สรุปเนื้อหา-ทบทวนก่อนสอบ

Week15

การออกแบบและวิเคราะห์ขั้นตอนวิธี

DESIGN AND ANALYSIS OF ALGORITHMS

อ.ธิดาวรรร คล้ายศรี

Algorithm/ขั้นตอนวิธี

❖ What? → a method to solve problem

วิธีการ/เครื่องมือที่ช่วยแก้ปัญหาอย่างเป็นขั้นตอนตามลำดับ เพื่อให้ได้ผลลัพธ์อย่างมีประสิทธิภาพ บางปัญหาต้องการวิธีการทางคอมพิวเตอร์ (computer algorithm)

กระบวนการทำงาน (procedure) ใน Algorithm เพื่อใช้แก้ปัญหาให้บรรลุเป้าหมายนั้นจะต้อง

- มีขั้นตอนการดำเนินงานที่ชัดเจน
- กระบวนการทำงานควรไม่ซับซ้อน เพื่อที่จะได้สามารถนำไปปฏิบัติการโดยเครื่องคอมพิวเตอร์
- กระบวนการทำงานมีจุดจบ (finiteness)

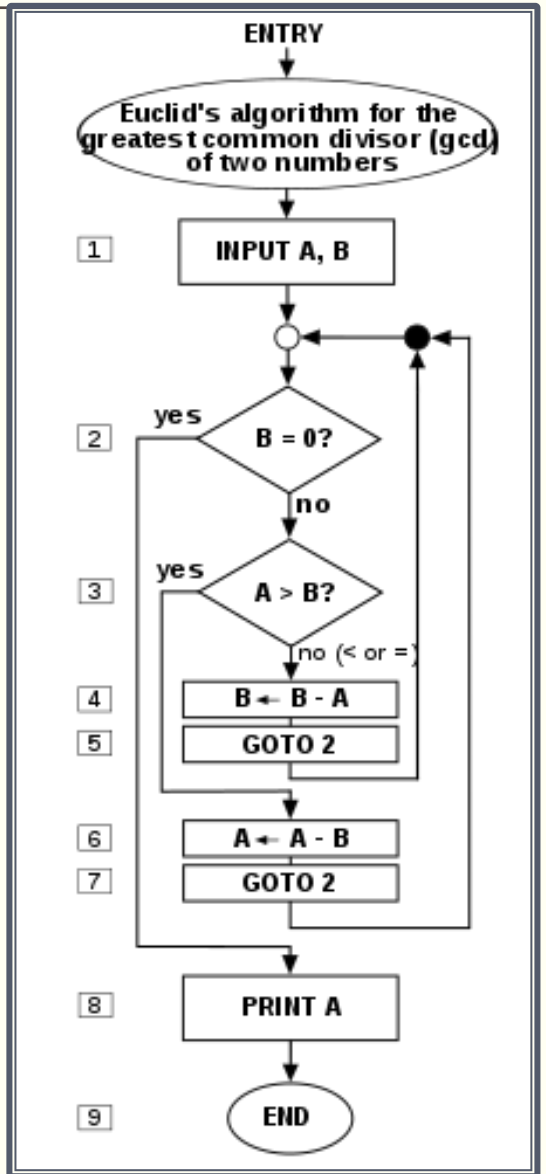
❖ Why? → heuristic/สามัญสำนึก ไม่สามารถแก้ปัญหาได้

Algorithm/ขั้นตอนวิธี

❖ How? → Pseudo code, Flow chart or natural languages

```
Algorithm LargestNumber
  Input: A list of numbers L.
  Output: The largest number in the list L.


  if L.size = 0 return null
  largest ← L[0]
  for each item in L, do
    if item > largest, then
      largest ← item
  return largest
```



[ภาพ: Wikipedia]

Analysis of Algorithm/การวิเคราะห์ขั้นตอนวิธี

ประสิทธิภาพของอัลกอริธึ่ม

⇨ การวิเคราะห์ Time Complexity (ความซับซ้อนทางด้านเวลา)

