

กราฟ (graph)

- กราฟ (graph) เป็นโครงสร้างข้อมูลแบบไม่เป็นเชิงเส้น (nonlinear data structure)
- กราฟประกอบด้วยโหนดและเอจ แต่ละโหนดสามารถมีความสัมพันธ์กับโหนดอื่นๆ ได้มากกว่าหนึ่ง โดยไม่พิจารณาถึงลำดับความสัมพันธ์ก่อนหลัง
- กราฟ เป็นโครงสร้างข้อมูลที่มีการนำไปใช้ในงานที่เกี่ยวข้องกับการแก้ปัญหาที่ค่อนข้างซับซ้อน เช่น การวางข่ายงานคอมพิวเตอร์ การวิเคราะห์เส้นทางวิกฤติ การวางแผนข่ายงาน และปัญหาเส้นทางที่สั้นที่สุด เป็นต้น

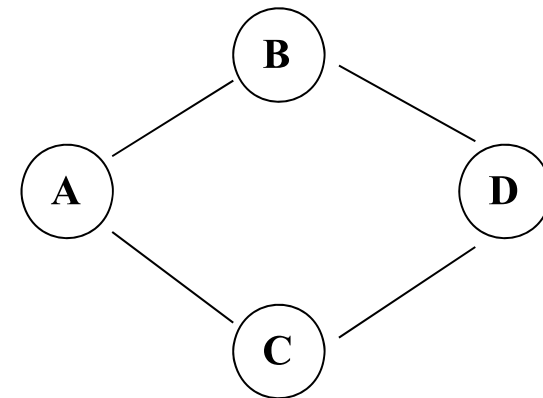
นิยาม : กราฟ (graph)

กราฟ เป็นโครงสร้างข้อมูลแบบไม่ใช้เชิงเส้นที่ประกอบด้วยกลุ่มของสิ่งสองสิ่งคือ

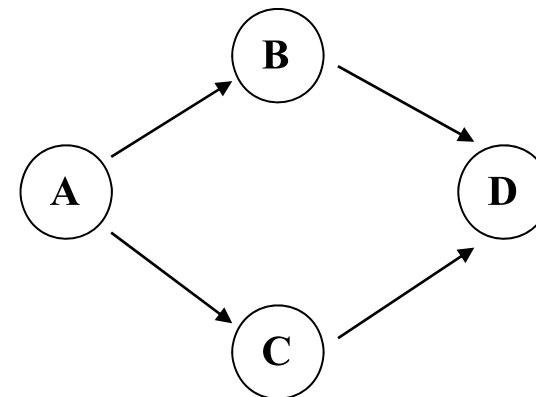
- (1) โหนด (nodes) หรือ เวอร์เทกซ์ (vertexes)
- (2) เส้นเชื่อมระหว่างโหนด เรียก เอจ (edges)

นิยาม : กราฟ (graph)

กราฟที่มีเองเชื่อมระหว่างโหนดสองโหนด ถ้าเองไม่มีลำดับ
ความสัมพันธ์จะเรียกกราฟนั้นว่า
กราฟแบบไม่มีทิศทาง (undirected graphs) และถ้ากราฟนั้นมีเองที่มีลำดับความสัมพันธ์หรือมีทิศทางกำกับด้วยเรียกกราฟนั้นว่า **กราฟแบบมีทิศทาง (directed graphs)** บางครั้งเรียกว่า **ไดกราฟ (digraph)**



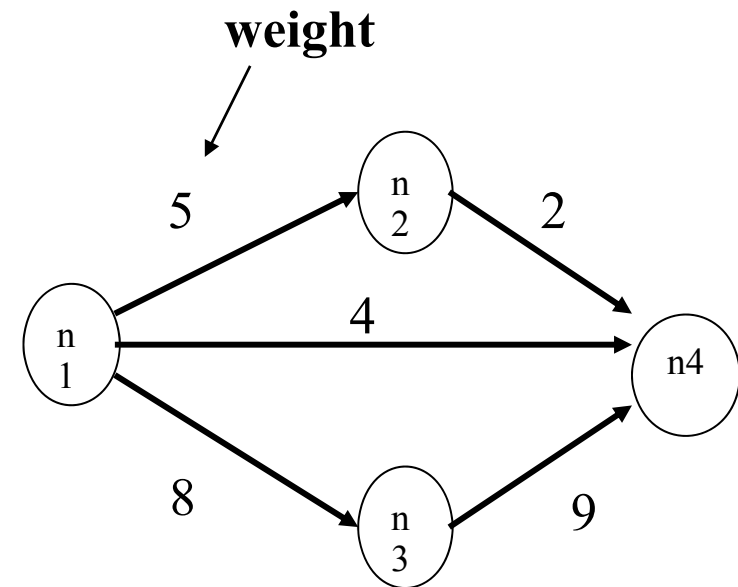
undirected graphs



directed graphs

นิยาม : กราฟ (graph)

- ค่ากำกับเส้นทาง (weight) หมายถึงค่าใช้แทนน้ำหนัก ระหว่างโหนดกับโหนด ซึ่งอาจแทนระยะทาง, ค่าความต้านทาน, ขนาดต่างๆ เป็นต้น
- เส้นทาง (path) หมายถึงทางเดินจากโหนดหนึ่งไปยังอีกโหนดหนึ่ง



Path จาก n1 ถึง n4 คือ

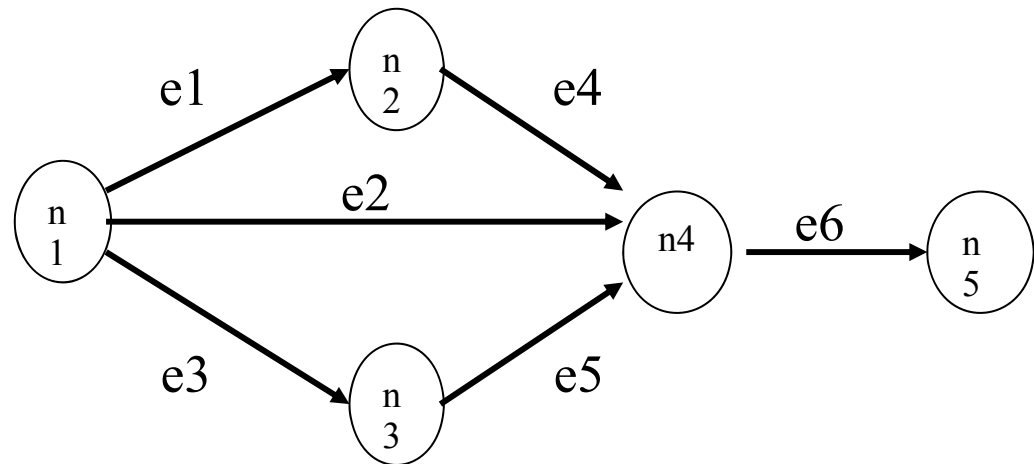
$n1 - n2 - n4$

$n1 - n4$

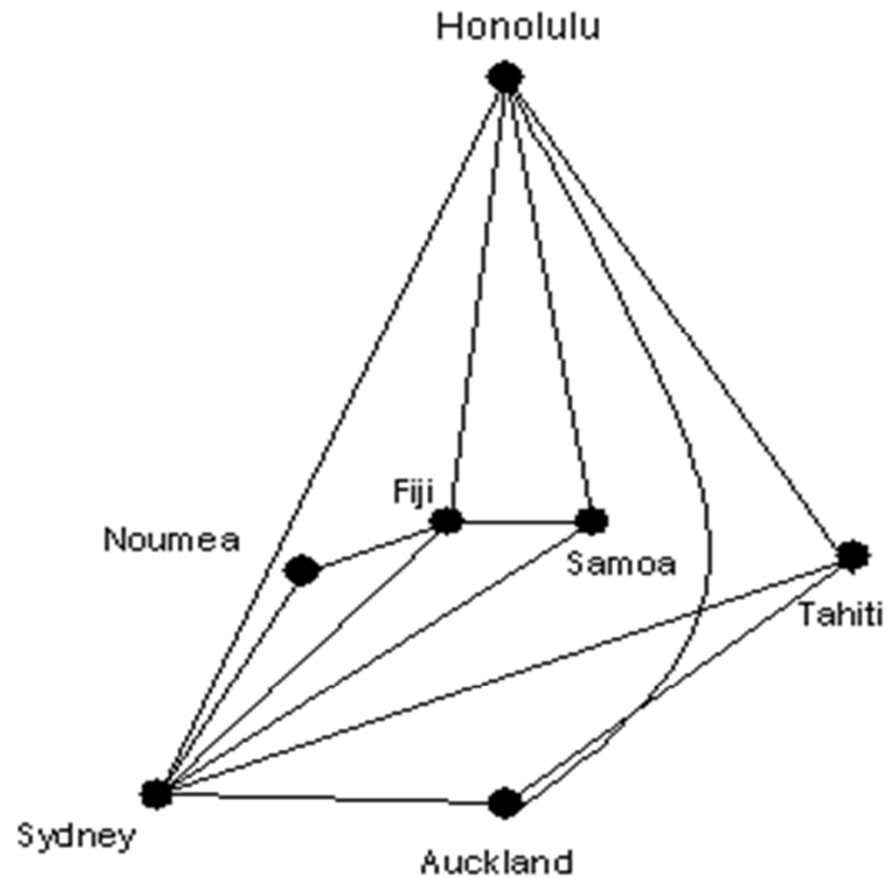
$n1 - n3 - n4$

นิยาม : กราฟ (graph)

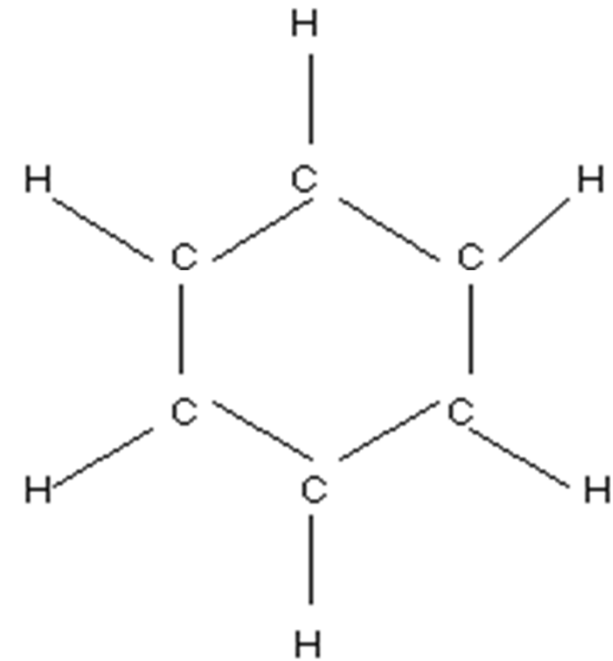
ถ้าต้องการอ้างอิงถึงเองแต่
ละเส้นสามารถเขียนชื่อ
เองกำกับไว้ก็ได้
ตัวอย่างกราฟต่อไปนี้
มีชื่อโหนดเป็น n1, n2,
n3, n4 และ n5 โดยมี
ชื่อเองเป็น e1, e2, e3,
e4, e5 และ e6



