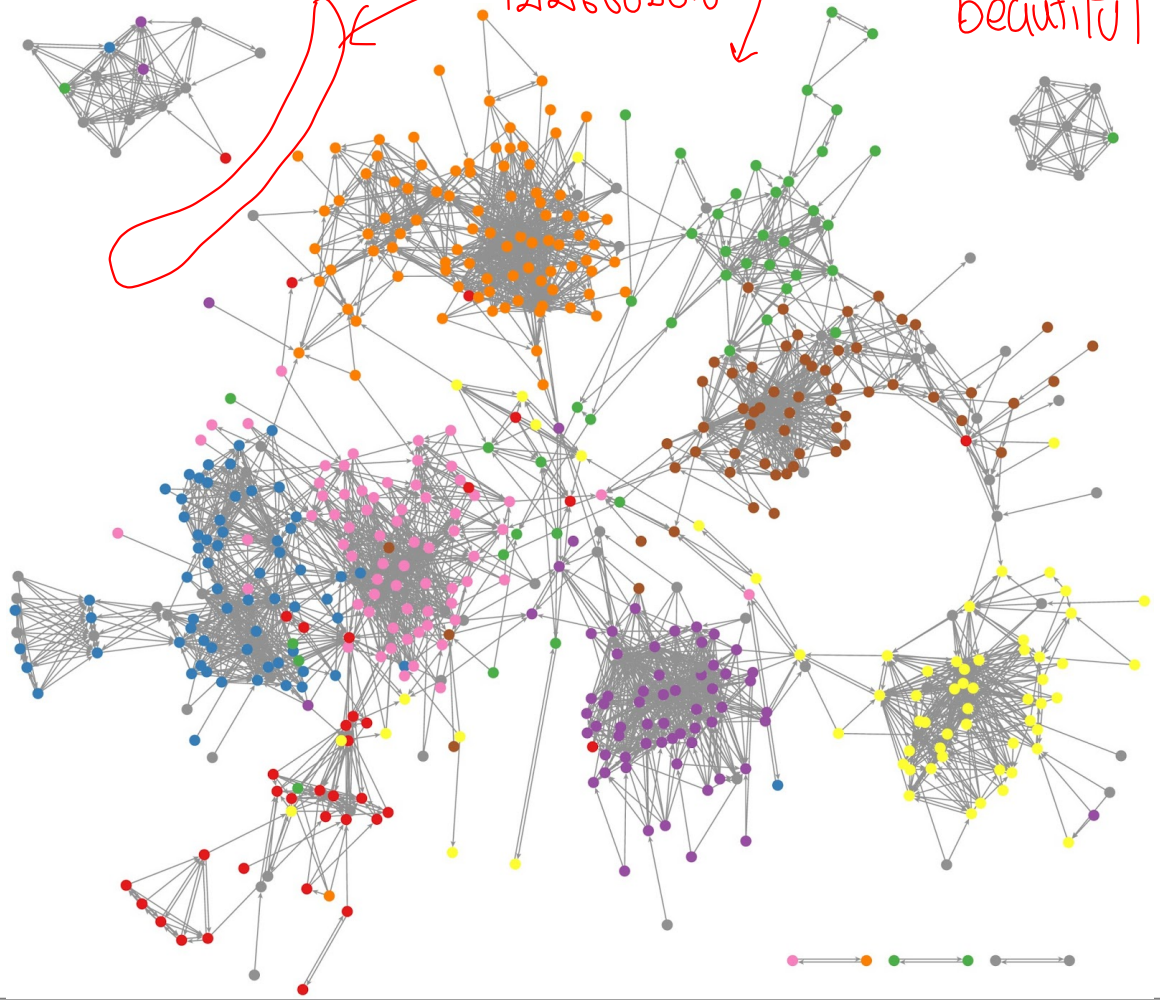
Ref: • (image) https://www.quora.com/What-are-message-passing-neural-networks

Community Structure in SNA



Ref: • (image) https://www.quora.com/What-are-message-passing-neural-networks

Community Structure in SNA



Ref: • (image) http://sociograph.blogspot.com/2013/05/revealing-community-structure-with.html