❖ เวลาที่เครื่องคอมพิวเตอร์ต้องใช้ในการประมวลผลอัลกอริธึ่ม ซึ่งวิเคราะห์เพื่อ:

- ประมาณการเวลาทั้งหมดที่ต้องใช้ในโปรแกรมได้
- มุ่งการแก้ไขไปที่อัลกอริธึ่มที่ใช้เวลาในการประมวลผลนานๆ ทำให้ไม่ต้องแก้ไขทั้งโปรแกรม
- เลือกคุณลักษณะของคอมพิวเตอร์/ทรัพยากร ที่จะใช้ติดตั้งโปรแกรมที่พัฒนาขึ้นได้อย่างเหมาะสม

Analysis of Algorithm/การวิเคราะห์ขั้นตอนวิธี

ประสิทธิภาพของอัลกอริธึ่ม *เราทราบได้อย่างไรว่าจะมีประสิทธิภาพ*

⇨ การวิเคราะห์ Time Complexity (ความซับซ้อนทางด้านเวลา)

❖ พิจารณาจำนวนอิลิเมนต์ (Elements) ที่จะประมวลผลหรือจากจำนวนรอบการทำงานของตัวดำเนินการนั้นๆ

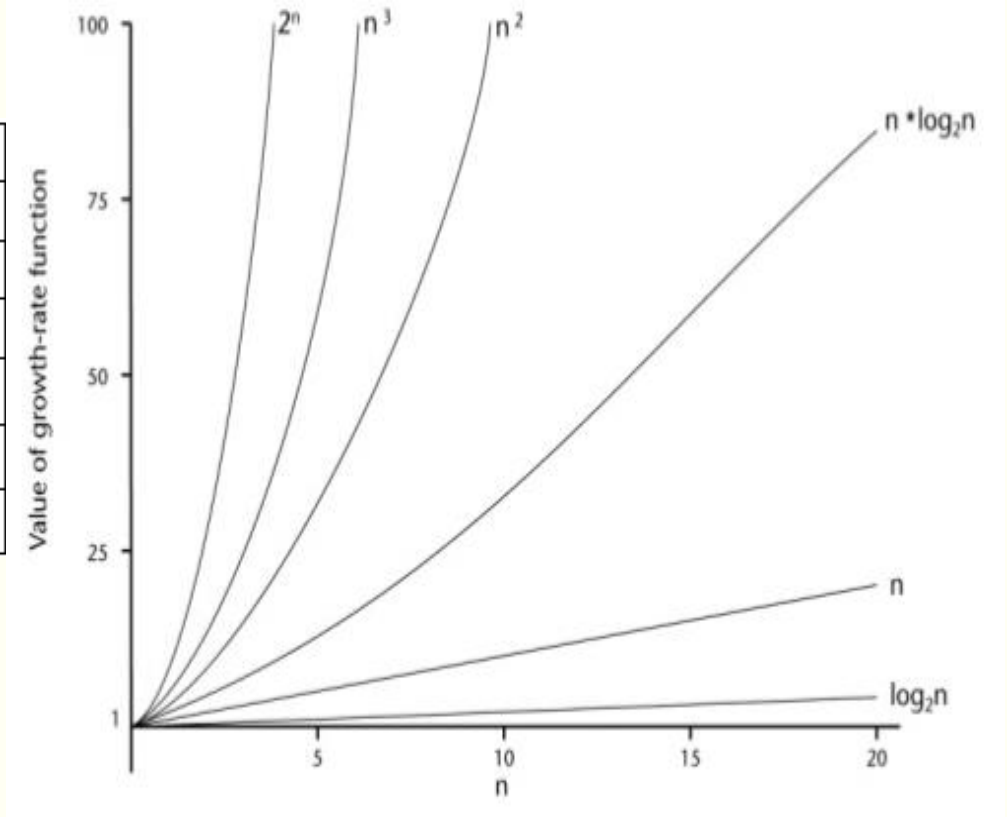
- f(n) = efficiency

→ อัตราการเติบโตของฟังก์ชั่น (growth rates) ที่บอกความสัมพันธ์ระหว่างจำนวนข้อมูลนำเข้ากับความเร็วในการประมวลผล