ตัวอย่างกราฟในงานต่างๆ



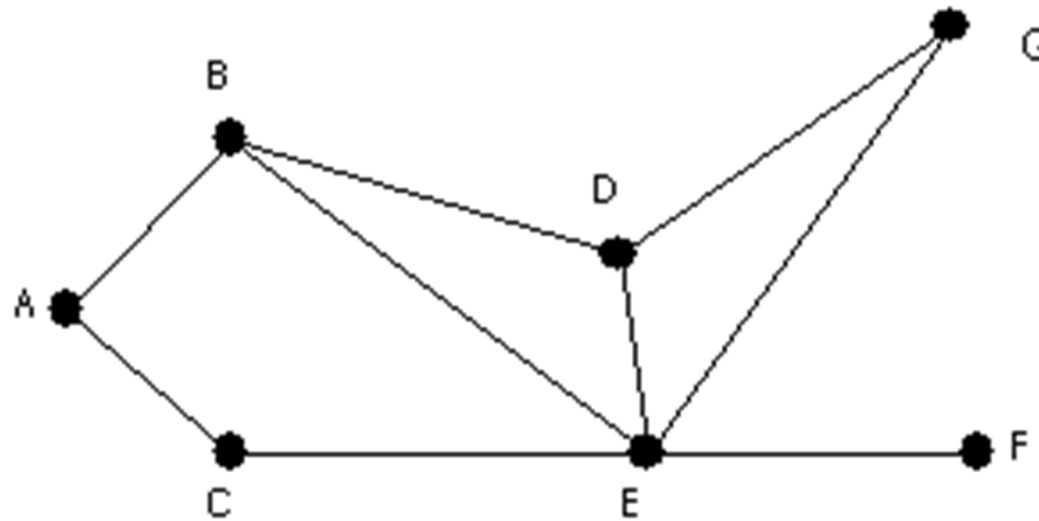
(ก) เส้นทางการบินของเซิร์สแปซิฟิก



(ข) โมเลกุลของเบนซีน

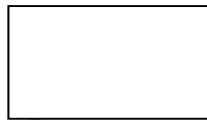
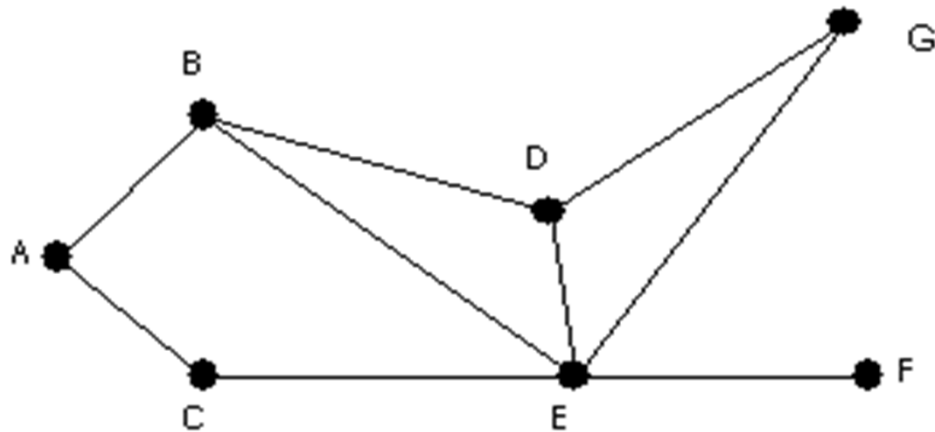
การแทนกราฟในหน่วยความจำ

วิธีที่ง่ายและตรงไปตรงมาที่สุดคือ การเก็บเองในแฉวลำดับ 2 มิติ ในรูปแสดงตัวอย่างกราฟแบบไม่มีทิศทาง และเมื่อแทนกราฟด้วยแฉวลำดับ 2 มิติได้ดังแสดงในรูป



ตัวอย่างกราฟแบบไม่มีทิศทาง

การแทนกราฟในหน่วยความจำ



ตัวอย่างกราฟแบบไม่มีทิศทาง

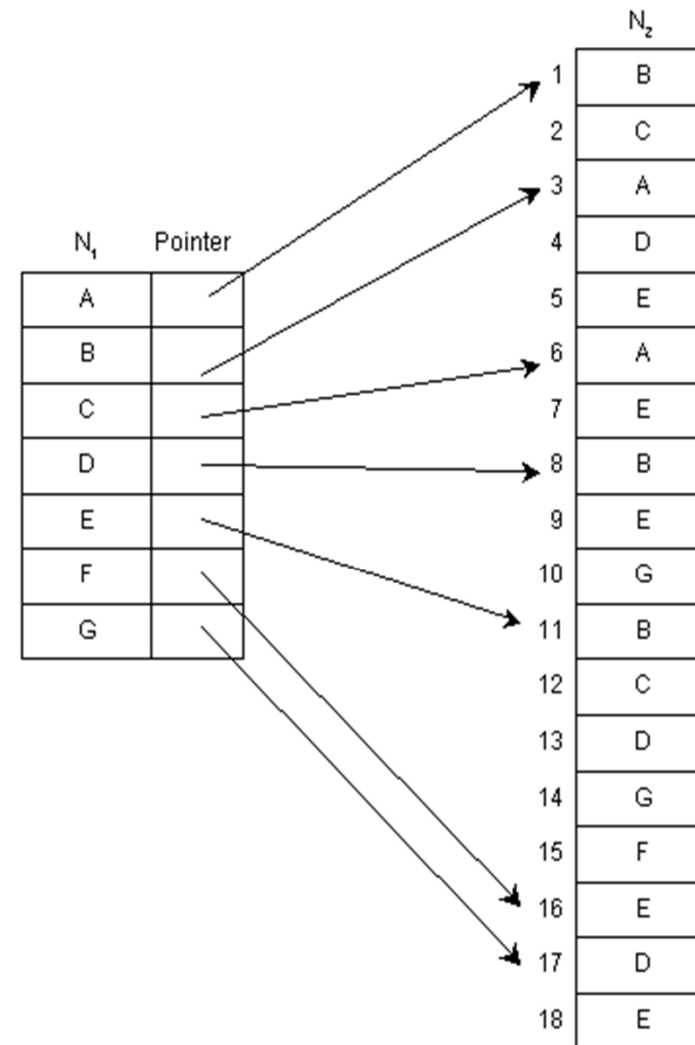
จะเห็นว่าการแทนกราฟในหน่วยความจำด้วย
วิธีเก็บเองทั้งหมดในแถวลำดับ 2 มิติค่อนข้าง
เปลืองเนื้อที่ เนื่องจากมีบางเอจที่เก็บซ้ำเดิม
โดยเฉพาะกรณีที่เป็นกราฟแบบไม่มีทิศทาง


N_1	N_2
A	B
A	C
B	A
B	D
B	E
C	A
C	E
D	B
D	E
D	G
E	B
E	C
E	D
E	G
E	F
F	E
G	D
G	E

การแทนกราฟในหน่วยความจำ

ใช้แฉวลำดับ 2 มิติเก็บโหนดและพอยน์เตอร์ชี้ไปยังตำแหน่งของโหนดต่าง ๆ ที่สัมพันธ์ด้วย และใช้แฉวลำดับ 1 มิติเก็บโหนดต่าง ๆ ที่มีความสัมพันธ์กับโหนดในแฉวลำดับ 2 มิติ

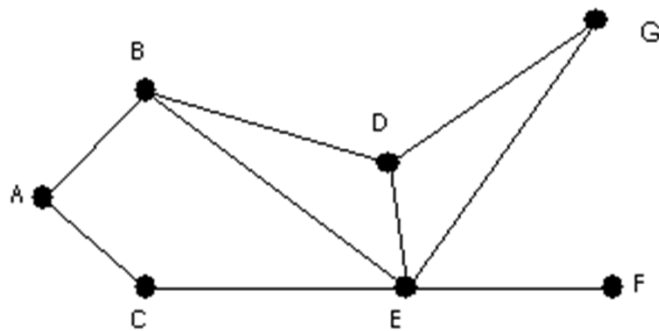
การจัดเก็บกราฟด้วยวิธีเก็บโหนดและพอยน์เตอร์นี้ยุ่งยากในการจัดการเพิ่มขึ้นเนื่องจากต้องเก็บโหนดและพอยน์เตอร์ในแฉวลำดับ 2 มิติ และต้องจัดเก็บโหนดที่สัมพันธ์ด้วยในแฉวลำดับ 1 มิติ



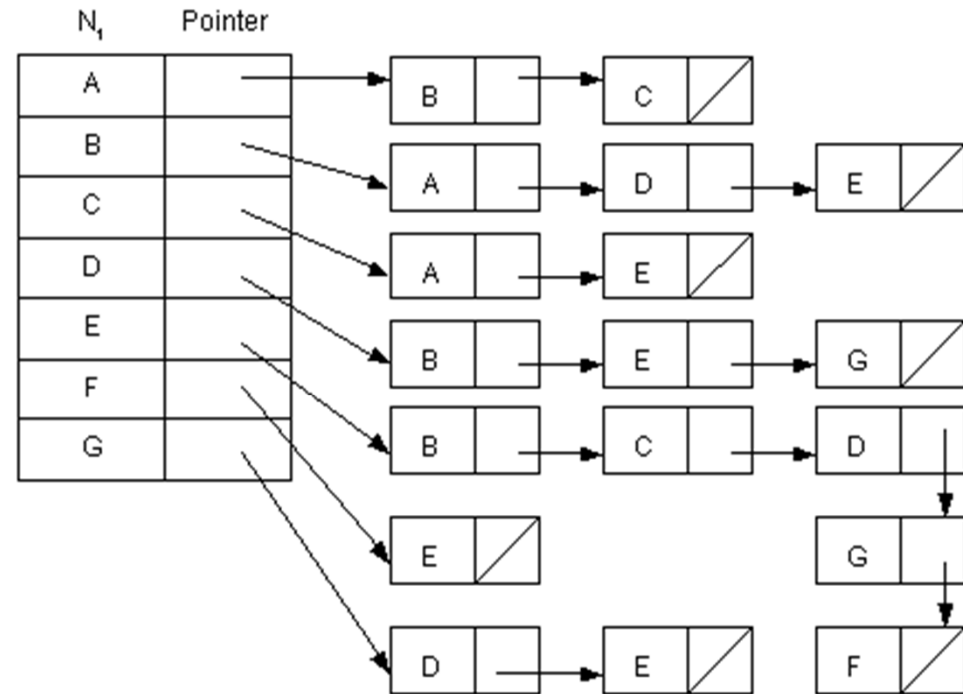
 แสดงกราฟในแฉวลำดับ 2 มิติ เก็บโหนดและพอยน์เตอร์ของกราฟ

การแทนกราฟในหน่วยความจำแอดจาเซนซีลิสต์ (adjacency list)

ซึ่งเป็นวิธีที่คล้ายวิธีจัดเก็บกราฟด้วยการเก็บโหนดและพอยน์เตอร์ แต่ต่างกันตรงที่แทนที่จะเก็บโหนดที่มีความสัมพันธ์ด้วยไว้ในแถวลำดับ 1 มิติ จะใช้ลิสต์ลิสต์แทนเพื่อความสะดวกในการเปลี่ยนแปลงแก้ไข



ตัวอย่างกราฟแบบไม่มีทิศทาง

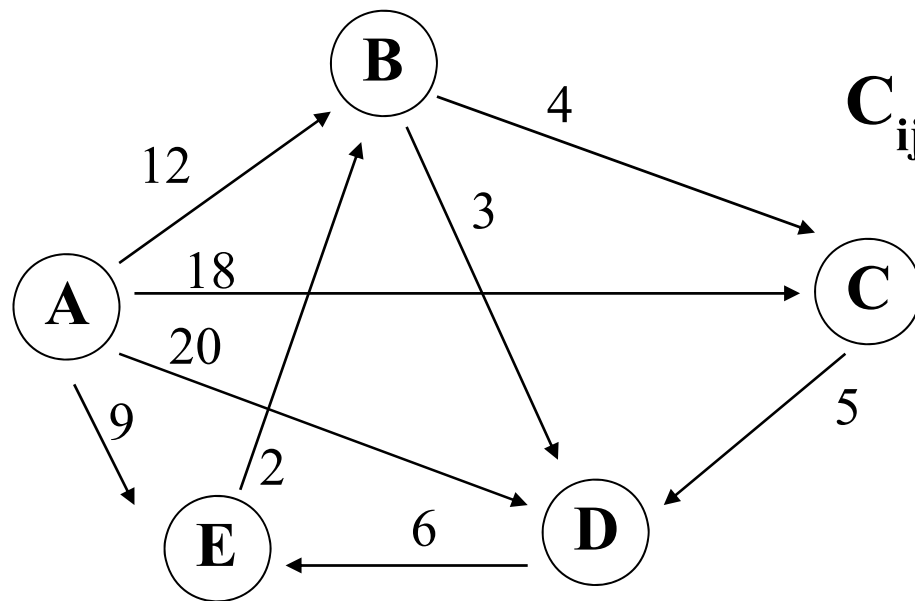


แสดงการแทนกราฟด้วยวิธีแอดจาเซนซีลิสต์

การแทนกราฟในหน่วยความจำ

อย่างไรก็ตามทั้งสามวิธีที่กล่าวมาข้างต้นไม่เหมาะกับกราฟที่มีการเปลี่ยนแปลงตลอดเวลา ควรใช้ในกราฟที่ไม่มีการเปลี่ยนแปลงตลอดอายุขัยของการใช้งาน เพราะถ้ามีการเปลี่ยนแปลงส่วนใดส่วนหนึ่งของกราฟจะกระทบกับส่วนอื่น ๆ ที่อยู่ในระดับที่ต่ำกว่าด้วยเสมอ