Equation

สมการในการหา

ค่าmodality

ค่าที่มีสัมพันธ์ที่สุด

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

i กับ j อยู่ในกลุ่มเดียวกัน ถึงเอาออก

Every pairs of nodes i and j being in the same community

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

m is the number of links จำนวนของ edge ทั้งหมด

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

Adjacency matrix

If having link, $A_{i,j} = 1$

If having no link, $A_{i,j} = 0$

Equation

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

probability a random edge would go between i and j

ความน่าจะเป็นที่จะเกิดขึ้นได้

Equation

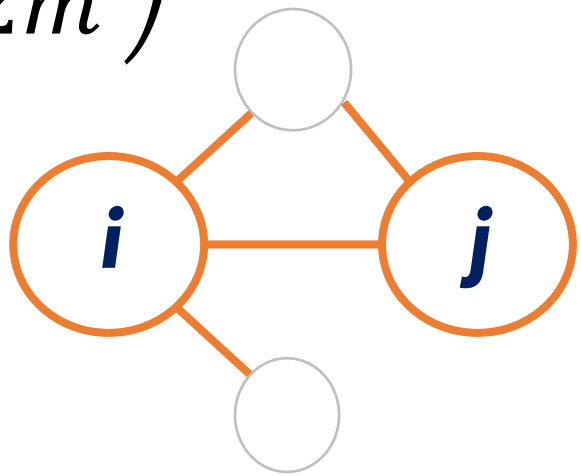
d is a degree of a node === number of edges connected

$$Q = \frac{1}{2m} \sum_{i,j \in C} \left(A_{i,j} - \frac{d_i d_j}{2m} \right)$$

In this case,

$d_i = 3$

$d_j = 2$

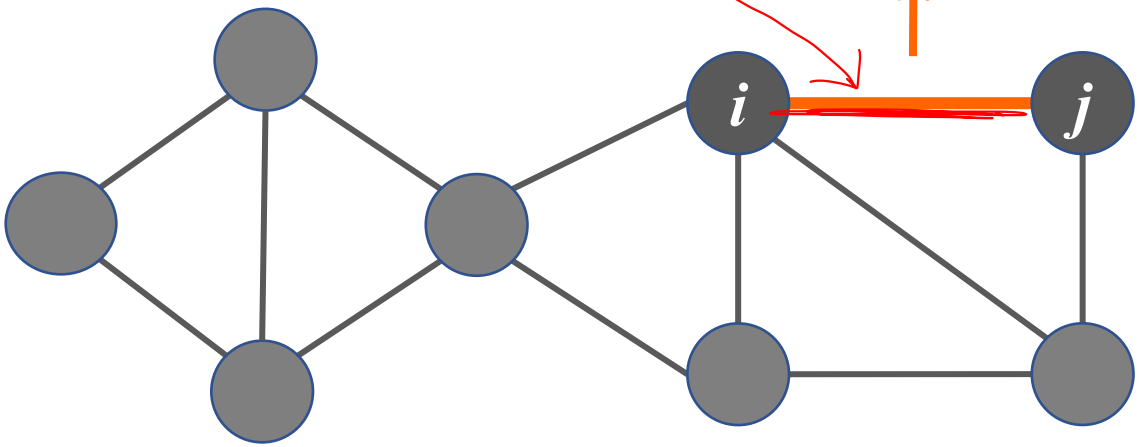


How To Calculate

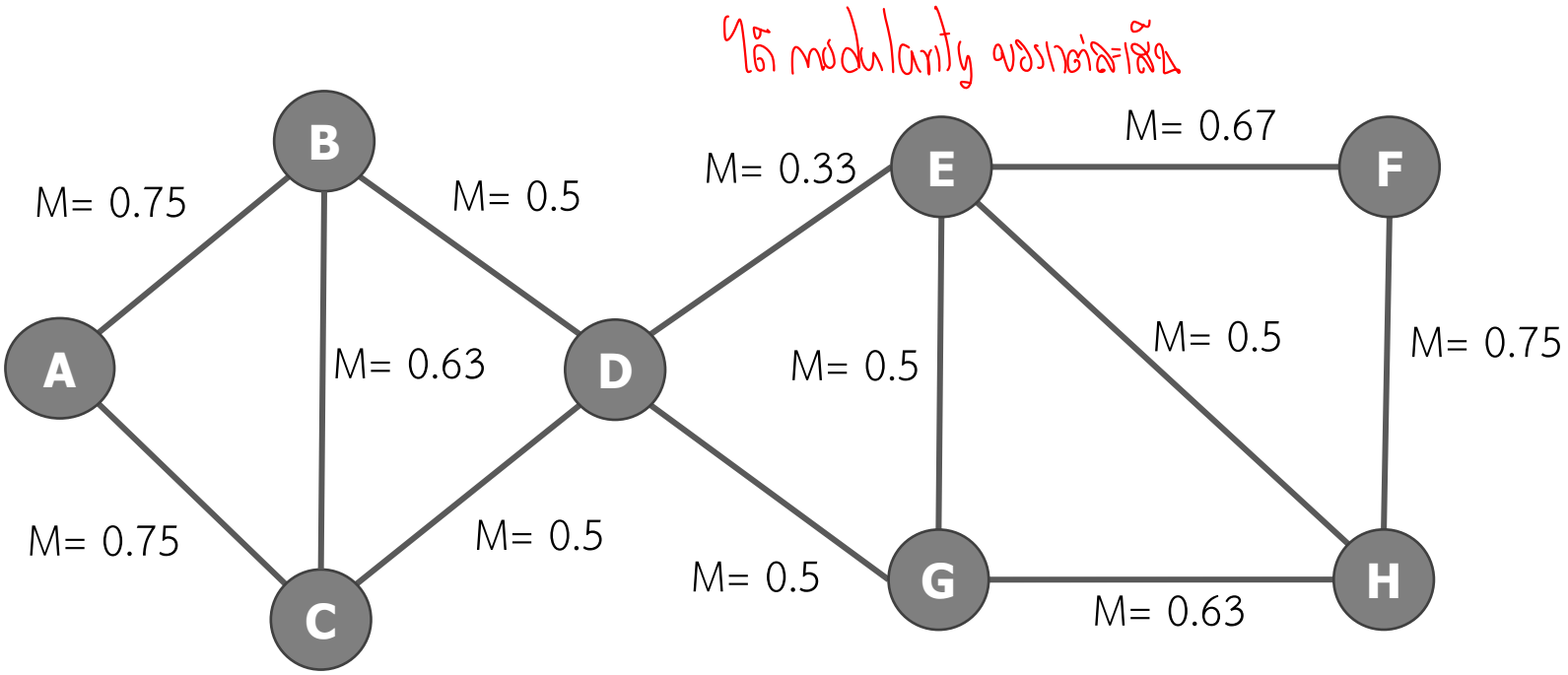
$$A_{i,j} - \frac{d_i d_j}{2m} = 1 - \frac{4 \times 2}{2 \times 12} = 0.67$$