- พิจารณาจากส่วนการทำงานของวนรอบประมวลผล (loop) เป็นสำคัญ

Functions of Growth Rate (อัตราการเติบโตของฟังก์ชั่น)

ประสิทธิภาพฟังก์ชั่น	ชื่อฟังก์ชั่น	ตัวอย่างอัลกอริธึ่ม
$\log_2 n$	Logarithmic	Binary search
n	Linear	Graph traversal
$n \log_2 n$	$n \log_2 n$	Merge/ quick sort
n^2	Quadratic	Shortest path
n^3	Cubic	Dynamic programming
2^n	Exponential	Traveling salesman



[ภาพ: internet 2017]

Analysis of Algorithm/การวิเคราะห์ขั้นตอนวิธี

Big-O notation

- การใช้สัญลักษณ์ O มีวัตถุประสงค์ใช้วิเคราะห์ประสิทธิภาพหรือความซับซ้อนทางด้านเวลาฟังก์ชั่นของอัลกอริธึ่มหนึ่งๆ เพื่อเลือกอัลกอริธึ่มที่มีประสิทธิภาพสูงสุด (ไม่ได้มีวัตถุประสงค์เพื่อวัดเวลา)

โดยพิจารณาจากฟังก์ชันการเติบโตทางด้านเวลา ของอัลกอริธึ่ม

ว่าเมื่อข้อมูลเข้ามีปริมาณเพิ่มมากขึ้น มีผลต่อ running time เพียงใด

- โดยจะพิจารณาปริมาณอินพุทข้อมูลมากๆ → Worst-case
- เช่น $O(\log_2 n)$ จะมีประสิทธิภาพสูงกว่า $O(2^n)$

Solving Problem with Divide-and-Conquer
(การแก้ปัญหาด้วยเทคนิคการแบ่งย่อยเพื่อพิชิต)

แบ่งปัญหาใหญ่ (ขนาด/ความซับซ้อน/ยาก) ออกเป็นปัญหาย่อยๆ (sub-programs/sub-problems) เพื่อที่จะได้แก้ไขได้ง่ายขึ้น แล้ว แก้ไขนำปัญหาย่อยเหล่านั้น แล้วนำผลลัพธ์มาผสมกันได้เป็นผลลัพธ์ของปัญหาหลักนั้น

3 Parts

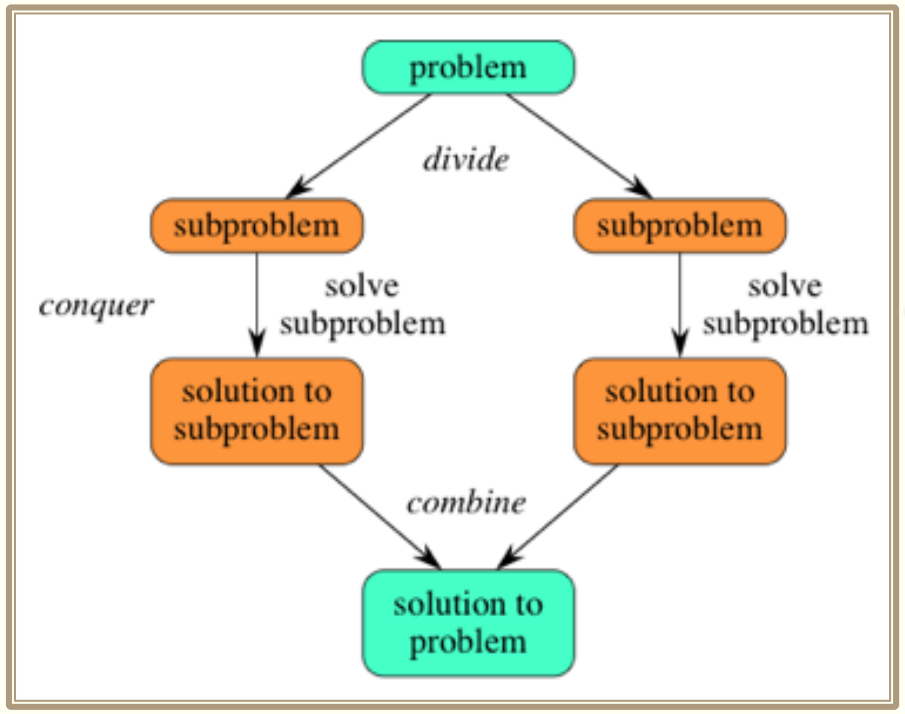
1. **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
2. **Conquer** the subproblems by solving them recursively. If they are small enough, solve the subproblems as base cases.
3. **Combine** the solutions to the subproblems into the solution for the original problem.