การแทนกราฟด้วยแอดจาเซนซีเมทริกซ์ (adjacency matrix)



$C_{ij} =$

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

เมทริกซ์ C_{ij} เป็น adjacency matrix ใช้แทนความสัมพันธ์ระหว่างโหนดบนกราฟ เช่นที่ตำแหน่ง C_{AB} หมายถึงค่า weight จากโหนด A ไปยังโหนด B ซึ่งมีค่าเท่ากับ 12

การแทนด้วยแอดจาเซนซีเมทริกซ์ (adjacency matrix)

- ค่าอื่นที่ไม่มี weight ให้เป็นค่า infinite number ใช้แทนเส้นทางที่ไม่มีอยู่จริงในกราฟ
(ทางคอมพิวเตอร์อาจแทนด้วยค่าตัวเลขที่ใหญ่มากๆ)
- โดยที่ถ้ากราฟมีทั้งหมด n โหนด
แอดจาเซนซีเมทริกซ์เป็นเมทริกซ์จัตุรัสขนาด $n \times n$

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

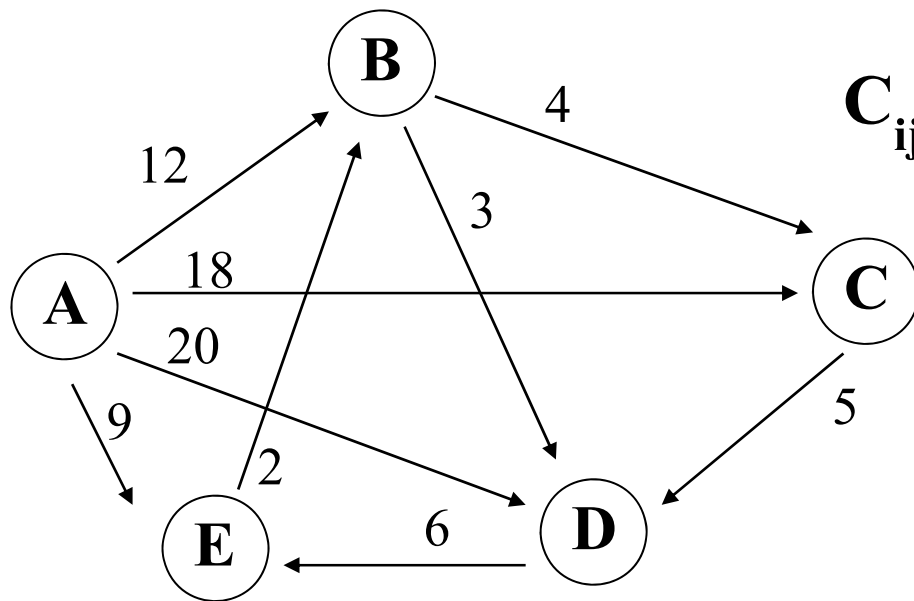
การประยุกต์ใช้กราฟกับปัญหาเส้นทางสั้นที่สุด

โครงสร้างข้อมูลแบบกราฟสามารถนำไปประยุกต์กับการใช้งานในหลายรูปแบบ เช่น ถ้าแทนโหนดเป็นจังหวัด และ weight แทนระยะทางระหว่างจังหวัด โดยกำหนดให้ edge เป็นเส้นทางระหว่างจังหวัด และต้องการทราบเส้นทางที่สั้นที่สุดระหว่างสำนักงานใหญ่กับสาขาในจังหวัดอื่นๆ เพื่อที่จะขนส่งสินค้าให้ได้ระยะทางที่สั้นที่สุด เป็นต้น

Shortest Path : Dijkstra Algorithm

- เป็นขั้นตอนการแก้ปัญหาเส้นทางที่สั้นที่สุดจากโหนดหนึ่งไปยังโหนดอื่นๆ บนกราฟ (Single Source Shortest Path)
- ถูกค้นพบโดยนักคณิตศาสตร์คอมพิวเตอร์ ที่ชื่อว่า **Dijkstra**
- รูปแบบของปัญหา เป็นการหาระยะทางที่สั้นที่สุดจาก source node ไปยัง โหนดต่างๆ (single source shortest path)

Shortest Path : Dijkstra Algorithm



$C_{ij} =$

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

แทนความสัมพันธ์ของ กราฟลงใน Adjacency Matrix

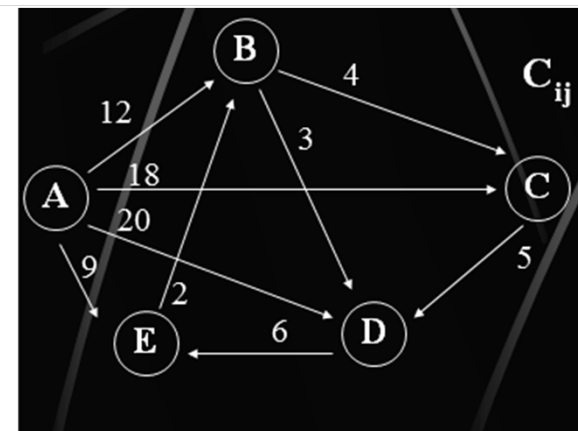
Shortest Path : Dijkstra Algorithm

Step 0

NO	S	W	d(B)	d(C)	d(D)	d(E)
O	A	-	12	18	20	9

ให้ S เป็น source node
w คือโหนดที่มี weight
ที่มีค่าน้อยที่สุดของ
โหนดที่ไม่อยู่ใน S
 $d(C_{ij})$ คือระยะทาง
จากโหนด i ไปยัง j

ขั้นตอน : นำค่าในแถว A มากำหนด
ให้เป็นค่าเริ่มต้นในตาราง



Step 1

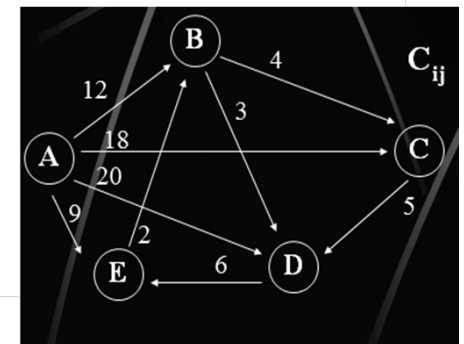
NO	S	W	d(B)	d(C)	d(D)	d(E)
O	A	-	12	18	20	9
1	A,E	E	11	18	20	9

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

ขั้นตอน : 1.1 เลือกค่า min จาก d(B), d(C), d(D), d(E) คือโหนด E นำโหนด E ไปไว้ใน S ร่วมกับ A

1.2 ปรับปรุงเส้นทางจากโหนด A ไปยังโหนดอื่นๆ ที่ไม่อยู่ใน S โดยเปรียบเทียบเส้นทางเดิมในขั้นตอนที่ 0 กับเส้นทางใหม่ จาก A ไปยังโหนดนั้นโดยผ่านโหนด E

ค่า 11 พิจารณาจาก $A \xrightarrow{12} B = 12$ และ $A \xrightarrow{9} E \xrightarrow{2} B = 11$



Step 2

NO	S	W	d(B)	d(C)	d(D)	d(E)
O	A	-	12	18	20	9
1	A,E	E	11	18	20	9
2	A,E,B	B	11	16	15	9

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

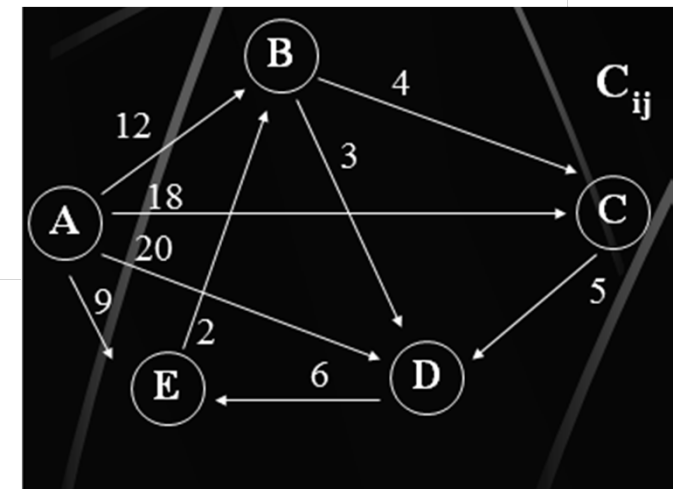
ขั้นตอน : 2 เลือก B เข้าไปใน w

ค่า 16 พิจารณาจาก $A \xrightarrow{18} C = 18$ และ $A \xrightarrow{12} B \xrightarrow{4} C = 16$

เลือกค่า minimize = 16

ค่า 15 พิจารณาจาก $A \rightarrow D = 20$ และ $A \xrightarrow{12} B \xrightarrow{3} D = 15$

เลือกค่า minimize = 15



Step 3

NO	S	W	d(B)	d(C)	d(D)	d(E)
0	A	-	12	18	20	9
1	A,E	E	11	18	20	9
2	A,E,B	B	11	16	15	9
3	A,E,B,D	D	11	16	15	9

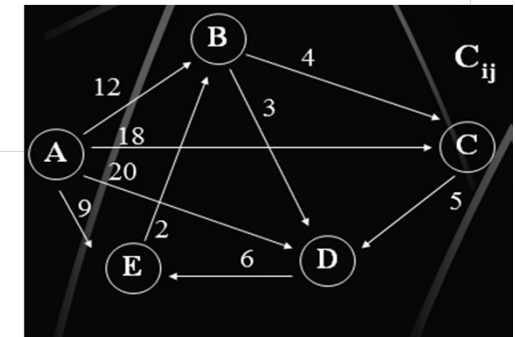
	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

ขั้นตอน : 3 เลือก D เข้าไปใน w

พิจารณาปรับปรุงเส้นทางจาก $A \rightarrow C$ โดย

$$A \xrightarrow{12} B \xrightarrow{4} C = 16 \text{ และ } A \xrightarrow{15} D \xrightarrow{\infty} C = \infty$$

เลือกค่า minimize = 16



Step 4

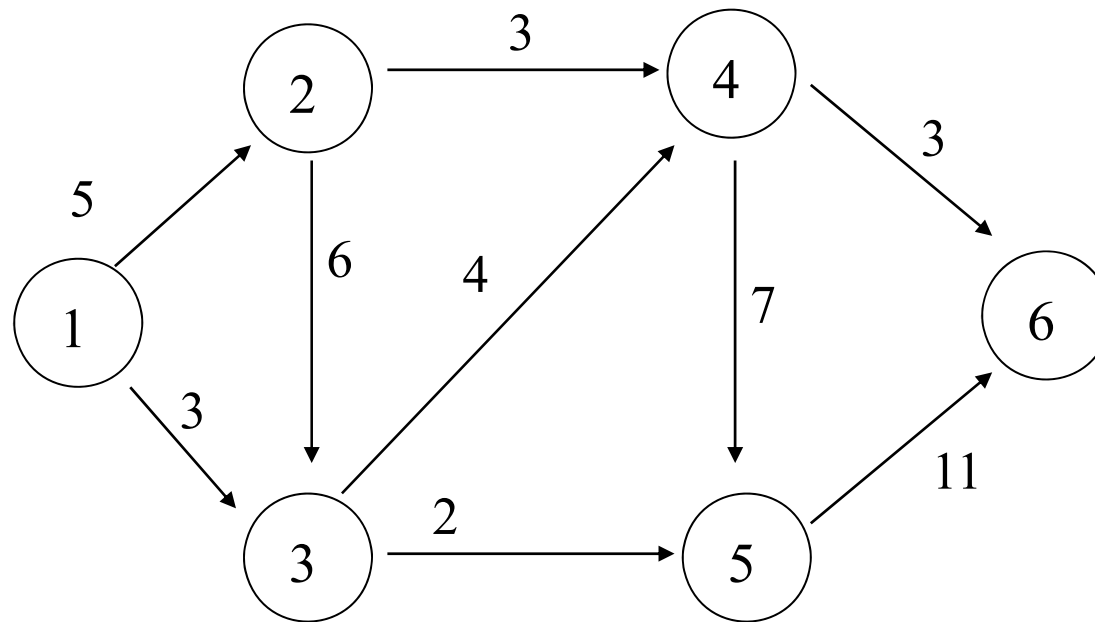
NO	S	W	d(B)	d(C)	d(D)	d(E)
O	A	-	12	18	20	9
1	A,E	E	11	18	20	9
2	A,E,B	B	11	16	15	9
3	A,E,B,D	D	11	16	15	9
4	A,E,B,D,C	C	11	16	15	9

