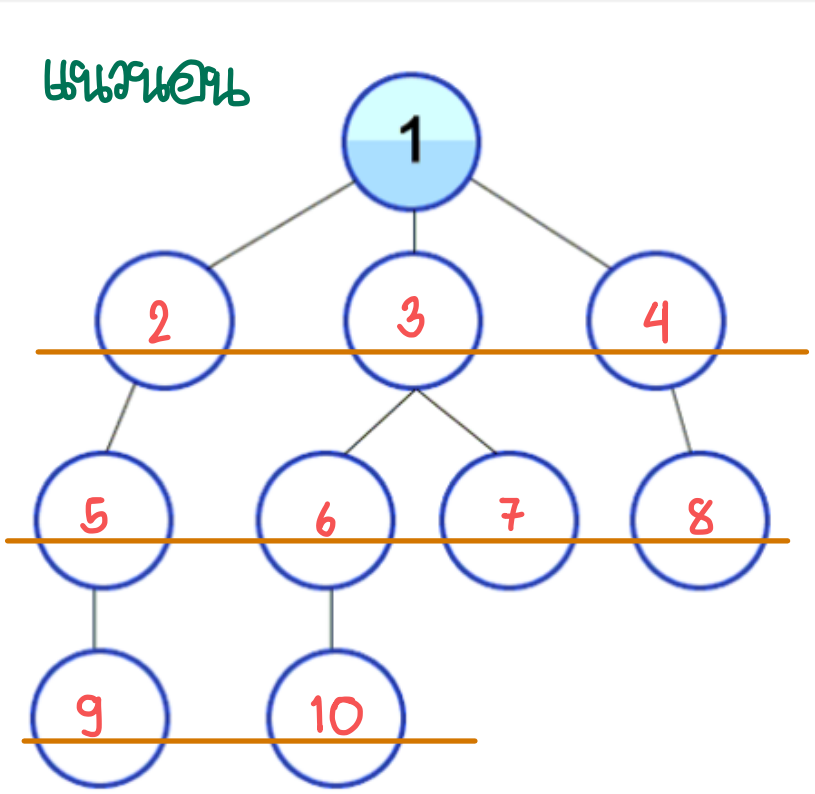
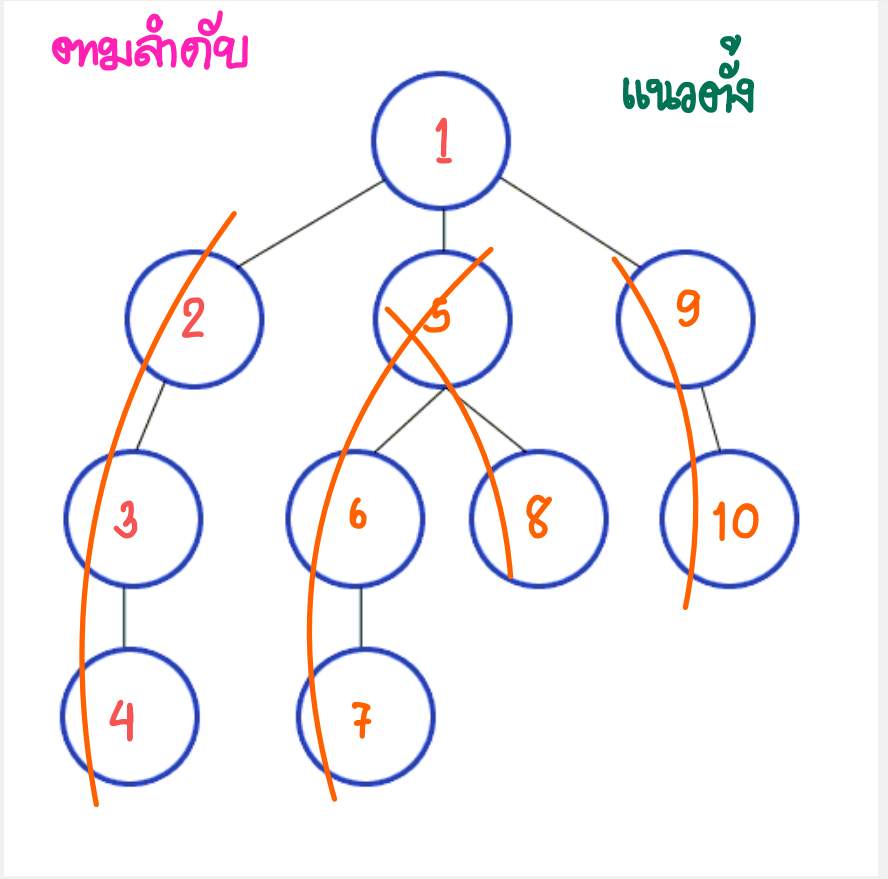
4.4 DFS คืออะไร