12 edges

m = 12

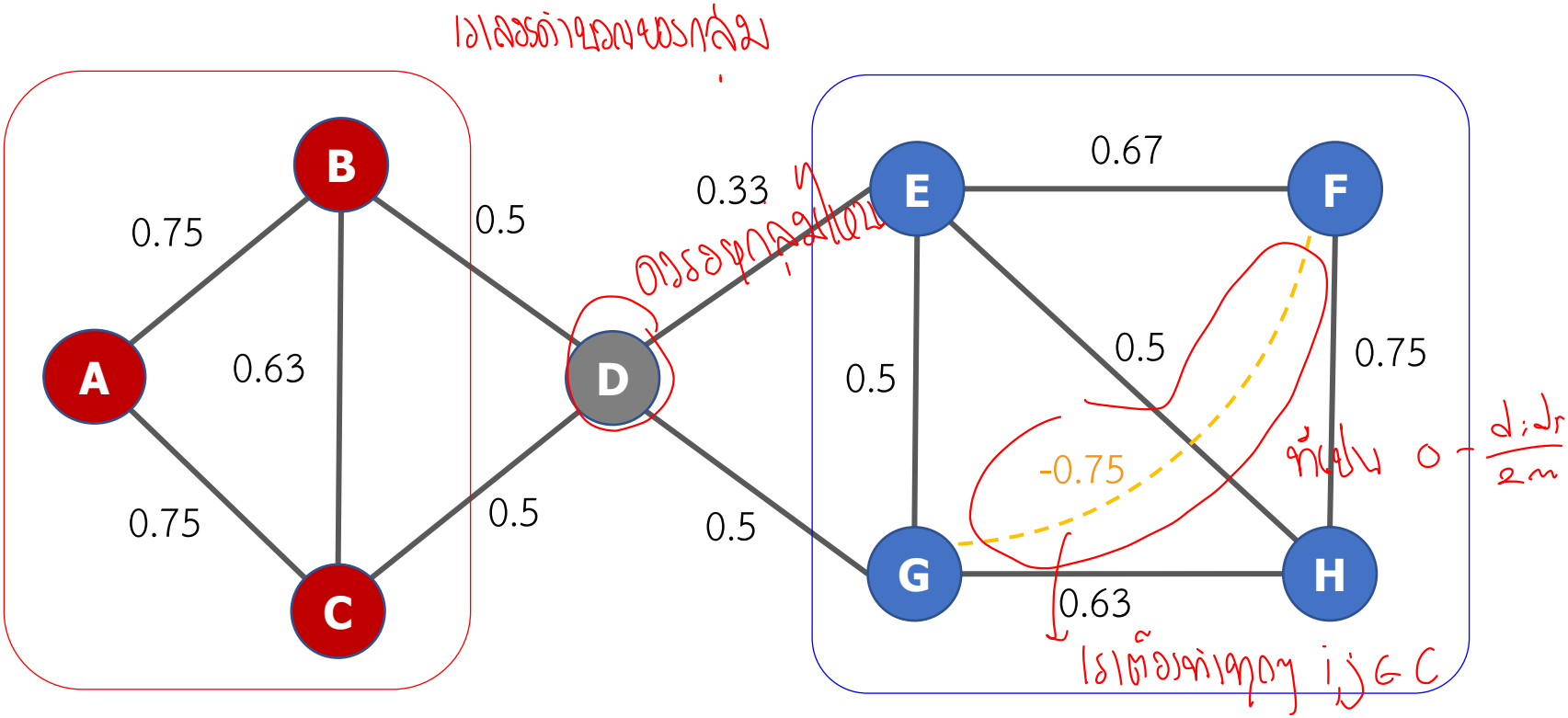


After Calculate



All edges have own Modularity Score

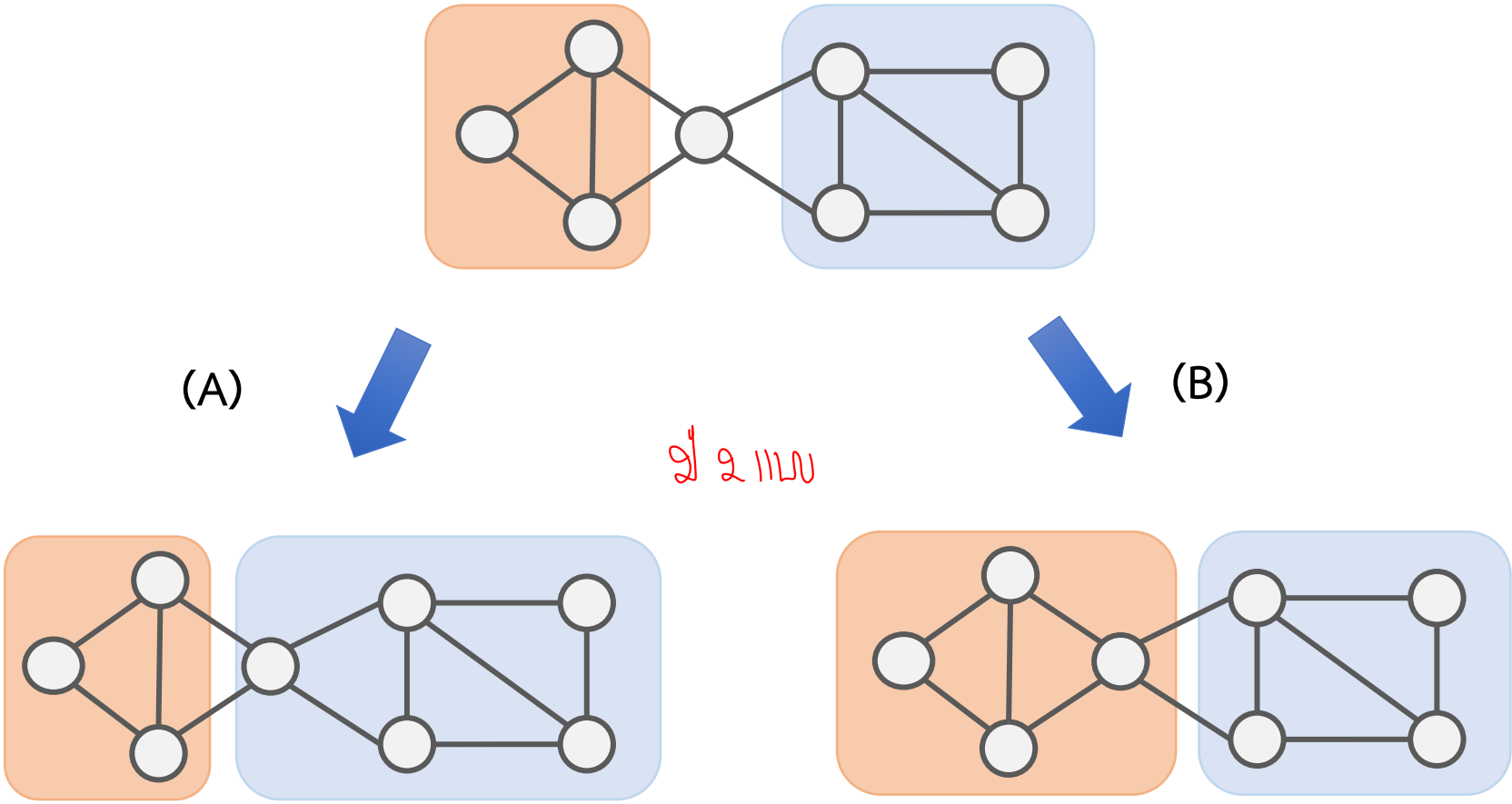