	A	B	C	D	E
A	∞	12	18	20	9
B	∞	∞	4	3	∞
C	∞	∞	∞	5	∞
D	∞	∞	∞	∞	6
E	∞	2	∞	∞	∞

ขั้นตอน :4 เลือกโหนดสุดท้ายคือ C เข้าไปไว้ใน w
ถือว่าสิ้นสุดการทำงานเนื่องจากทุกโหนดเข้าไปอยู่ใน S แล้ว

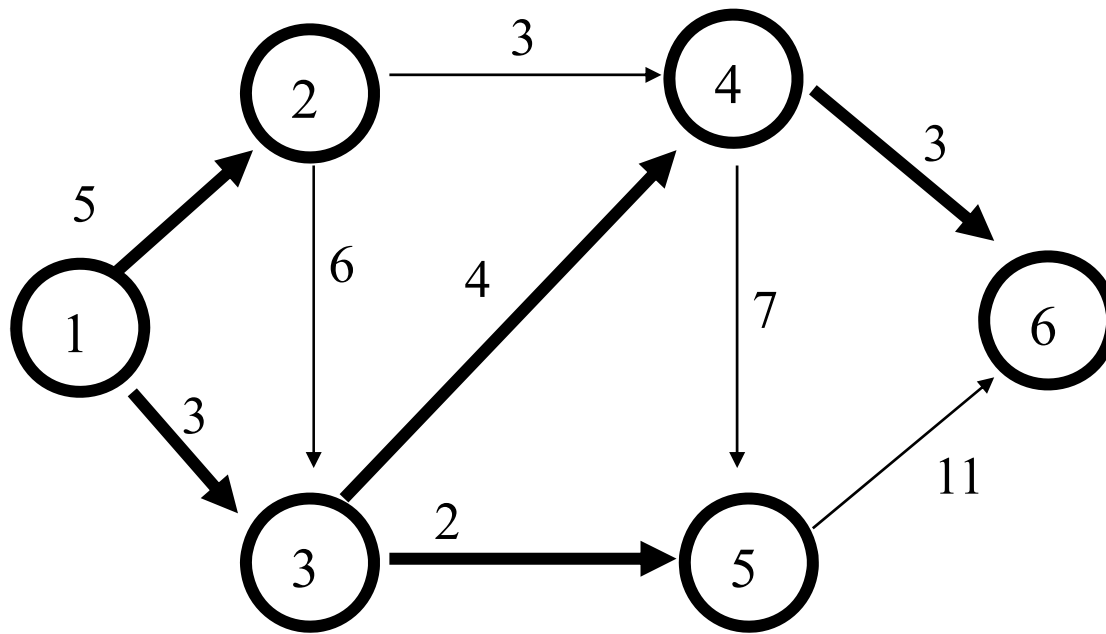
คำตอบ : ระยะทางที่สั้นที่สุดจาก A ไป B มีค่า = 11 มี path คือ $A \xrightarrow{9} E \xrightarrow{2} B$
 ระยะทางที่สั้นที่สุดจาก A ไป C มีค่า = 16 มี path คือ $A \xrightarrow{9} E \xrightarrow{2} B \xrightarrow{4} C$
 ระยะทางที่สั้นที่สุดจาก A ไป D มีค่า = 15 มี path คือ $A \xrightarrow{9} E \xrightarrow{2} B \xrightarrow{3} D$
 ระยะทางที่สั้นที่สุดจาก A ไป E มีค่า = 9 มี path คือ $A \xrightarrow{9} E$

Example :



จงแสดงการหาเส้นทางสั้นที่สุดจาก เมือง 1 ไปในแต่ละเมือง โดยมีจุดเริ่มต้นการเดินทางทุกครั้งอยู่ที่เมือง 1 ด้วยการใช้ขั้นตอนวิธีของ Dijkstra (Single- Source Shortest Path)

Example :



จงแสดงการหาเส้นทางสั้นที่สุดจาก เมือง 1 ไปในแต่ละเมือง โดยมีจุดเริ่มต้นการเดินทางทุกครั้งอยู่ที่เมือง 1 ด้วยการใช้ขั้นตอนวิธีของ Dijkstra (Single- Source Shortest Path)

Longest Path : CPM

- เป็นขั้นตอนการหาเส้นทางที่ยาวที่สุดจากโหนดเริ่มต้นไปยังโหนดสิ้นสุด บนกราฟ
- CPM (Critical Path Method) เอ็ม บี วอลเกอร์ แห่ง ดูปองต์ USA.

PERT (Program Evaluation and Review Technique) กองทัพเรือ USA. Polaris Project

PERT/CPM Chart

- PERT Chart : Project Evaluation and Review Technique Chart
- CPM Chart : Critical Path Method

รูปแบบของการเขียนข่ายงาน

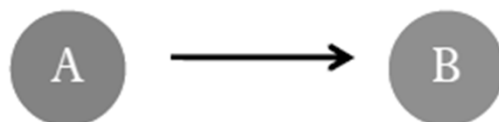
- รูปแบบของการเขียนข่ายงาน PERT/CPM แบ่งเป็น

1. ข่ายงานแบบผังลูกศร(arrow diagram) หรืออาจเรียกว่า Activity-On-Arrow(A-O-A) เนื่องจากการเขียนข่ายงานรูปแบบนี้จะใช้ลูกศรแทนกิจกรรม และสัญลักษณ์วงกลมแทนจุดงานที่เป็นจุดเริ่มต้นหรือจุดสิ้นสุดของกิจกรรม ทิศทางของหัวลูกศรแสดงความก้าวหน้าของกิจกรรมโดยความยาวของลูกศร ไม่มีความสัมพันธ์กับเวลาหรือทรัพยากรของกิจกรรม



รูปแบบของการเขียนผังงาน

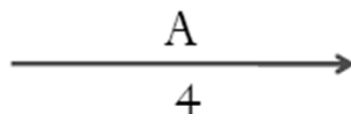
2. ข่ายงานแบบผังวงกลม(node diagram) หรืออาจเรียกว่า Activity-On-Node(A-O-N) เนื่องจากการเขียนข่ายงานรูปแบบนี้จะใช้สัญลักษณ์วงกลมแทนกิจกรรมและสัญลักษณ์ลูกศรแทนความสัมพันธ์ของกิจกรรม



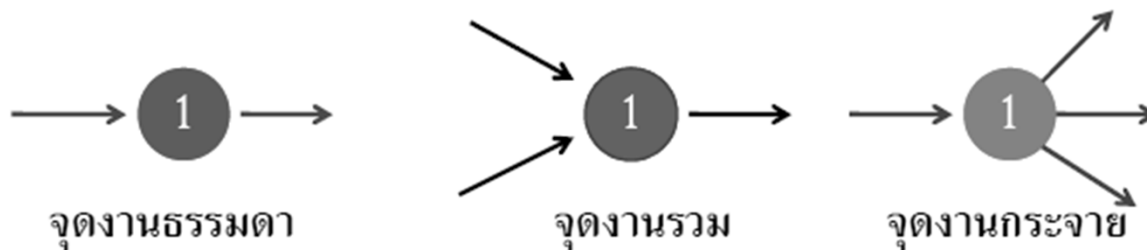
ในกรณีนี้จะใช้วิธี ข่ายงานแบบผังลูกศร

องค์ประกอบของข่ายงาน

- 1.กิจกรรม(activity) แสดงถึงงานที่ต้องกระทำในโครงการ ซึ่งต้องใช้เวลาหรือทรัพยากรจำนวนหนึ่ง กิจกรรมจะแทนด้วยลูกศร ซึ่งจะระบุชื่อของกิจกรรมไว้ด้านบนและเวลาหรือทรัพยากรของกิจกรรมจะเขียนไว้ด้านล่างของกิจกรรม



- 2.จุดงาน(node) แสดงถึงจุดเริ่มต้นหรือจุดสิ้นสุดของกิจกรรม โดยแทนด้วยสัญลักษณ์วงกลมที่ภายในระบุตัวเลขที่แสดงถึงจุดงาน



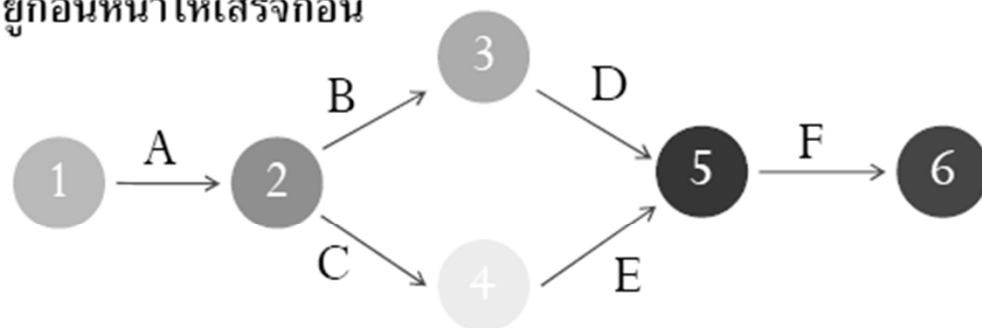
องค์ประกอบของข่ายงาน

- ความสัมพันธ์ก่อนหลังของกิจกรรม (precedence relationship) คือ ลำดับก่อนหลังของกิจกรรมที่แสดงให้เห็นว่ากิจกรรมใดต้องทำก่อนหรือหลังกิจกรรมใด

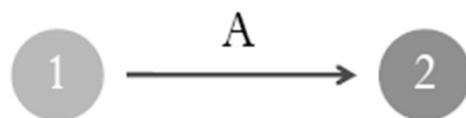


กฎการเขียนผังงาน

- กฎข้อที่ 1 ก่อนที่จะเริ่มต้นทำกิจกรรมใด ๆ จะต้องทำกิจกรรมทั้งหมดที่อยู่ก่อนหน้าให้เสร็จก่อน

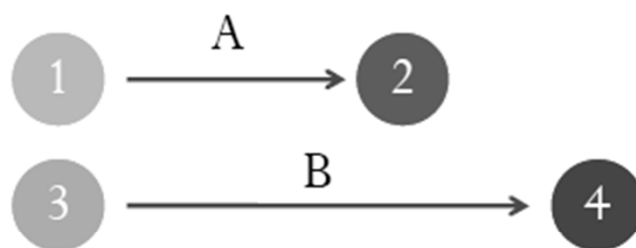


- กฎข้อ 2 ลูกศรแต่ละเส้นแทนกิจกรรมได้เพียง 1 กิจกรรม หางลูกศรจะหมายถึงจุดเริ่มต้นของกิจกรรม หัวลูกศรจะหมายถึงจุดสิ้นสุดของกิจกรรม โดยลูกศรจะเริ่มจากซ้ายไปขวา

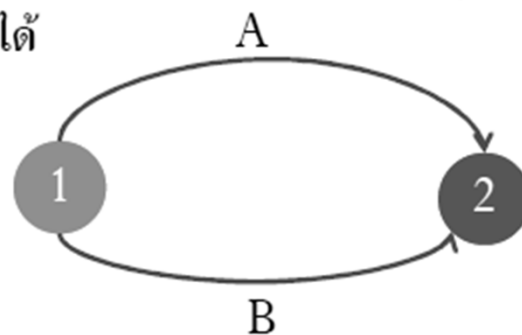


กฎการเขียนข่ายงาน

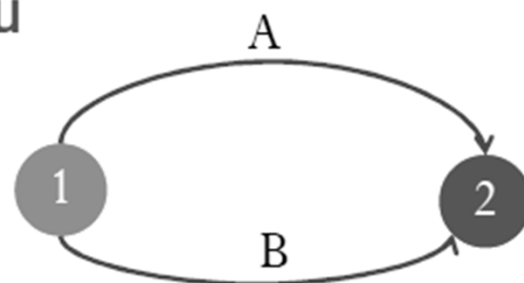
- กฎข้อที่ 3 ความยาวของลูกศรไม่มีความสัมพันธ์ใด ๆ กับระยะเวลาของกิจกรรม



