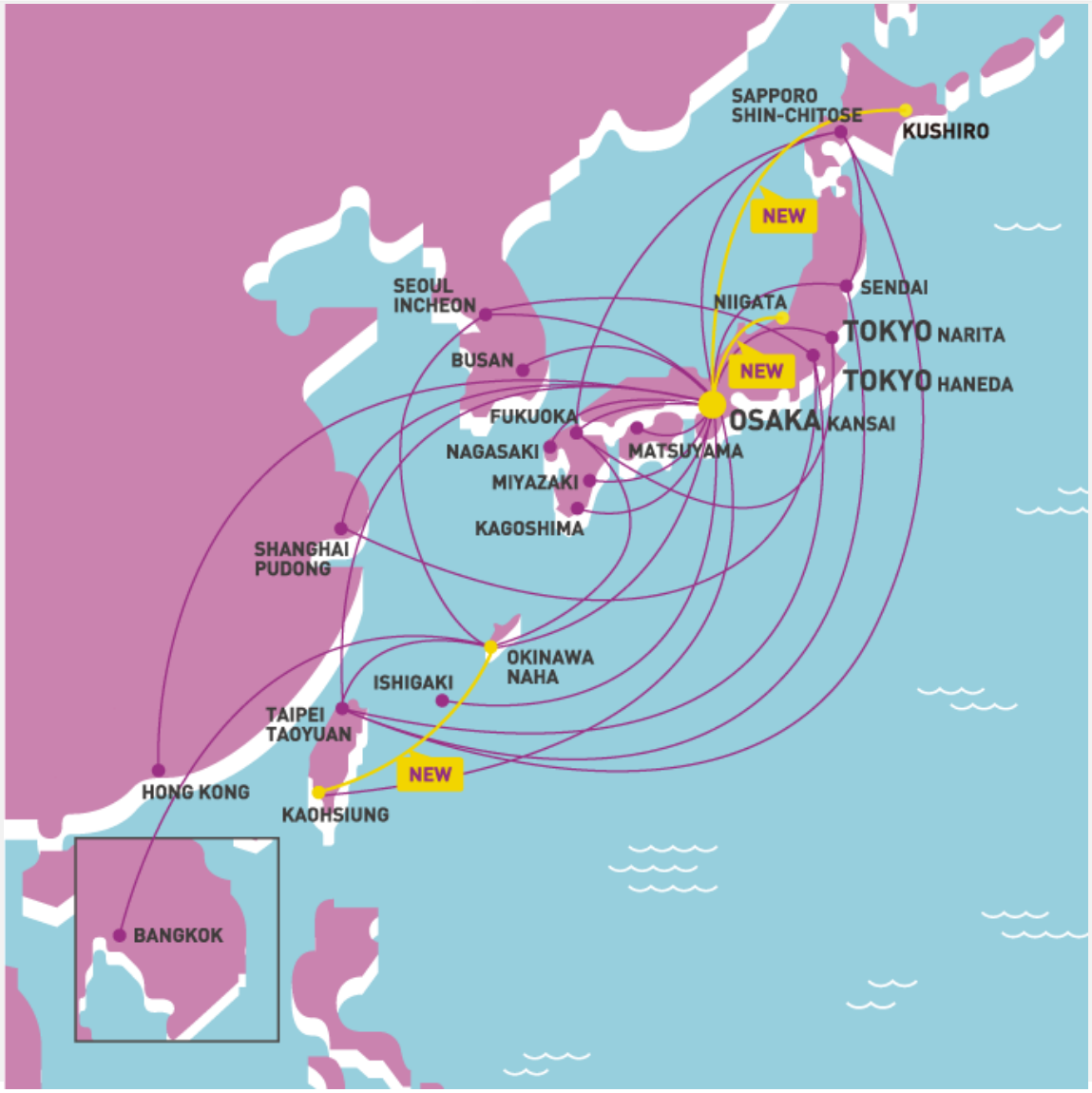
# Module4—Graph

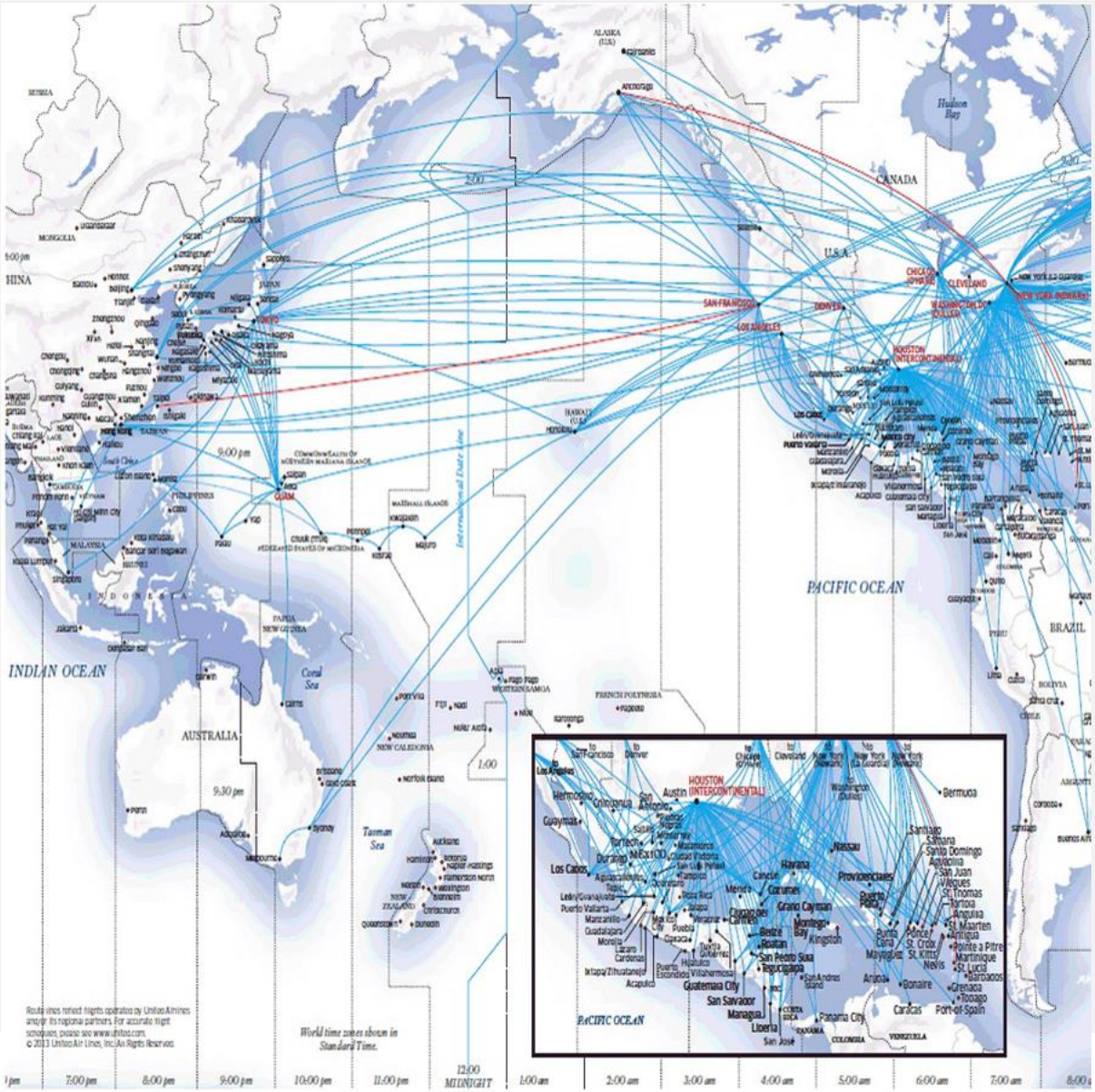
1. กราฟคืออะไร ทำไมต้องใช้โครงสร้างข้อมูลแบบกราฟ
2. นิยามของกราฟ digraph, undigraph, vertice, edge, adjacent, incident, degree, path
3. การเก็บกราฟด้วย adjacency list
4. การท่องกราฟด้วย BFS , DFS