After Calculate



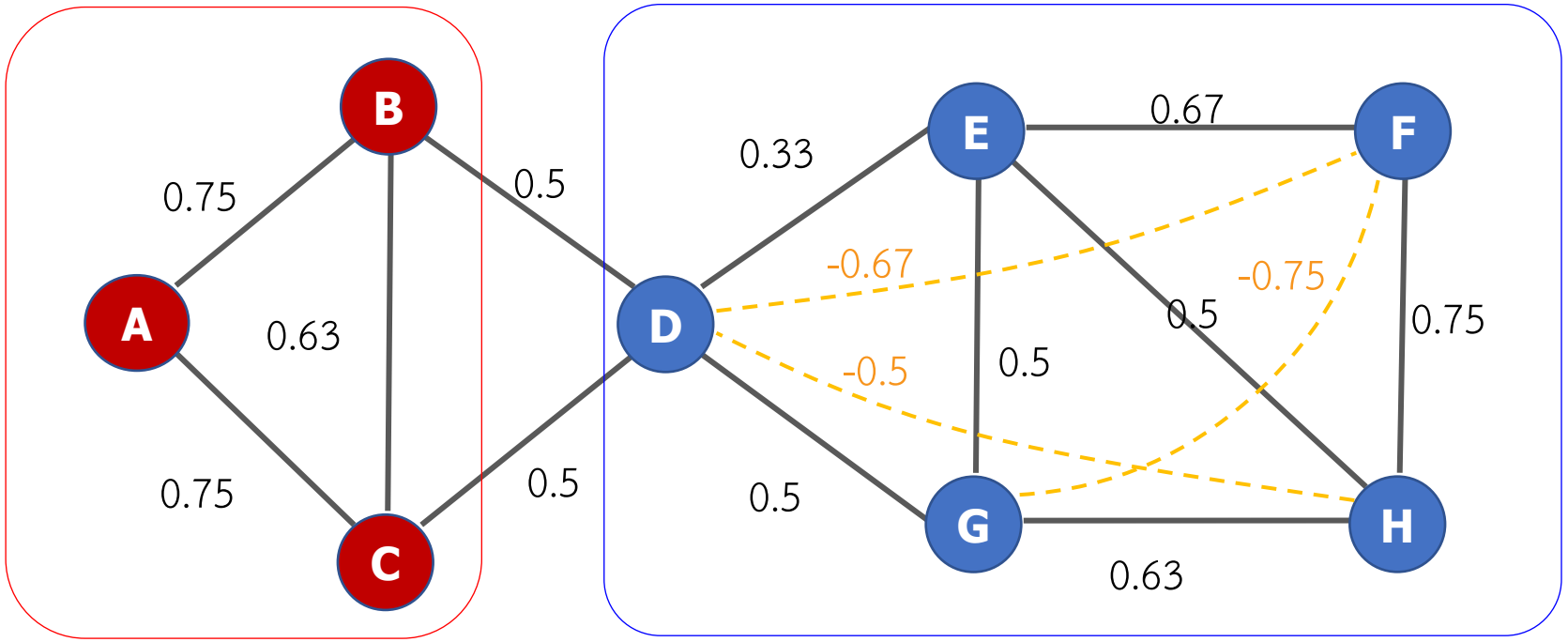
Q = ((0.75 + 0.75 + 0.63) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 –0.75))/2m