Breaking problem

Solving problem

Combining solutions

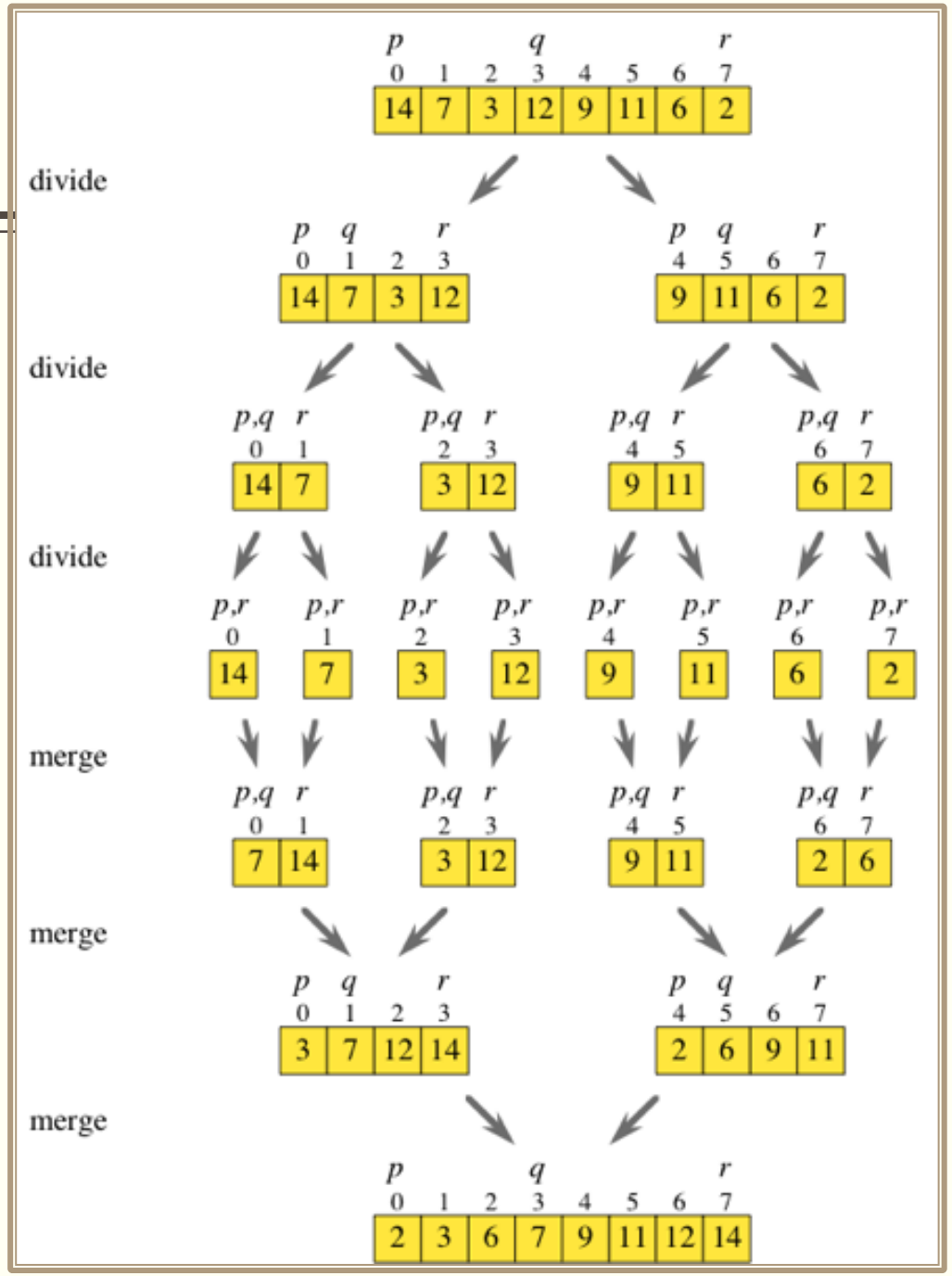
Divide-and-Conquer Approach



Breaking problem

Solving problems

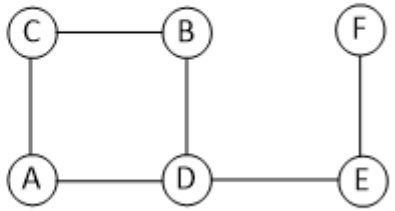
Combining solutions



Content: https://www.khanacademy.org

Graph Theory/Social Network Analysis (SNA)

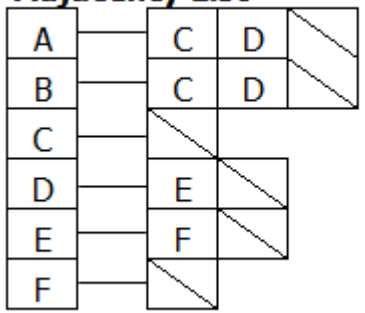
V1	V2
A	C
A	D
B	C
B	D
D	E
E	F



Adjacency Matrix

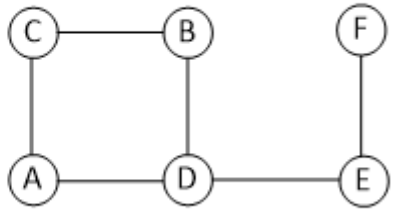
	A	B	C	D	E	F