- มีทิศทาง
- มีระยะทาง

**ตัวอย่างงาน**

- การวางข่าย งานคอมพิวเตอร์
- การวิเคราะห์เส้นทางวิกฤติ
- ปัญหาเส้นทาง ที่สั้นที่สุด


CANADA
U.S.A.
PACIFIC OCEAN
INDIAN OCEAN
AUSTRALIA
BRAZIL
HOUSTON (INTERCONTINENTAL)
PACIFIC OCEAN
World time zones shown in Standard Time.
MIDNIGHT

## 4.1 Definition

กราฟเส้นตรง

เซ็ตของ node

$V = \{A, B, C, D, E\}$

**1) A directed graph(or digraph) G** : is a pair(V,E), where V is a finite set and E is binary relation on V.

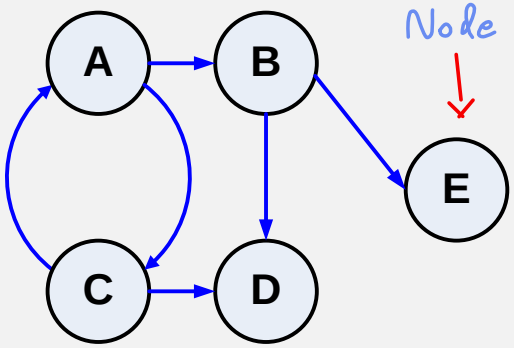
edge

ความสัมพันธ์ที่เชื่อม จุด 2 จุด

เส้นเชื่อม

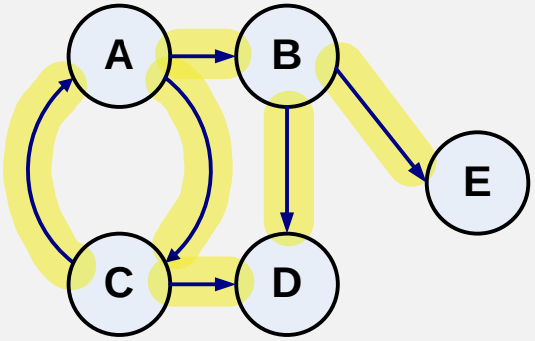
- The ***set V*** is called the **vertex set** of G, and its elements are called vertices.

V={A,B,C,D,E}



edge มีทิศทาง

- The ***set E*** is called the **edge set** of G, and its elements are called edges.




The edge set E consists of ordered pairs of vertices

มีลำดับของมัน

V={A,B,C,D,E}

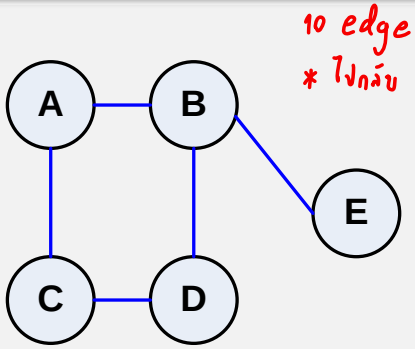
E={ (A,B), (A,C), (B,D), (B,E), (C,A) } , (C , D)

## 2) An undirected graph G :

กราฟที่ไม่มีทิศทาง

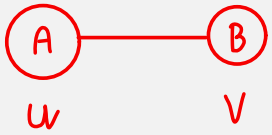
G = (V,E) , the edge set E consists of unordered pairs of vertices, rather than ordered pairs.