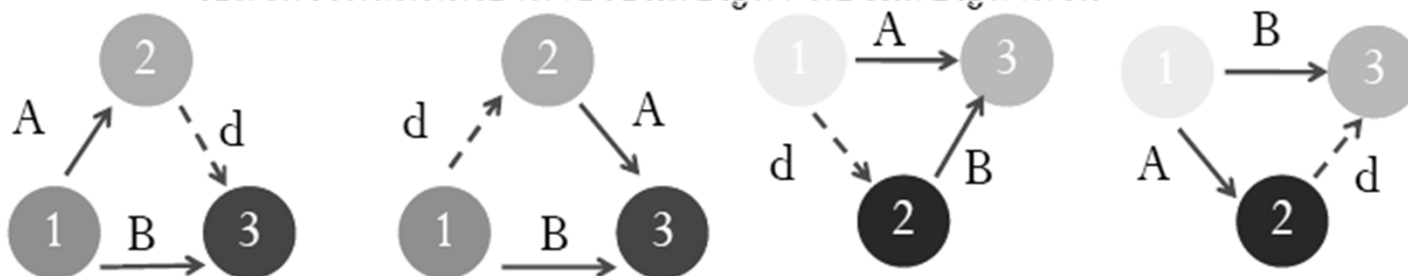
- กฎข้อที่ 4 กิจกรรมสองกิจกรรมใด ๆ จะเริ่มต้นที่จุดงานเดียวกันแต่สิ้นสุดที่จุดงานเดียวกันไม่ได้



กฎการเขียนข่ายงาน



- จะเห็นว่ากิจกรรม **A** และ **B** มีจุดเริ่มต้นและจุดสิ้นสุดเดียวกัน ซึ่งจะต้องใช้กิจกรรมสมมติเข้ามาช่วยแก้ปัญหา ต้องแก้ปัญหาดังนี้



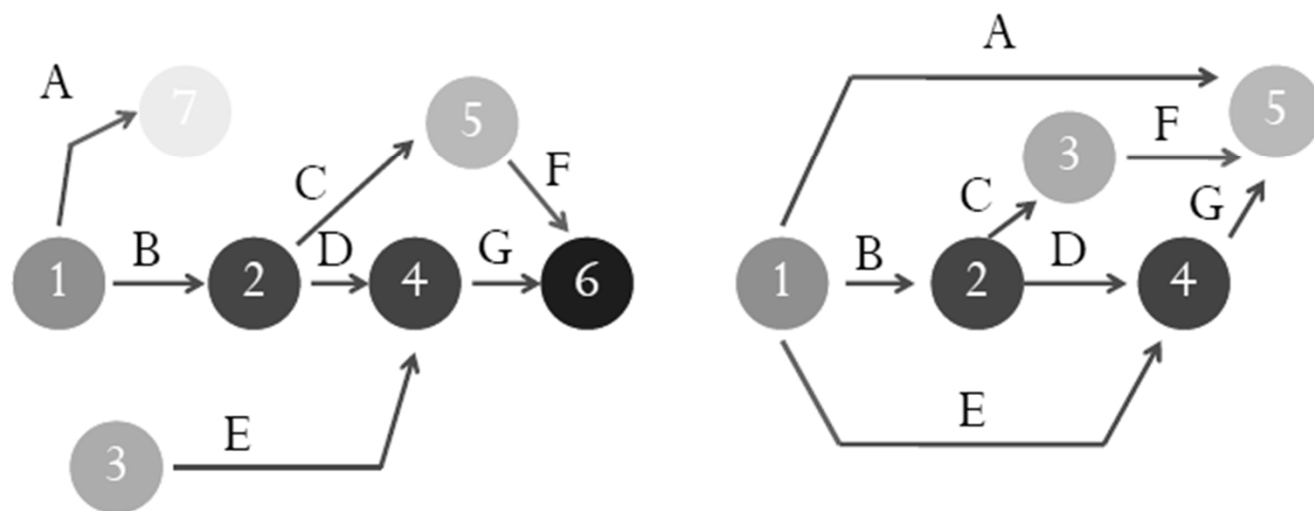
กฎการเขียนผังงาน

- กฎข้อที่ 5 หมายเลขในวงกลมจะต้องเพิ่มจากตัวเลขที่ทางลูกศรไปสู่ตัวเลขที่มากกว่าทางหัวลูกศรเสมอ



กฎการเขียนข่ายงาน

- กฎข้อที่ 6 ข่ายงานจะต้องต่อเนื่องกันตลอดจากจุดงานเริ่มต้นจนถึงจุดงานสุดท้าย โดยจุดงานเริ่มต้นและจุดงานสุดท้ายต้องมีเพียงจุดงานเดียว



โจทย์ทดสอบ การเขียนข่ายงาน

- โครงการที่ 1

กิจกรรม	กิจกรรมที่ต้องทำ เสร็จ	ระยะเวลาดำเนิน งาน(วัน)
A	-	4
B	-	1
C	A	6
D	B	5
E	C,D	6

- โครงการที่ 2

กิจกรรม	กิจกรรมที่ต้องทำ เสร็จ	ระยะเวลาดำเนิน งาน(วัน)
A	-	2
B	A	1
C	A	1
D	B,C	5

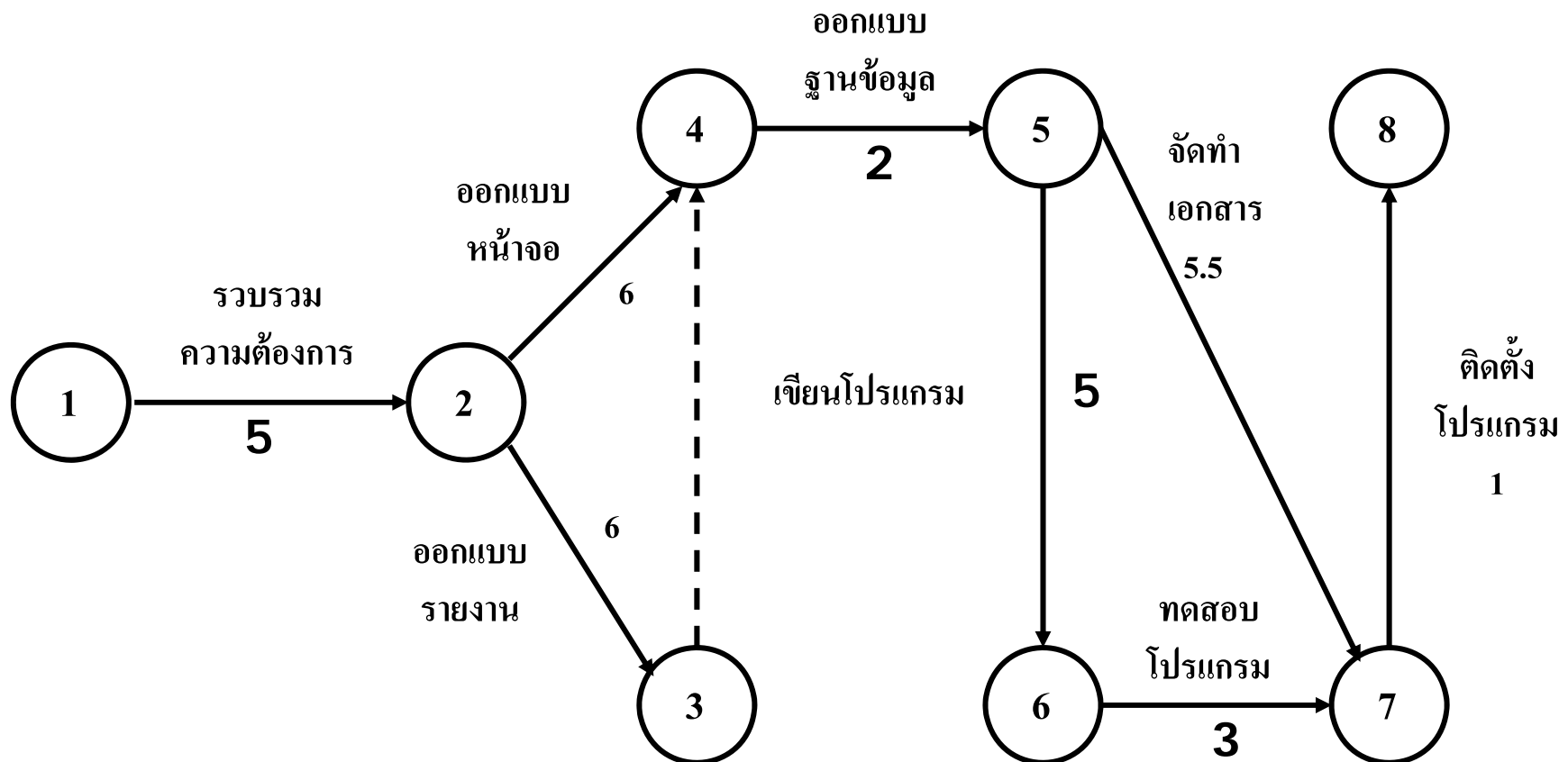
PERT Chart

- เป็นแผนภาพแสดงกิจกรรมของโครงการที่เชื่อมโยงกันในลักษณะของเครือข่าย (ข่ายงาน) ทำให้ทราบว่า จะต้องดำเนินกิจกรรมใดให้เสร็จสิ้นก่อนกิจกรรมถัดไป
- โดยแต่ละกิจกรรมจะแทนด้วยเส้นลูกศร และเชื่อมโยงกันด้วยวงกลม (เรียกว่า โหนด) เพื่อบอกให้ทราบถึงจุดเริ่มต้นและจุดสิ้นสุดของแต่ละกิจกรรม

PERT Chart

- เหมาะสำหรับโครงการใหม่ที่ไม่เคยเกิดขึ้นเลย
- การกำหนดเวลากิจกรรมของ **PERT Chart** จึงเป็นการกำหนดในรูปของความน่าจะเป็น (Probabilistic)