A	-	0	1	1	0	0
B	0	-	1	1	0	0
C	1	1	-	0	0	0
D	1	1	0	-	1	0
E	0	0	0	1	-	1
F	0	0	0	0	1	-

Adjacency List



Graph Theory/Social Network Analysis (SNA)

V1	V2
A	C
A	D
B	C
B	D
D	E
E	F

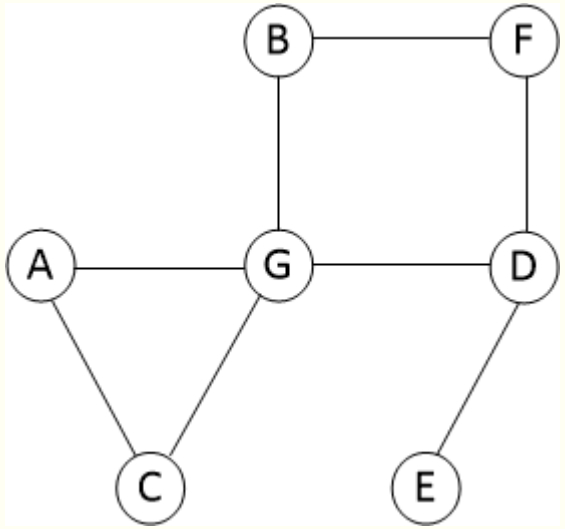


All Pairs Shorstest Paths

	A	B	C	D	E	F
A	-	2	1	1	2	3
B	2	-	1	1	2	3
C	1		-			
D	1			-		
E	2				-	
F	3					-

G = (V, E) เป็นกราฟแบบไม่มีทิศทาง (undirected graph) โดยมี Vertices, V = {A, B, C, D, E, F, G} และ Edges, E = {(A,C), (A,G),…(D,F)}