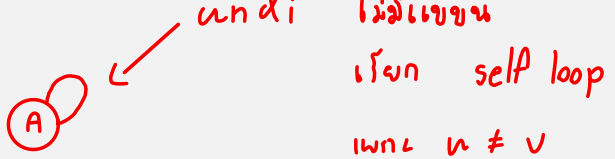
ไม่มีลำดับ




V={A,B,C,D,E}

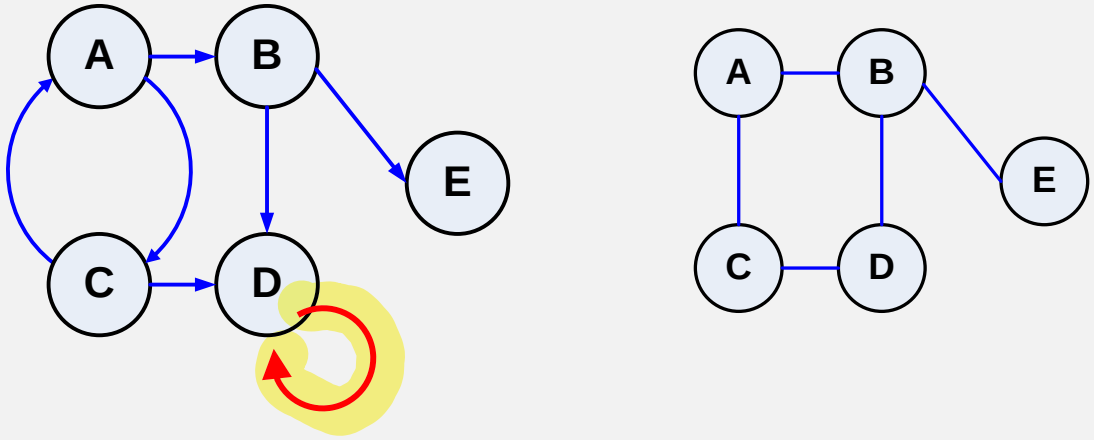
E={ (A,B), (B,A), (A,C), (C,A), (B,D),(D,B), (B,E), (E,B), (C,D),(D,C) }



**3) self-loops** : edges from a vertex to itself.

**Undigraph self loop are forbidden**, and so every edge consists of exactly two distinct vertices.



## 4) Incident :

ทิศทาง

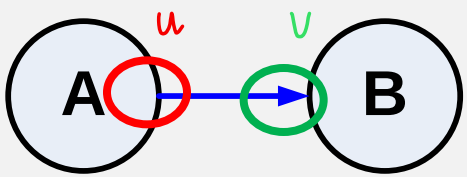
**digraph** : if (u,v) is an edge in a directed graph G=(V,E), we say that (u,v) is