PERT Chart



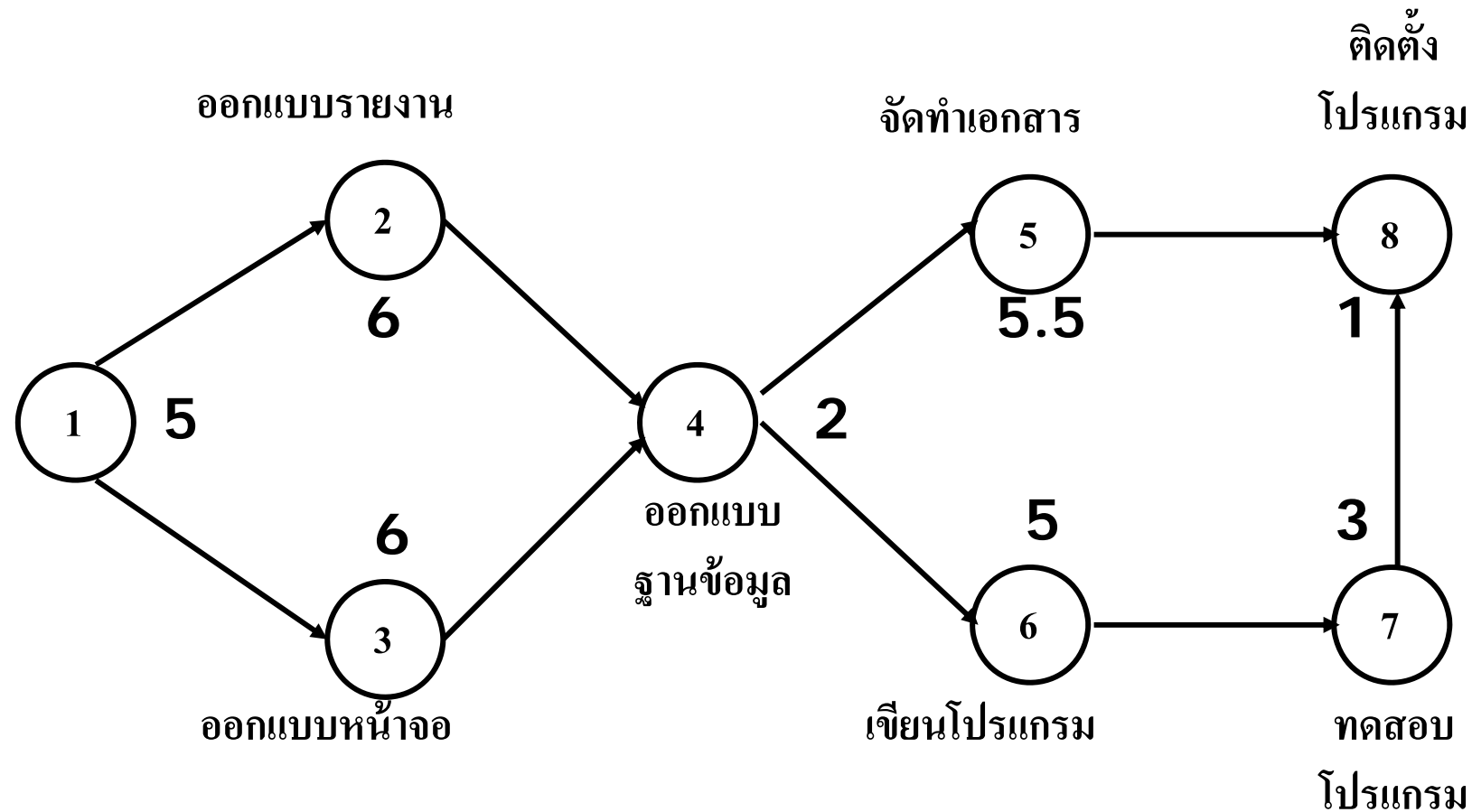
CPM Chart

- เป็นแผนภาพแสดงกิจกรรมของโครงการที่เชื่อมโยงกันในลักษณะ
เครือข่าย (ข่ายงาน) ทำให้ทราบว่าต้องดำเนินกิจกรรมใดให้เสร็จสิ้น
ก่อนกิจกรรมถัดไป เช่นเดียวกับ **PERT Chart**

CPM Chart

- เหมาะสำหรับโครงการที่เคยเกิดขึ้นแล้วในอดีต ทำให้มีข้อมูลเพื่อกำหนดระยะเวลาของกิจกรรมได้เป็นที่แน่นอน
(Deterministic)

CPM Chart



Critical Path : เส้นทางวิกฤต

- หมายถึง เส้นทางที่ใช้เวลาในการดำเนินกิจกรรมรวมของโครงการนานที่สุด และกิจกรรมที่อยู่บนเส้นทางวิกฤตจะเรียกว่า กิจกรรมวิกฤต **Critical Activity**

การกำหนดระยะเวลาด้วย Statistic

- แยกแยะกิจกรรมของโครงการ
- กำหนดกิจกรรมที่ต้องดำเนินให้เสร็จสิ้นก่อนดำเนินกิจกรรมต่อไป
- กำหนดระยะเวลาทั้งหมด 3 ค่า
 - เวลาทำกิจกรรมให้เสร็จสิ้นเร็วสุด **Optimistic**
 - เวลาทำกิจกรรมให้เสร็จสิ้นช้าสุด **Pessimistic**
 - เวลาทำกิจกรรมให้เสร็จสิ้นที่เป็นไปได้มากที่สุด **Realistic**

การกำหนดระยะเวลาด้วย Statistic

- นำค่าทั้ง 3 มาคำนวณหาค่าใช้จริงเพียงค่าเดียว เรียกว่า ค่าระยะเวลาคาดหวัง **Expected Time** โดยใช้สูตร

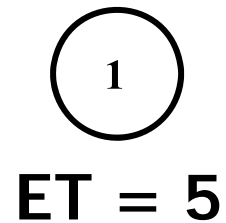
$$ET = \frac{o + 4r + p}{6}$$

ขั้นตอนที่ 3 : กำหนดค่าระยะเวลาคาดหวัง

กิจกรรม	กิจกรรม ก่อนหน้า	กำหนดระยะเวลา (สัปดาห์)			ค่าระยะเวลา คาดหวัง
		o	r	p	ET
T ₁	-	1	5	9	5
T ₂	1	5	6	7	6
T ₃	1	3	6	9	6
T ₄	2, 3	1	2	3	2
T ₅	4	3	6	7	5.5
T ₆	4	4	5	6	5
T ₇	6	1	3	5	3
T ₈	5, 7	1	1	1	1