v1	v2
A	C
A	G
B	F
B	G
C	G
D	E
D	G
D	F

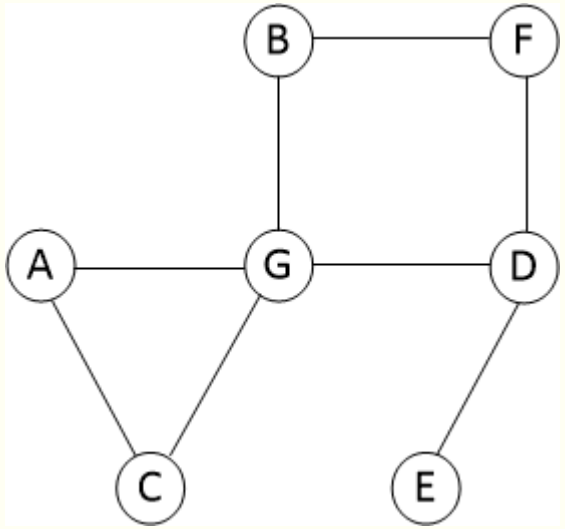


Adjacency Matrix

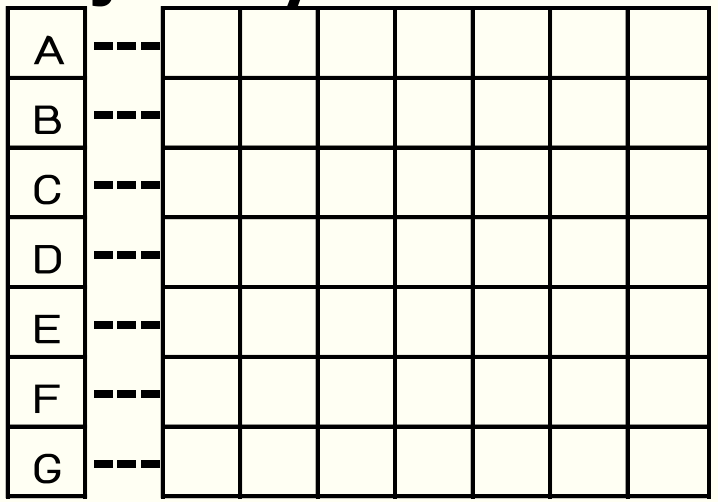
	A	B	C	D	E	F	G
A	-						
B		-					
C			-				
D				-			
E					-		
F						-	
G							-

G = (V, E) เป็นกราฟแบบไม่มีทิศทาง (undirected graph) โดยมี Vertices, V = {A, B, C, D, E, F, G} และ Edges, E = {(A,C), (A,G),…(D,F)}

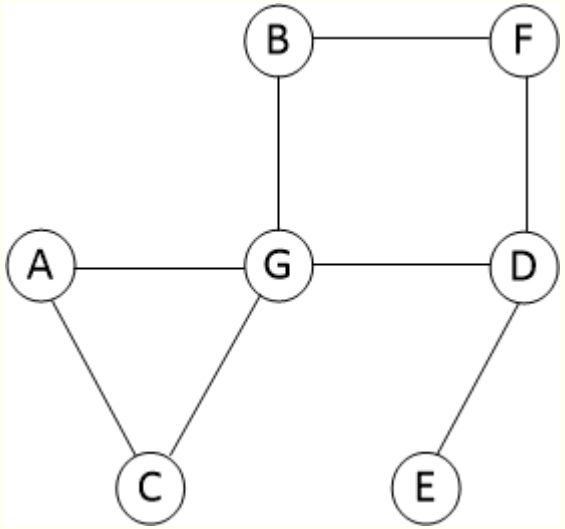
v1	v2
A	C
A	G
B	F
B	G
C	G
D	E
D	G
D	F



Adjacency List



G = (V, E) เป็นกราฟแบบไม่มีทิศทาง (undirected graph) โดยมี Vertices, V = {A, B, C, D, E, F, G} และ Edges, E = {(A,C), (A,G),…(D,F)}