***incident from*** or leaves vertex u

ออก

***incident to*** or enters vertex v.

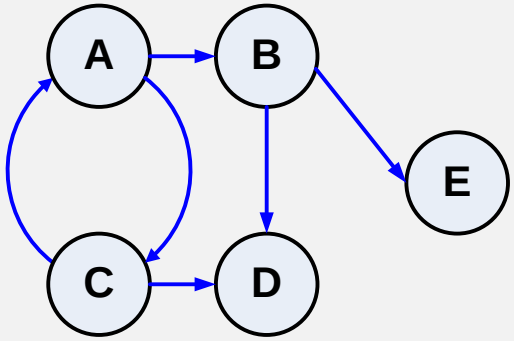
เข้า




**Question** edge (B,E)

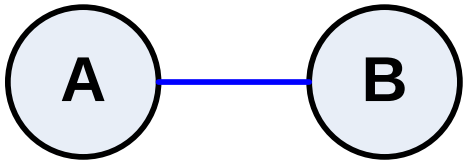
Incident from ..... B

Incident to ..... E



**undigraph** : if (u,v) is an edge in a directed graph G=(V,E), we say that(u,v) is ***incident on*** vertices u and v

is ***incident on*** vertices u and v



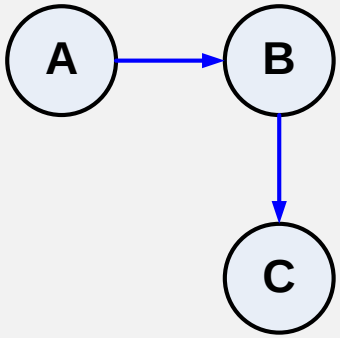

**Question** edge (A,B)
Incident from .....
Incident to .....

Edge (A,B) is incident on vertices A and B

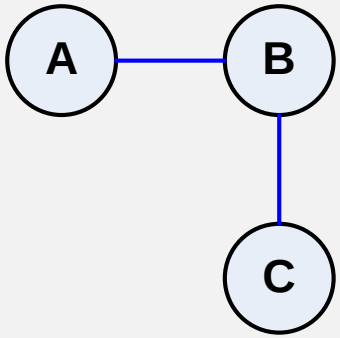
## 5) adjacent (ติดกัน , ประชิด)

**Digraph** : If (u,v) is an edge in a graph G=(V,E), we say that v is adjacent to vertex u. If v is adjacent to u denote by u -> v (A,B) B ติดกับ A คือ A -> B

**Undigraph** : Adjacent relation is symmetric.




B ประชิด A A->B




B ประชิด A และ A ประชิด B

## 6) Degree

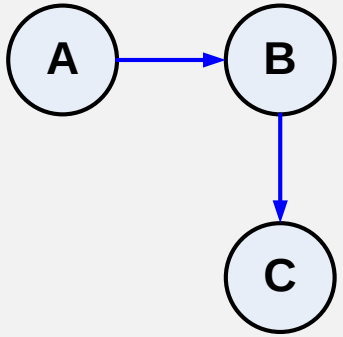
ลูก

**Digraph** : in degree (number of edges entering it.),

out degree

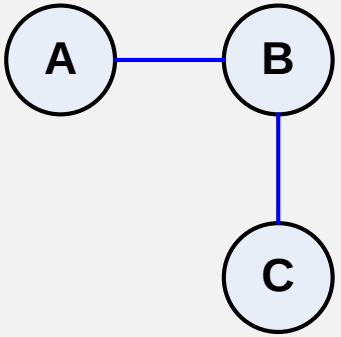
ลูก ศร ชี้เข้าหา

(Opposite in degree.) ลูกศรชี้ออก




A in degree = 0
A out degree = 1

**Undigraph** : is the number of edges incident on it.

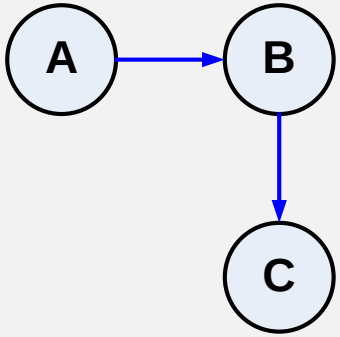



A degree = 1

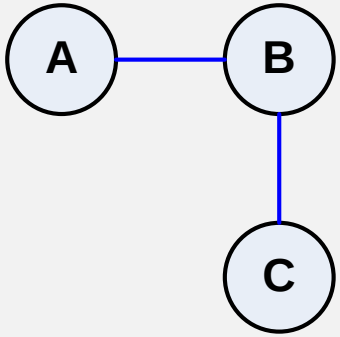
B degree = 2

C degree = 1

**7) path of length :** Path k from a vertex u to a vertex u' in a graph G=(V,E) is a sequence $(v_o, v_1, .., v_k)$ of vertices such that $u = v_0$, $u' = v_k$, and $(v_{i-1}, v_i) \in E$ for i = 1, 2, .., k




PATH A,C = (A,B), (B,C)