= 0.184

Next Iteration



Candidate - A

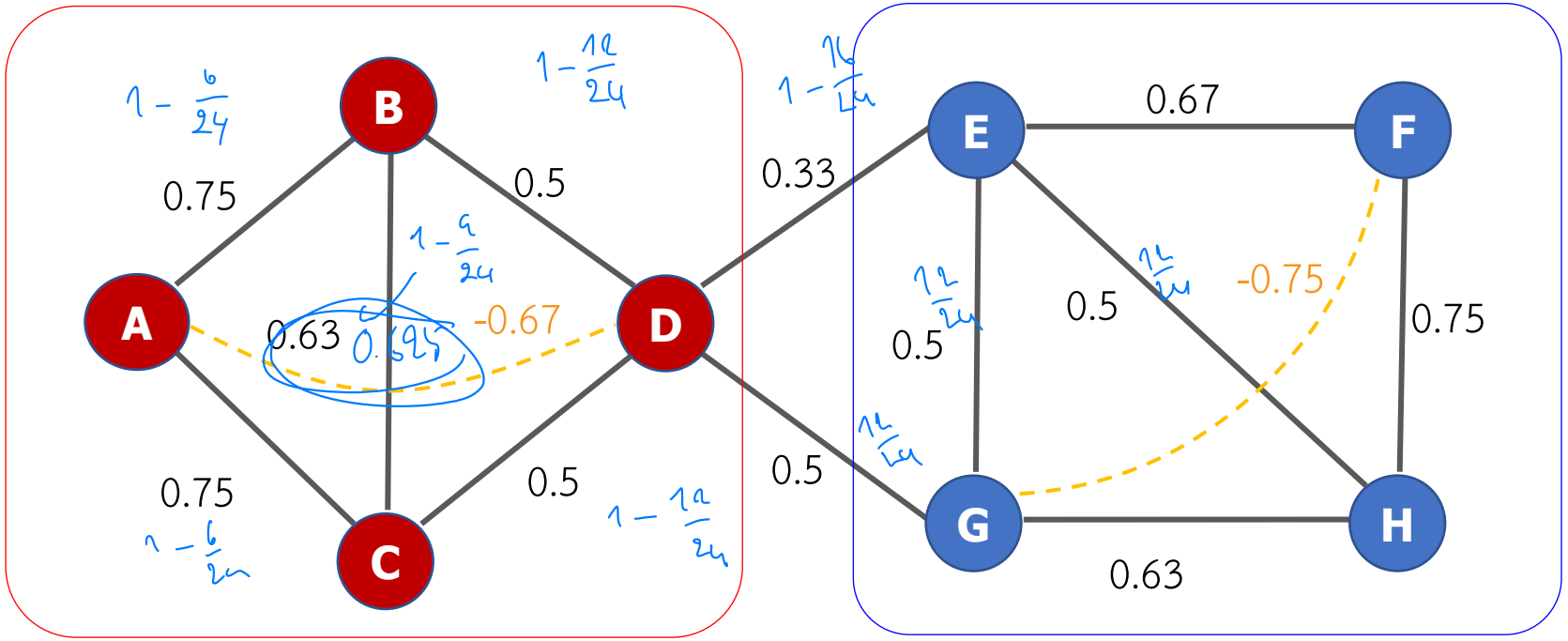


Q = ((0.75 + 0.75 + 0.63) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 +0.33 + 0.5 –0.5 -0.67 -0.75))/2m

= 0.170 ≈ สอดคล้องกับ A

(A)

Candidate - B



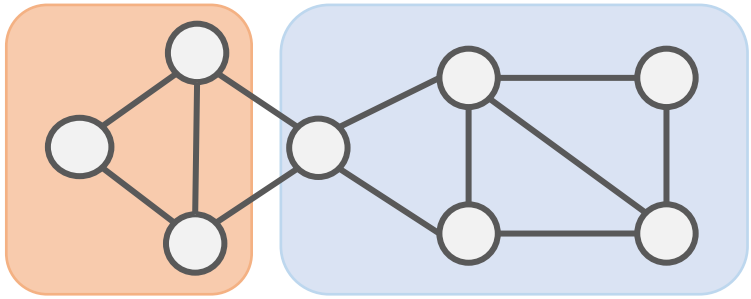
Q = ((0.75 + 0.75 + 0.63 + 0.5 + 0.5 −0.67) + (0.5 + 0.5 + 0.63 + 0.67 + 0.75 −0.75))/2m

= 0.198

(B)

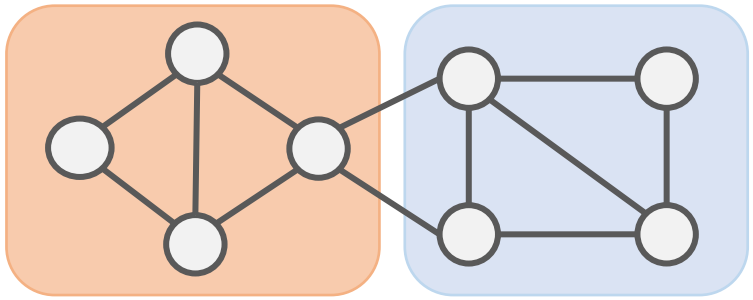
We Choose

(A)