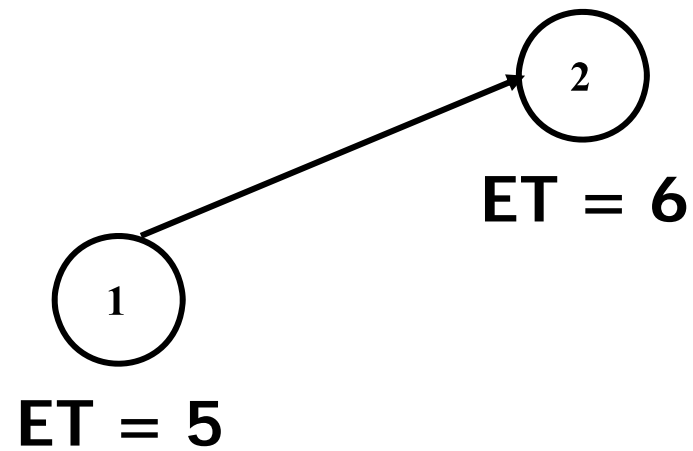
ขั้นตอนที่ 4 : วาดแผนภาพ PERT/CPM

4.1 วาดเริ่มจากโหนดกิจกรรมที่ 1



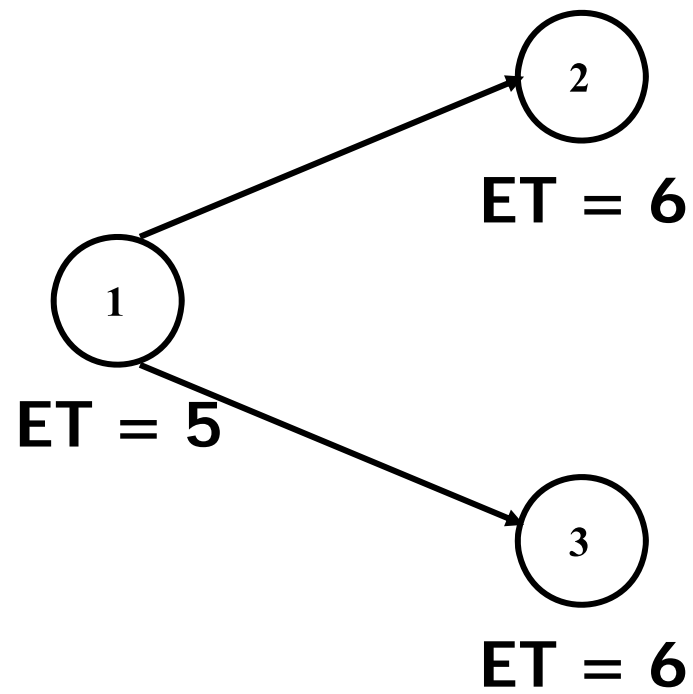
ขั้นตอนที่ 4

4.2 วาดโหนดกิจกรรมที่ 2 ซึ่งมีกิจกรรมที่ 1 ก่อนหน้า



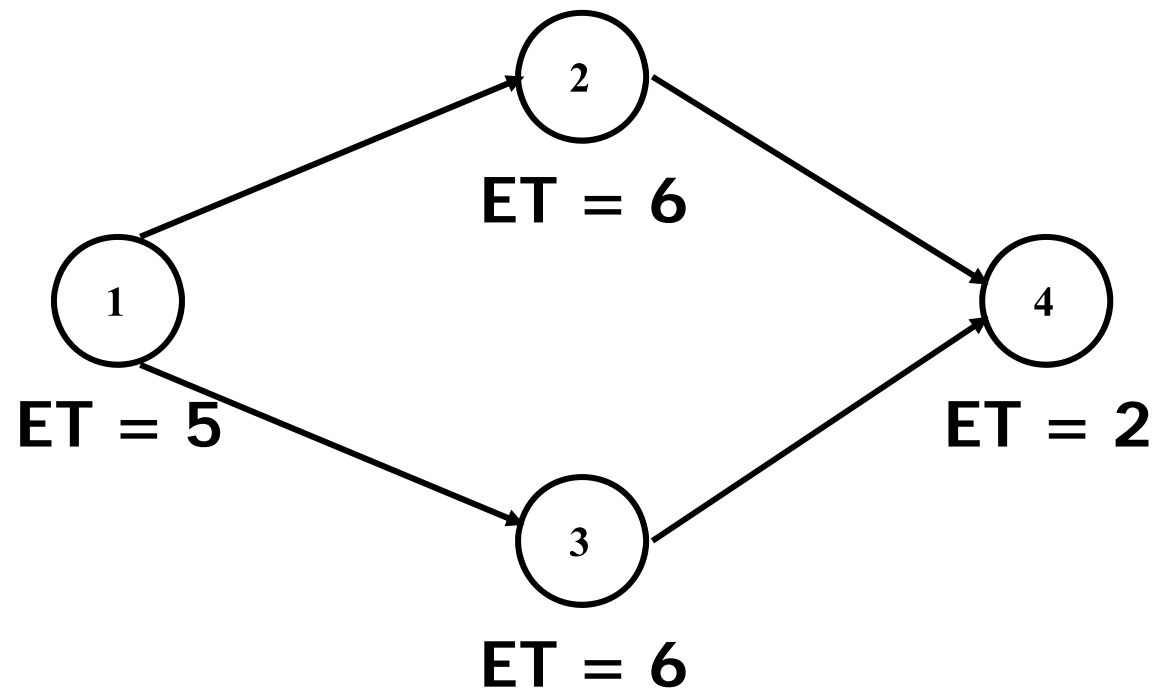
ขั้นตอนที่ 4

4.3 วาดโหนดกิจกรรมที่ 3 ซึ่งมีกิจกรรมที่ 1 ก่อนหน้า



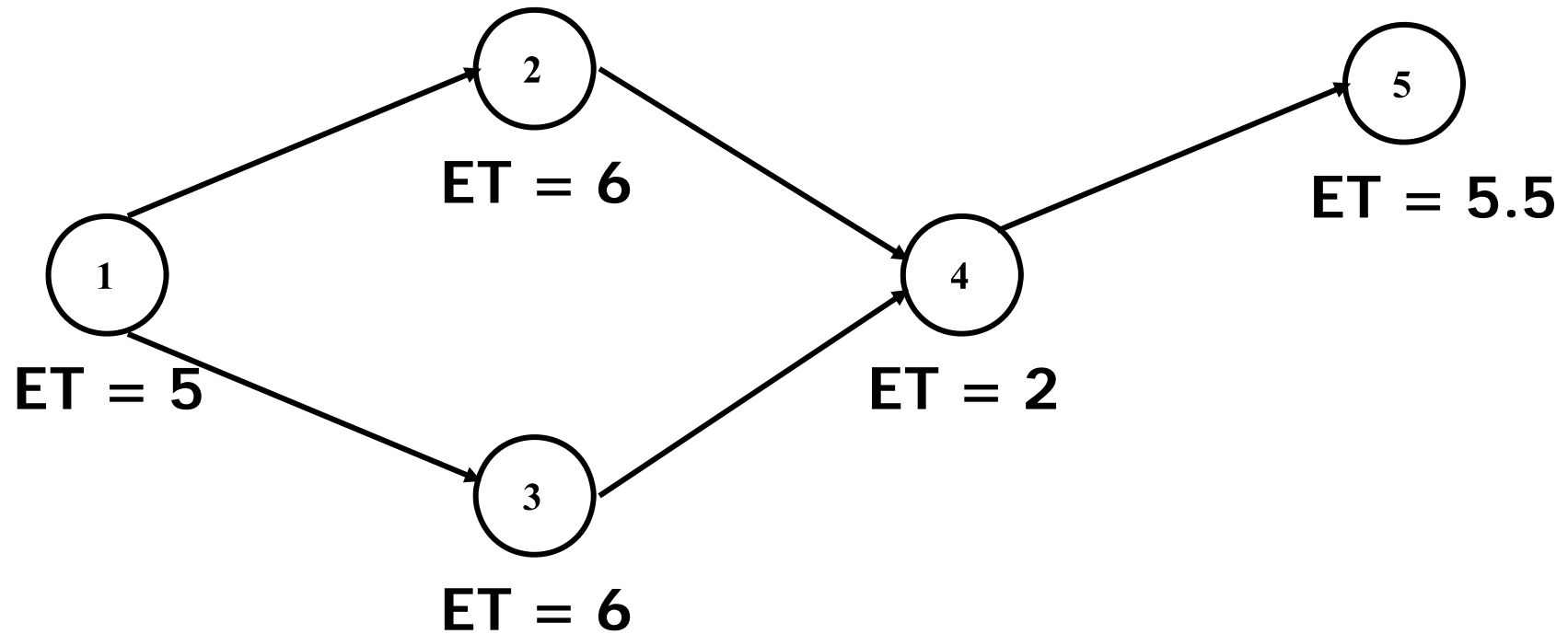
ขั้นตอนที่ 4

4.4 วาดโหนดกิจกรรมที่ 4 ซึ่งมีกิจกรรมที่ 2,3 ก่อนหน้า



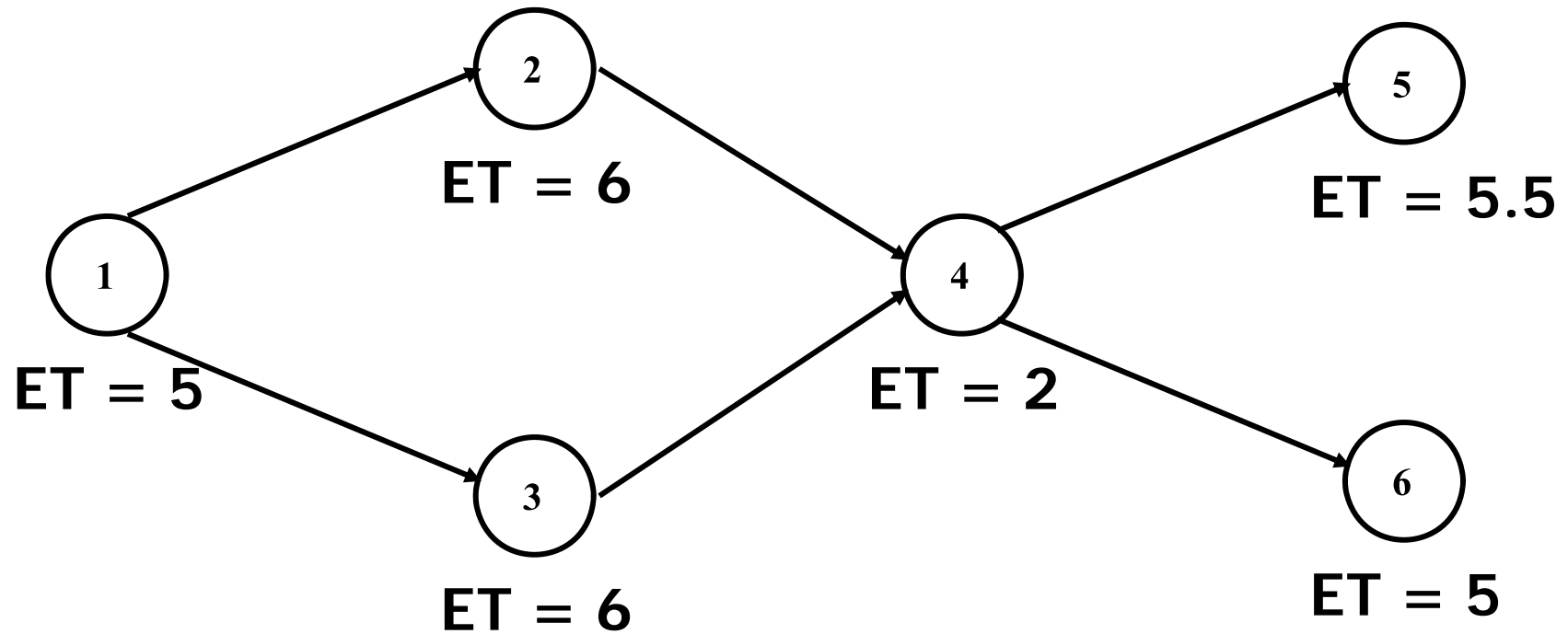
ขั้นตอนที่ 4

4.5 วาดโหนดกิจกรรมที่ 5 ซึ่งมีกิจกรรมที่ 4 ก่อนหน้า



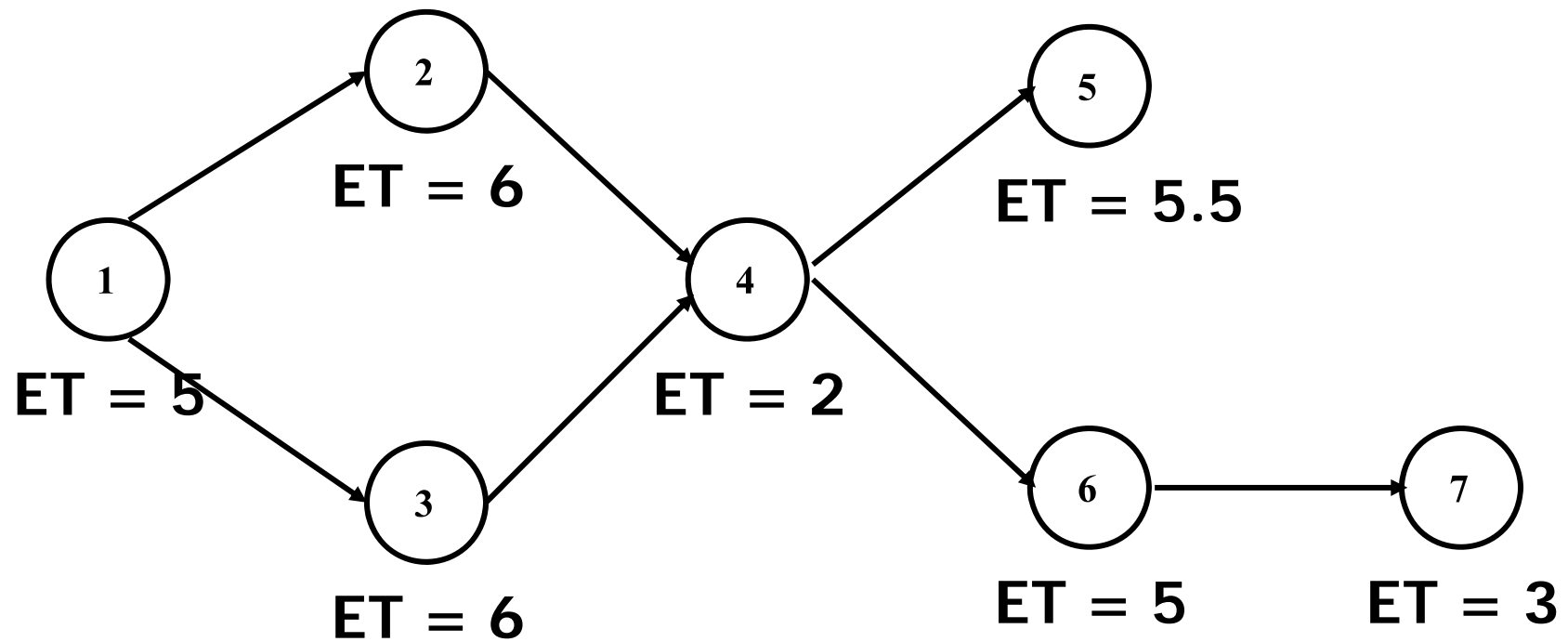
ขั้นตอนที่ 4

4.6 วาดโหนดกิจกรรมที่ 6 ซึ่งมีกิจกรรมที่ 4 ก่อนหน้า



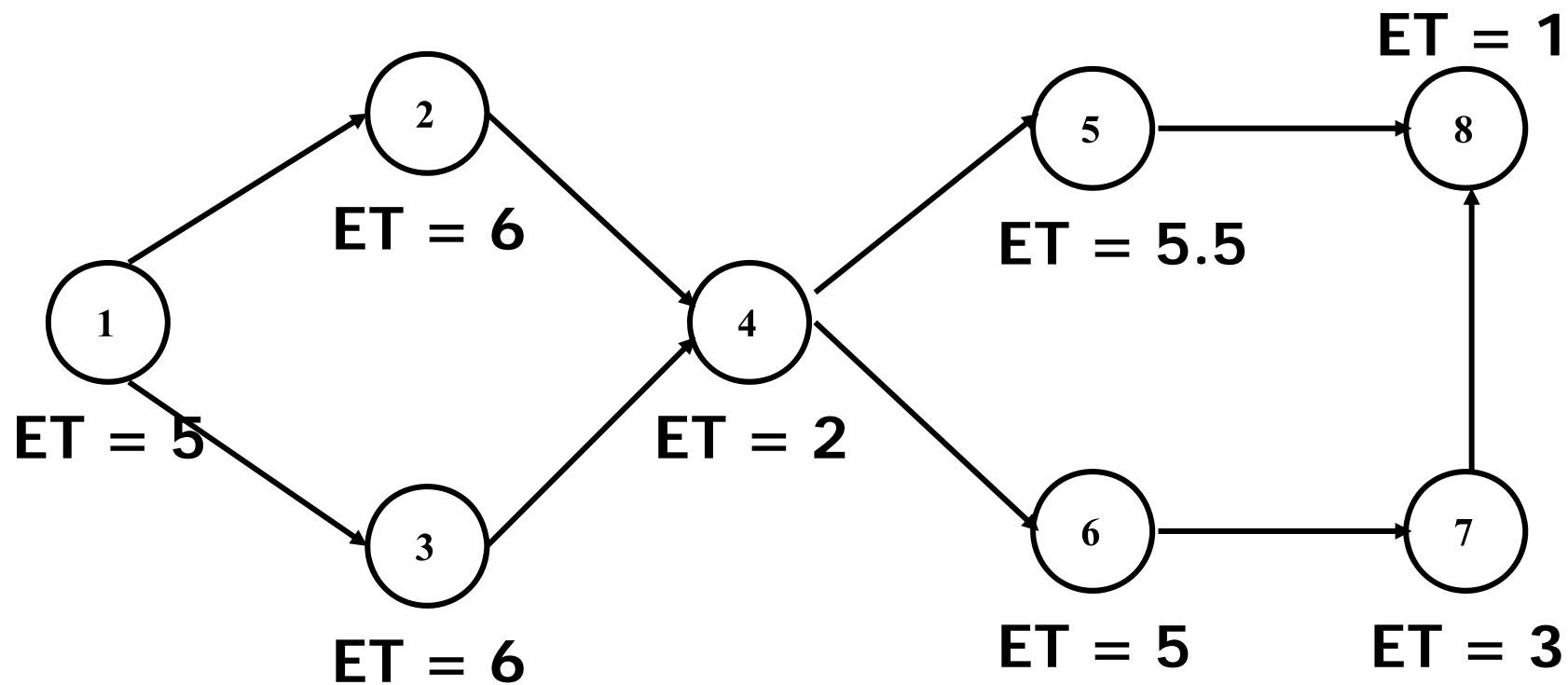
ขั้นตอนที่ 4

4.7 วาดโหนดกิจกรรมที่ 7 ซึ่งมีกิจกรรมที่ 6 ก่อนหน้า



ขั้นตอนที่ 4

4.8 วาดโหนดกิจกรรมที่ 8 ซึ่งมีกิจกรรมที่ 5,7 ก่อนหน้า



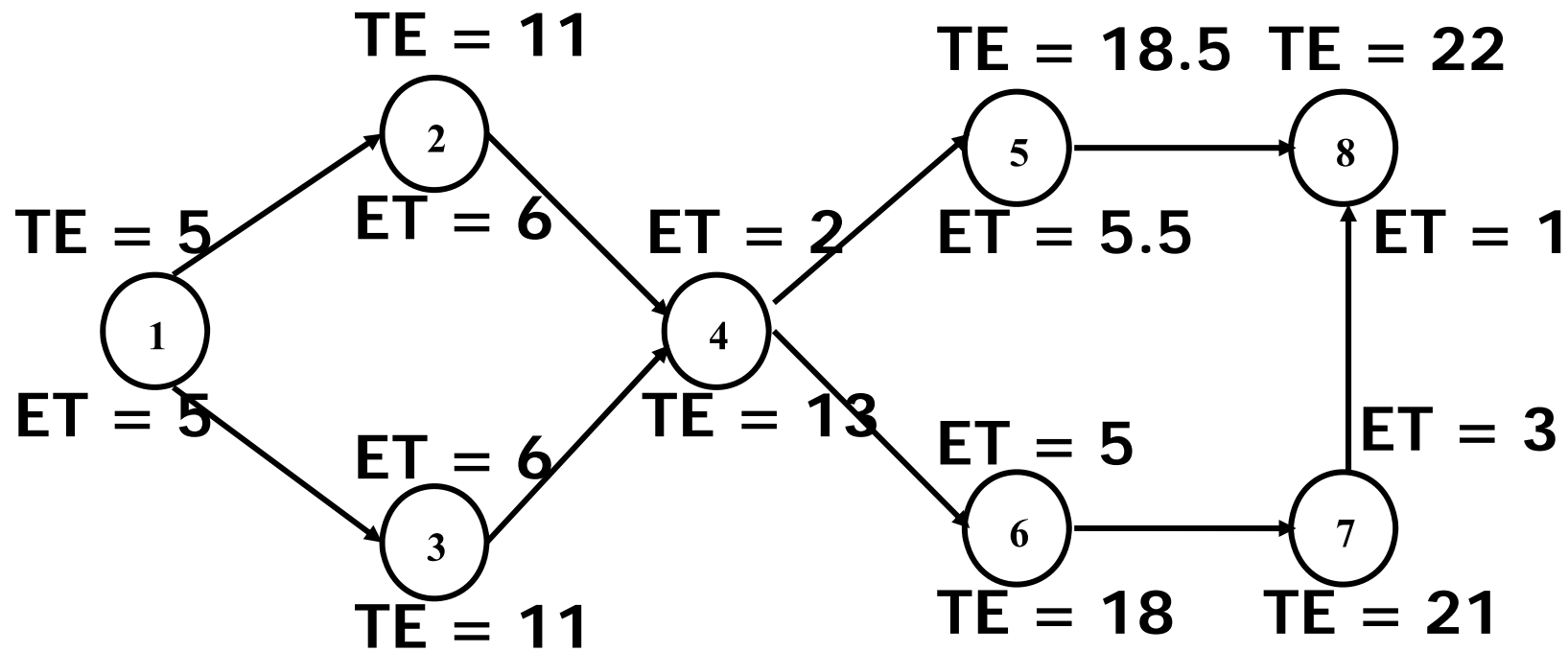
ขั้นตอนที่ 5 : คำนวณหาเส้นทางวิกฤต

5.1 เริ่มหาจากวันแรกสุด (T_E)

Earliest Expected Completion Time : T_E

- โดยทำการบวกสะสมค่า **ET** จากโหนดซ้ายมือไปทางขวาจนถึงโหนดสุดท้ายของแต่ละเส้นทาง

ขั้นตอนที่ 5.1 : หาค่า T_E



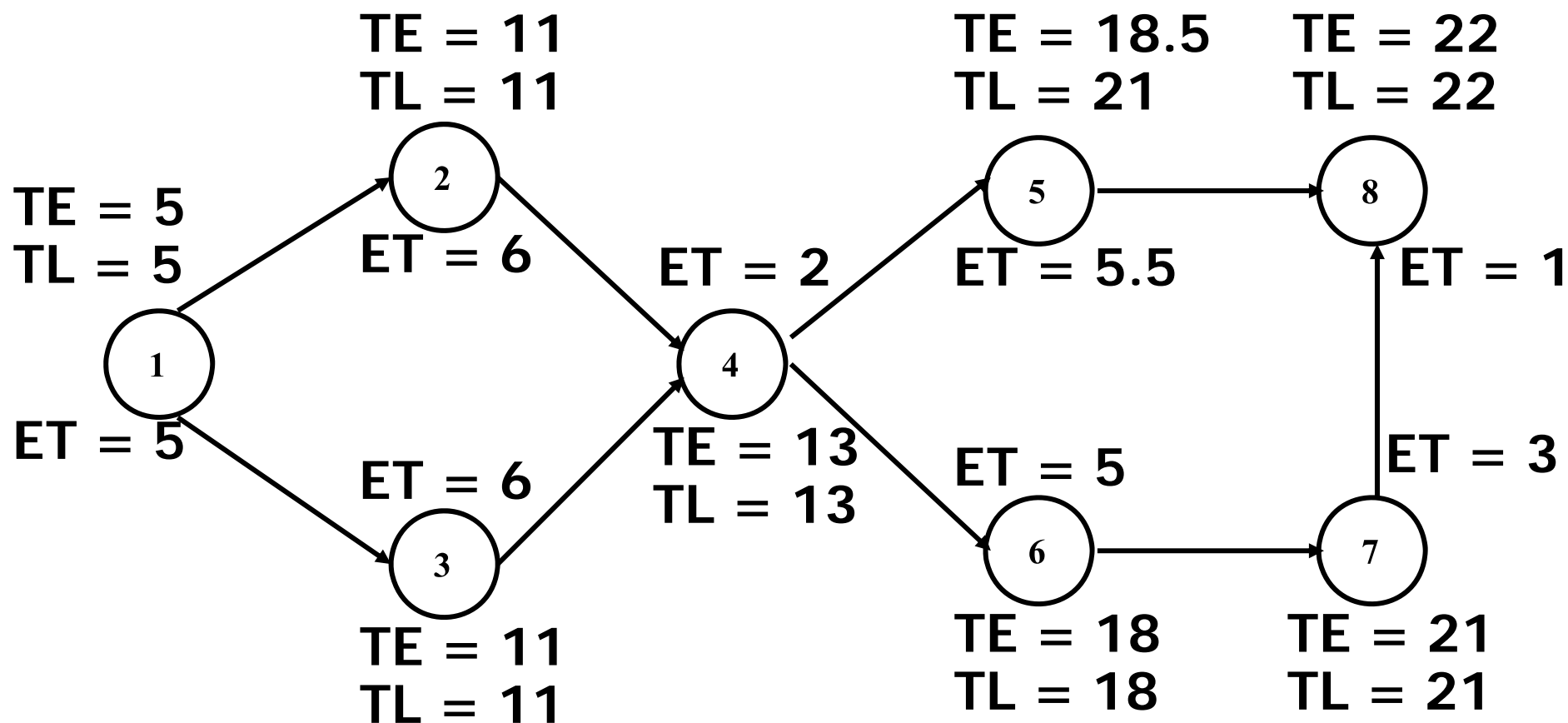
ขั้นตอนที่ 5 : คำนวณหาเส้นทางวิกฤต

5.2 เริ่มหาจากวันสุดท้าย (T_L)

Latest Expected Completion Time : T_L

- ค่าเริ่มต้นของ T_L จะมีค่าเท่ากับ T_E ค่าสุดท้าย จากนั้นให้ทำการลบออกด้วยค่า **ET** ของแต่ละโหนด เริ่มต้นจากโหนดทางขวามือไปทางซ้าย จนถึงโหนดแรกของแต่ละเส้นทาง

ขั้นตอนที่ 5.2 : หาค่า T_L



ขั้นตอนที่ 5 : คำนวณหาเส้นทางวิกฤต

5.3 คำนวณหาค่าเวลายืดหยุ่น (Slack Time)

- คือ ระยะเวลาที่กิจกรรมสามารถล่าช้าโดยไม่ส่งผลกระทบต่อโครงการล่าช้า ซึ่งกิจกรรมที่มีเวลายืดหยุ่นจะอยู่บนเส้นทางที่ไม่ใช่เส้นทางวิกฤต (Noncritical Path)
- หาได้จากผลต่างของ T_E และ T_L
- ถ้าค่าเวลายืดหยุ่นเป็นศูนย์ (0) แสดงว่ากิจกรรมนั้นเป็นกิจกรรมที่อยู่บนเส้นทางวิกฤต

ขั้นตอนที่ 5.3 : คำนวณค่าเวลายืดหยุ่น