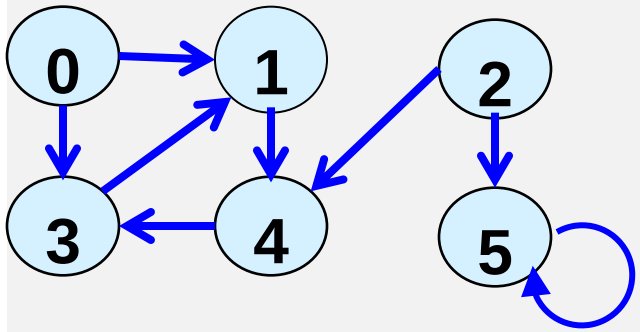

(A,B), (B,A), (A,B), (B,C)

## 4.2 โครงสร้างข้อมูลที่ใช้เก็บข้อมูลในรูปแบบกราฟคือ

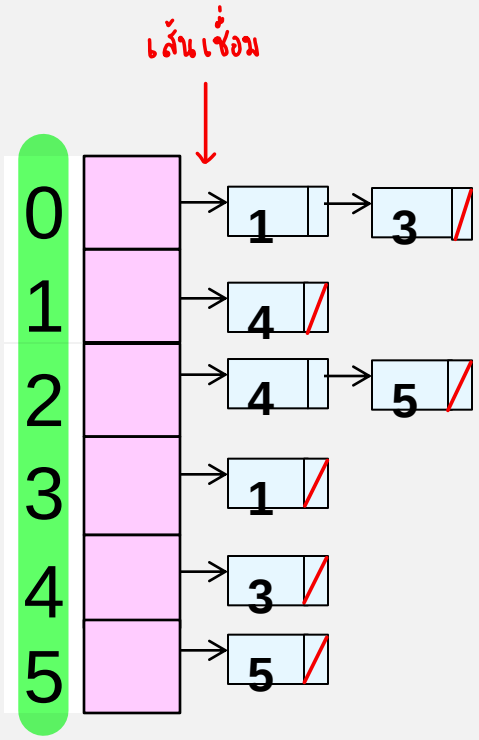
1. Adjacency list * ใช้ list ในการเรียนการสอนนี้
2. Adjacency Matrix

โครงสร้างข้อมูลทั้ง 2 แบบนี้จะสามารถเก็บกราฟได้ทั้ง digraph และ undigraph

## Digraph




1. Adjacency list
2. Adjacentcy matix

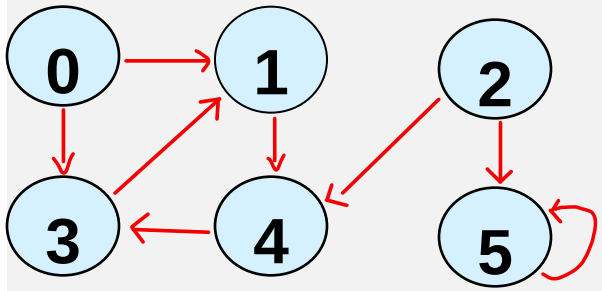



Space O(V+E)

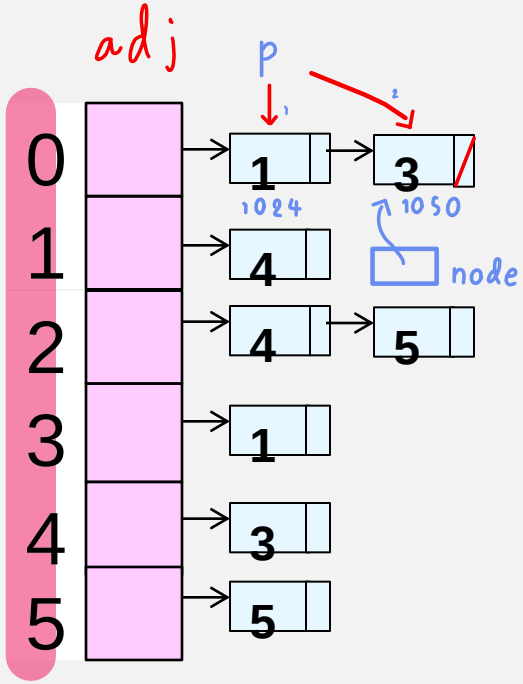
| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 |