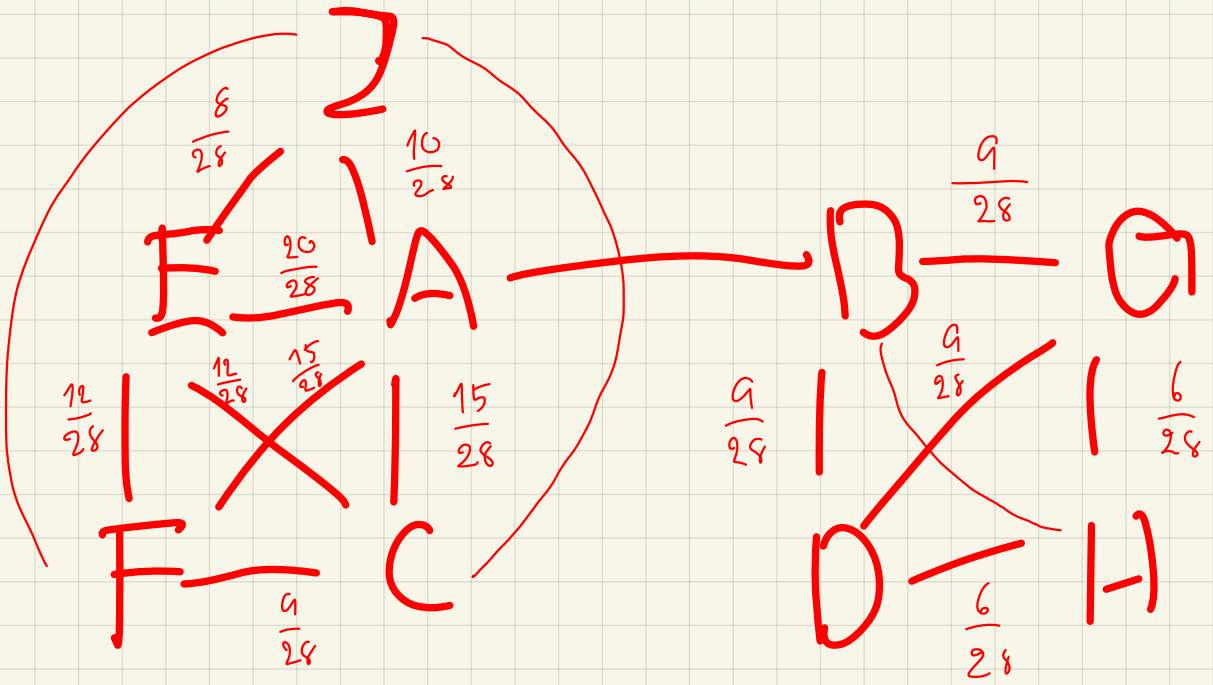


Q = 0.170

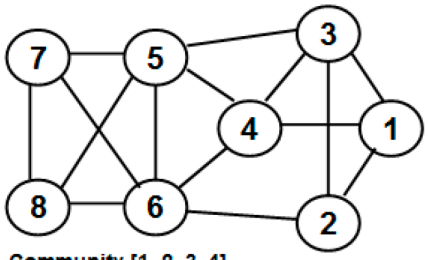
(B)



Q = 0.198


J
8/28
10/28
E
20/28
A
12/28
15/28
12/28
15/28
F
C
9/28
B
9/28
9/28
6/28
D
6/28
H

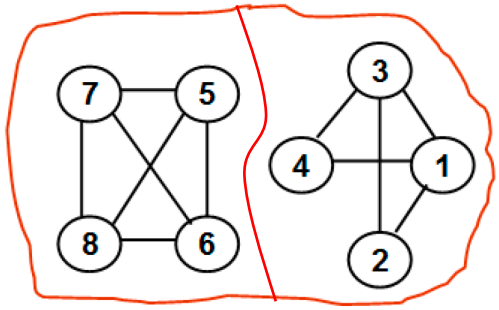
Other Example



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]

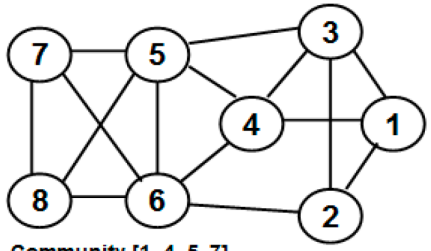
Edges with Aij = 1	Modularity
5 – 6	1 – (5)(5)/(2*15) = 0.17
5 – 7	1 – (3)(5)/(2*15) = 0.50
5 – 8	1 – (3)(5)/(2*15) = 0.50
6 – 7	1 – (3)(5)/(2*15) = 0.50
6 – 8	1 – (3)(5)/(2*15) = 0.50
7 – 8	1 – (3)(3)/(2*15) = 0.70

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

Cumulative Modularity Score for the two Communities: 2.67 + 2.87 = 5.54

(a)