กิจกรรม	T_E	T_L	เวลายืดหยุ่น $T_E - T_L$	เส้นทางวิกฤต
1	5	5	0	วิกฤต
2	11	11	0	วิกฤต
3	11	11	0	วิกฤต
4	13	13	0	วิกฤต
5	18.5	21	2.5	-
6	18	18	0	วิกฤต
7	21	21	0	วิกฤต
8	22	22	0	วิกฤต

ข้อแตกต่างระหว่าง Gantt และ PERT/CPM

Gantt Chart	PERT/CPM Chart
<ul style="list-style-type: none">1. เหมาะสำหรับโครงการที่มีขนาดเล็ก2. สามารถแสดงให้เห็นถึงกิจกรรมที่ทำในเวลาเดียวกันได้3. แสดงกิจกรรมที่สำคัญต่อโครงการได้ (Critical Path)	<ul style="list-style-type: none">1. เหมาะสำหรับโครงการที่มีขนาดใหญ่2. สามารถแสดงกิจกรรมที่สำคัญได้ (Critical Path) ทำให้มีการควบคุมการใช้ทรัพยากรได้อย่างคุ้มค่า

Ex 2 PERT/CPM

Activity	ET	PA
1. Gather Requirement	5	-
2. Analyze	6	1
3. Architecture Design	6	2
4. Report Design	2	3
5. Screen Design	5.5	3
6. Document	5	4,5
7. Coding	3	4,5
8. Test	1	7
9. Installation/Launch	1	6,8

สรุป : กราฟ (graph)

- กราฟ (graph) เป็นโครงสร้างข้อมูลแบบไม่เป็นเชิงเส้น (nonlinear data structure)
- กราฟประกอบด้วยโหนดและเอจ แต่ละโหนดสามารถมีความสัมพันธ์กับโหนดอื่นๆ ได้มากกว่าหนึ่ง โดยไม่พิจารณาถึงลำดับความสัมพันธ์ก่อนหลัง
- กราฟ เป็นโครงสร้างข้อมูลที่มีการนำไปใช้ในงานที่เกี่ยวข้องกับการแก้ปัญหาที่ค่อนข้างซับซ้อน เช่น การวางข่ายงานคอมพิวเตอร์ การวิเคราะห์เส้นทางวิกฤติ การวางแผนข่ายงาน และปัญหาเส้นทางที่สั้นที่สุด เป็นต้น

สรุป : กราฟ (graph)

- การแทนกราฟในความจำหลักวิธีที่ง่ายและตรงไปตรงมาที่สุดคือ การเก็บเอ็จทุก ๆ เอ็จในแถวลำดับ 2 มิติ แต่วิธีนี้ค่อนข้างเปลืองเนื้อที่เนื่องจากมีบางเอ็จที่เก็บซ้ำเดิม
- อีกวิธีหนึ่งก็คือใช้แถวลำดับ 2 มิติเก็บโหนดและพอยน์เตอร์ชี้ไปยังตำแหน่งโหนดต่าง ๆ ที่สัมพันธ์ด้วย
- หรือใช้วิธีแอดจาเซนซีลิสต์โดยใช้ลิงค์ลิสต์แทนแถวลำดับ 1 มิติ

สรุป : กราฟ (graph)

❁ วิธีที่นิยมมากที่สุดคือ การแทนด้วยแอดจาเซนซีเมทริกซ์ โดยถ้ากราฟของเรามี n โหนดต้องสร้างเมทริกซ์จัตุรัสขนาด $n \times n$ และค่าในเมทริกซ์จะเก็บค่าระหว่างโหนดสองโหนดมีคู่ใดบ้างที่มีความสัมพันธ์กัน เราสามารถหาได้ว่ามีเส้นทางขนาดเท่าใด และมีกี่เส้นทาง