Space $O(n^2)$

# Digraph




1. Adjacency list
2. Adjacentcy matix



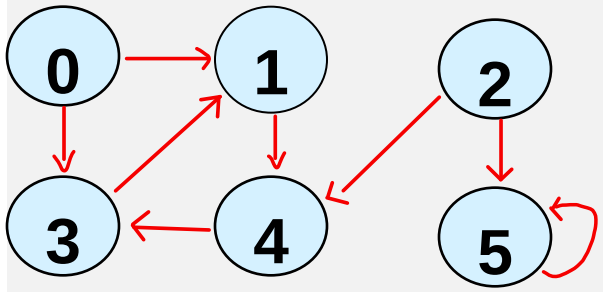

```
struct record {
        int value;
        struct record *next;
};

Enter 0# : 1  3  -1

if(adj[i] == NULL){
   adj[i] = new struct record;
   adj[i] -> value = data;
   adj[i] -> next = NULL;
   p = adj[i];
}
```

```
struct record *adj[6]; // ไว้เก็บตำแหน่ง ขนาด array ขึ้นอยู่กับ node
for(i=0; i<6; i++){
    adj[i] = NULL;
}
```

# Digraph



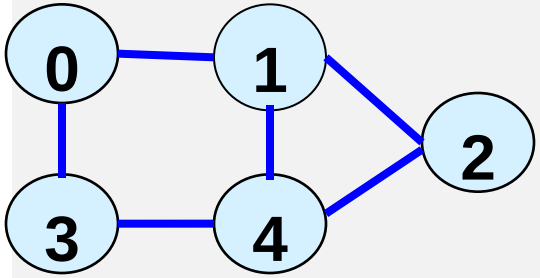

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 |

1. Adjacency list
2. Adjacentcy matix

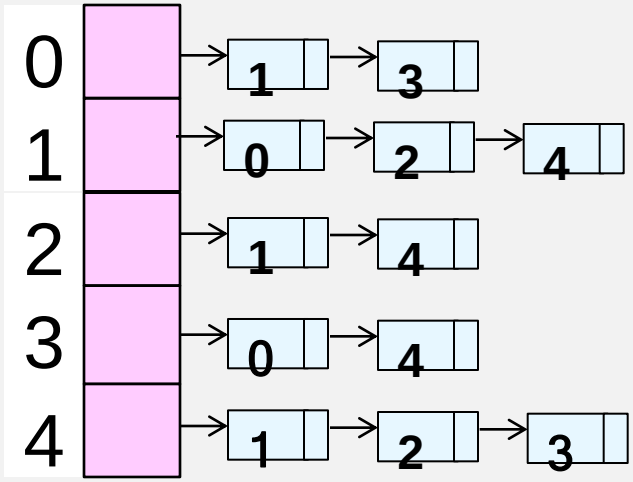
*ถ้า ทนชาโบส G จะกลับทิศ