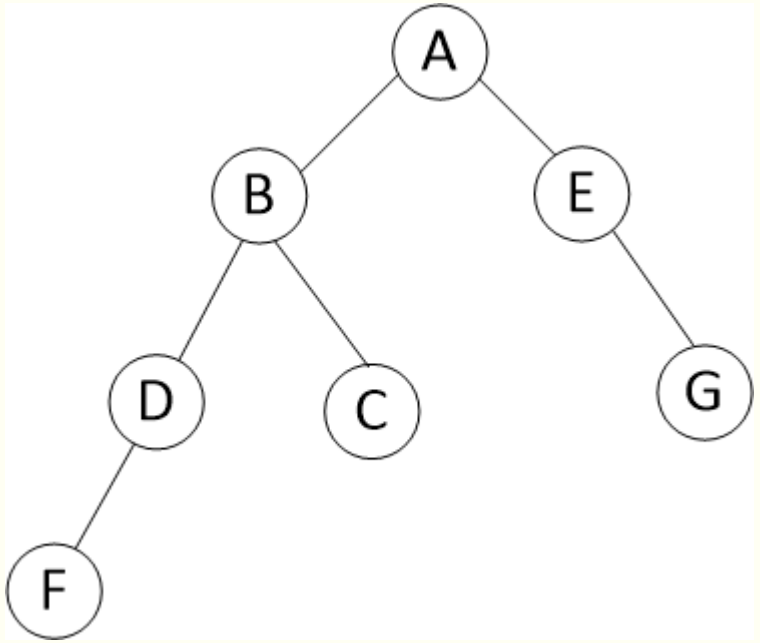


All Pairs Shorstest

	A	B	C	D	E	F	G
A	-						
B		-					
C			-				
D				-			
E					-		
F						-	
G							-

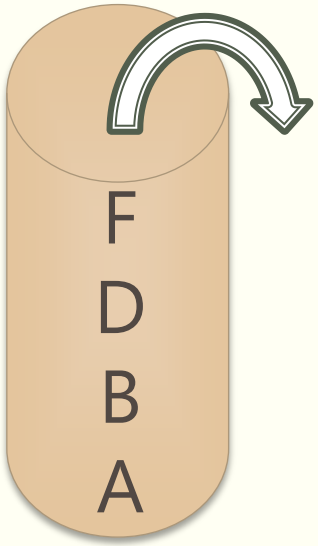
Algorithms ในการค้นหา ข้อมูล ใน Tree/Graph: DFS and BFS