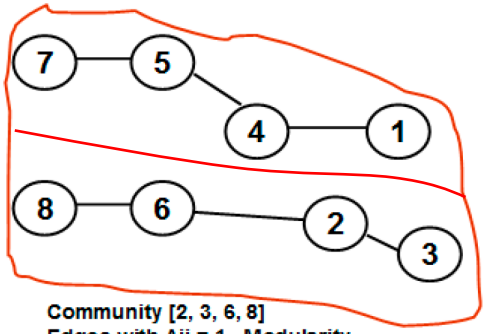
Q= 5.54/2m



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

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



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

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

Cumulative Modularity Score for the two Communities: 0.23 + 0.23 = 0.46

(b)

Q= 0.44/2m

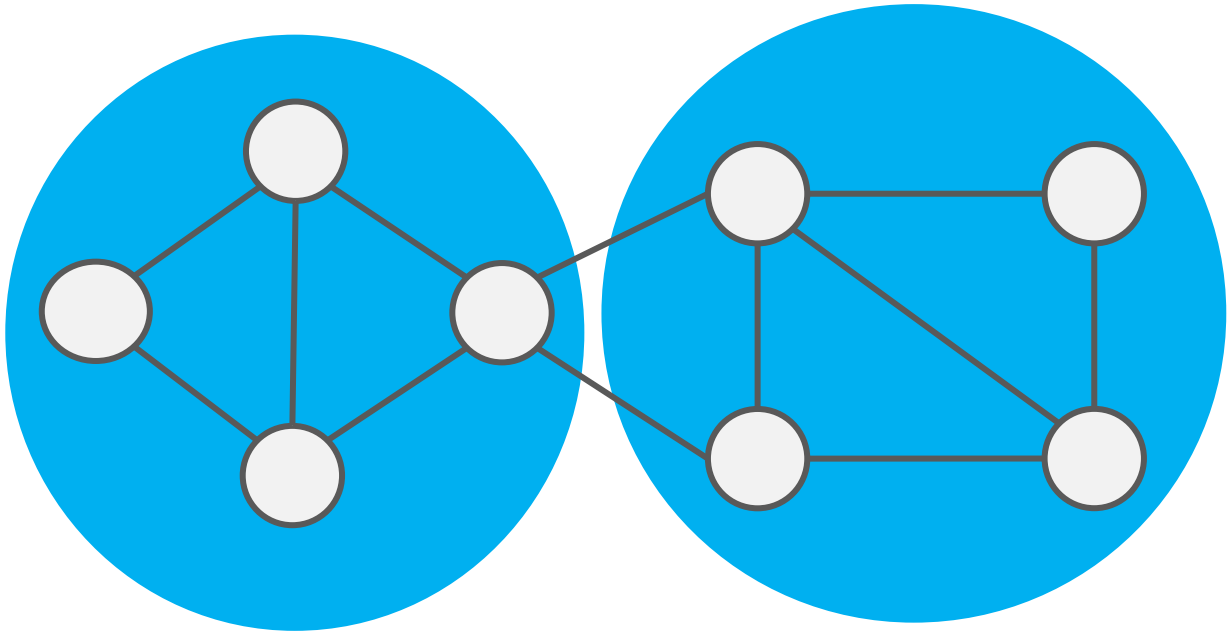
Ref: • N. Meghanathan: "A Greedy Algorithm for Neighborhood Overlap-Based Community Detection"

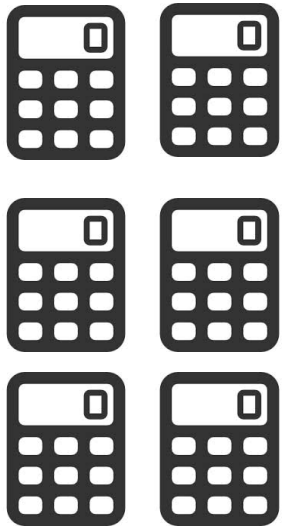
Walk Trap

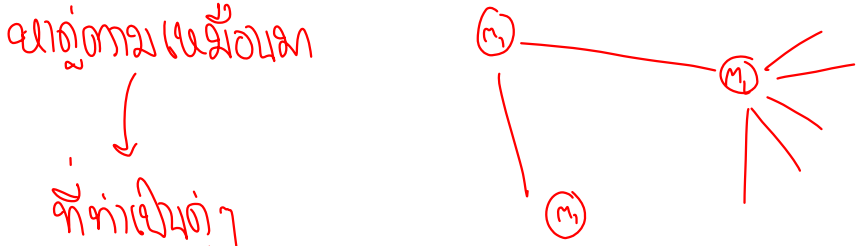
เหมือนไม่สวย

Random Walk

- Calculate **Whole Modularity** in every iteration







Every once in a while, a new technology, an old problem, and a big idea turn into an innovation.

ชื่อเต็ม

เพิ่มเติม ของ visualize graph อื่นๆ Gephi → Java → Google & free

Dean Kamen

ยกตัวอย่าง

つづく