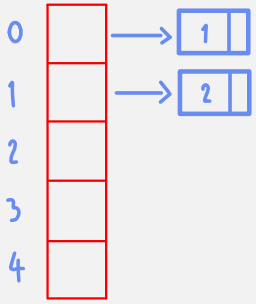
## Undigraph

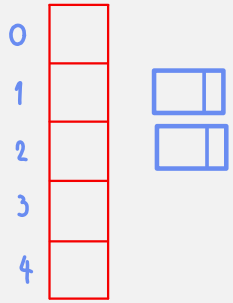
12 edge
5 vertex



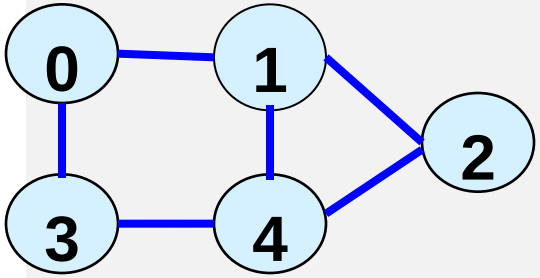

## Adjacency list



# Adjacency Matrix



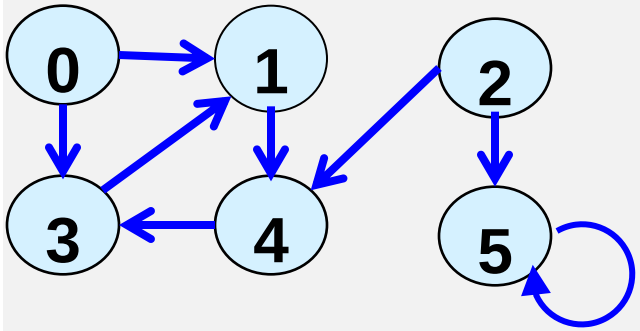

ปลาย

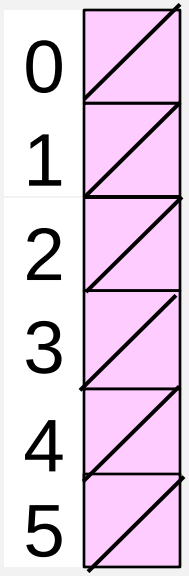
| ต้น Node | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 1 | 0 |

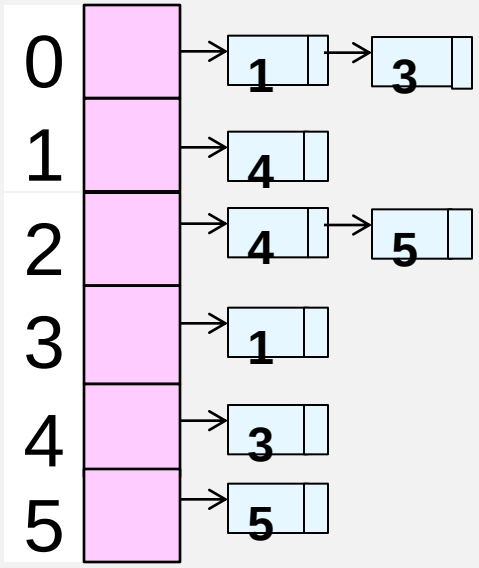
* เส้นทแยงมุมเป็น 0 เสมอ เพราะไม่มี self-loop

* ถ้าแลีย แถว -> หลัก, หลัก -> แถว จะได้กราฟเดิม

## การบ้าน

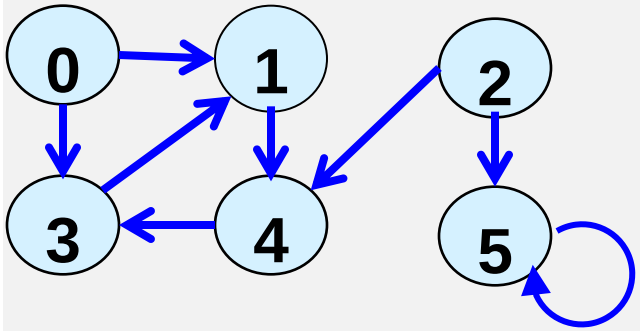


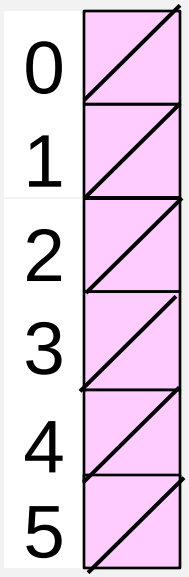




Enter
#0 : 1 3 -1
#1 : 4 -1
…

**การบ้าน**




Enter
#0 : 1 3 -1
#1 : 4 -1
…