- DFS → Stack



Input G=(V,E); V={A B C D E F G}

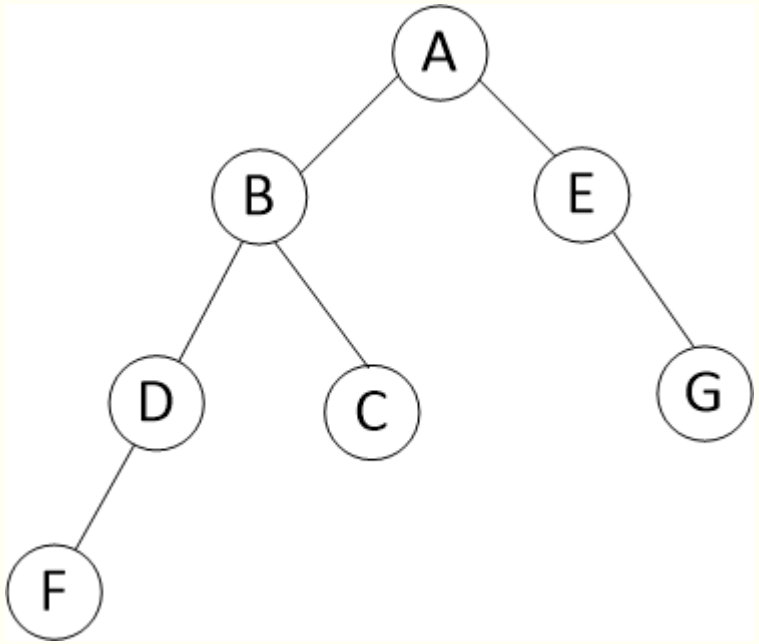
ต้องการหา node F



Output A B E D C G F

Algorithms ในการค้นหา ข้อมูล ใน Tree/Graph: DFS and BFS

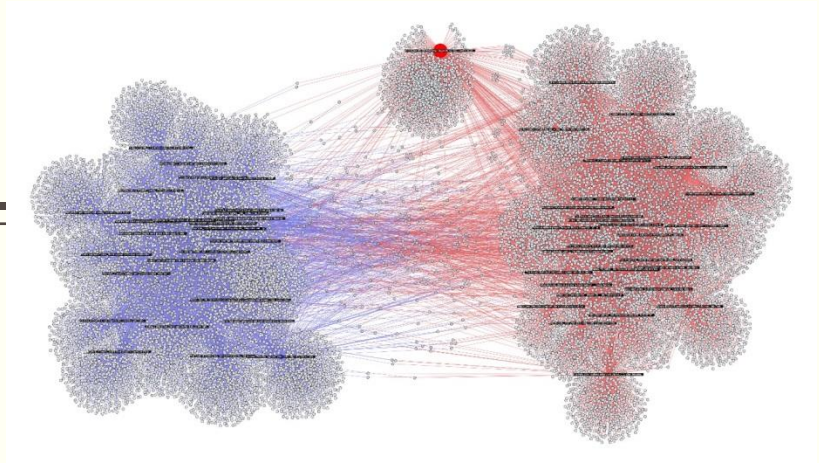
- BFS → Queue



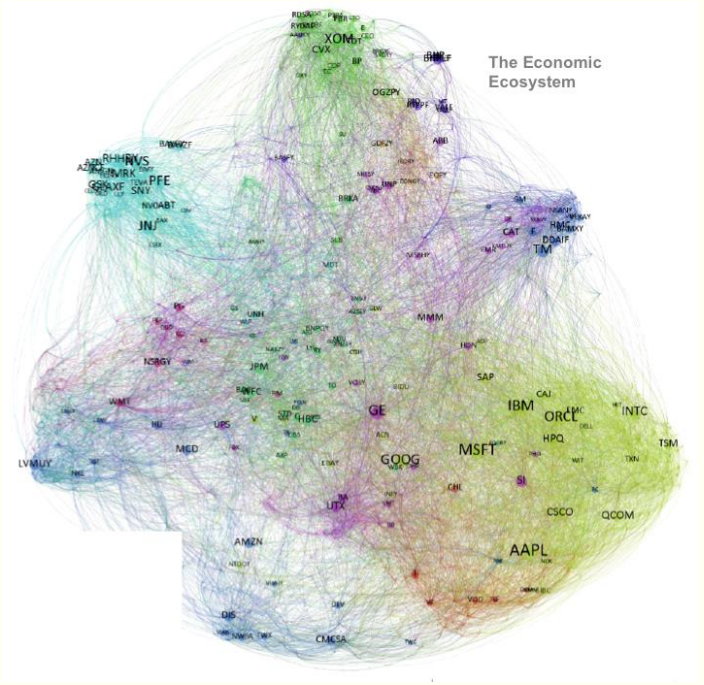
Input G=(V,E); V={A B C D E F G}
ต้องการหา node F

Output A B E D C G F

ตัวอย่าง การหา communities ใน SNs



- กลุ่มลูกค้าที่ชอบ ออกกำลังกายแบบเดียวกัน
- ผู้โจมตีระบบ
- กลุ่มผู้รับ-ส่งจม. เวียน ผ่านทางอีเมลล์
- ....



[ภาพ: internet 2018]

การประยุกต์ใช้งาน Algorithms ในชีวิตประจำวัน

- BFS/DFS/Union Find
- Dijkstra’s, shortest path
- Quick sort/ merge sort