- The **Depth First Search** is fundamental search algorithm used to explore node and edges of a graph.
- It runs with a time O(V+E) big O (vertex + edge)(4 vertex + 3 edge)
- Often used as a building block in other algorithms.

vector = array แบบ dynamic

ยืดหดได้ ไม่ต้องจองพื้นที่





BFS DFS

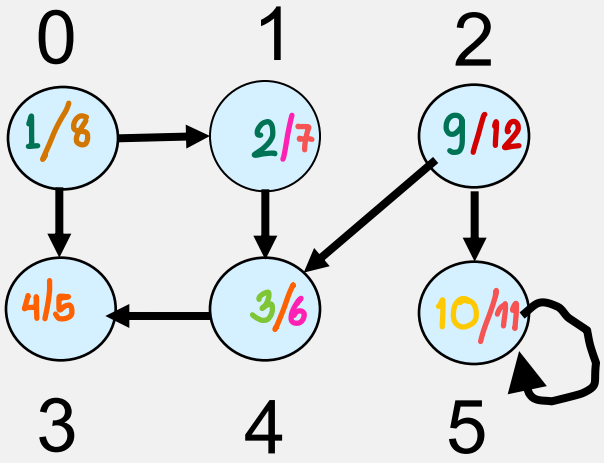
https://commons.wikimedia.org/wiki/File:Depth-First-Search.gif

วัตถุประสงค์ & ขั้นตอนการทำงาน

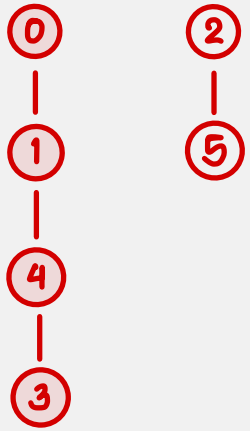
- สำรวจตามความลึกของกราฟ
- โดยหาเส้นทางที่ลึกที่สุดจากการเดินจาก เวอร์เท็กซ์ u ไปยัง เวอร์เท็กซ์ v ซึ่งเป็นเวอร์เท็กซ์ข้างเคียงที่ยังไม่ได้เดินผ่าน ทำเช่นนี้จนกระทั่งไม่มี edge ที่สามารถเดินต่อไปได้อีก จากนั้นจึงทำการเดินกลับ (backtrack) ไปยัง source
- A DFS plunges depth first into a graph without regards for which edge it takes next until it cannot go any further at which point it backtracks and continues.

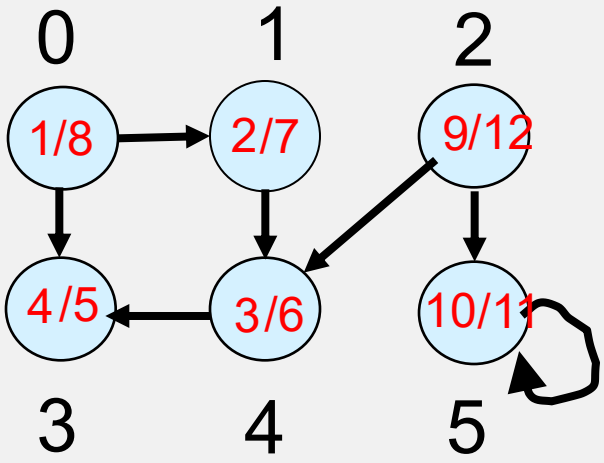
ขั้นตอน(ภาษาอังกฤษ)

1. Edges are explored out of the most recently discovered vertex v that still has unexplored edge leaving it.
2. When all of v's edges have been explored, the search backtracks to explore edges leaving the vertex from which v has discovered. Until we have discovered all the vertices that are reachable from the original source vertex

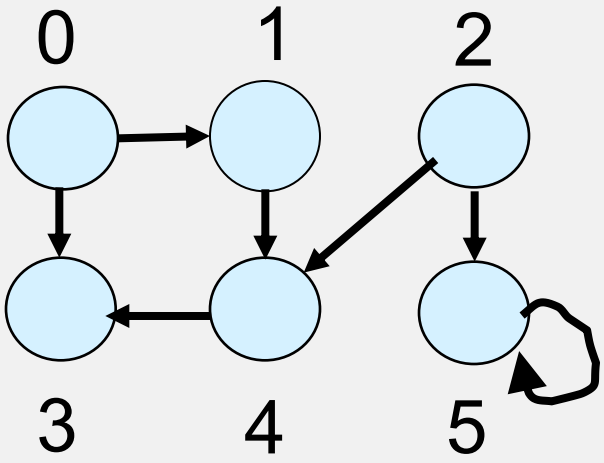


S= 0

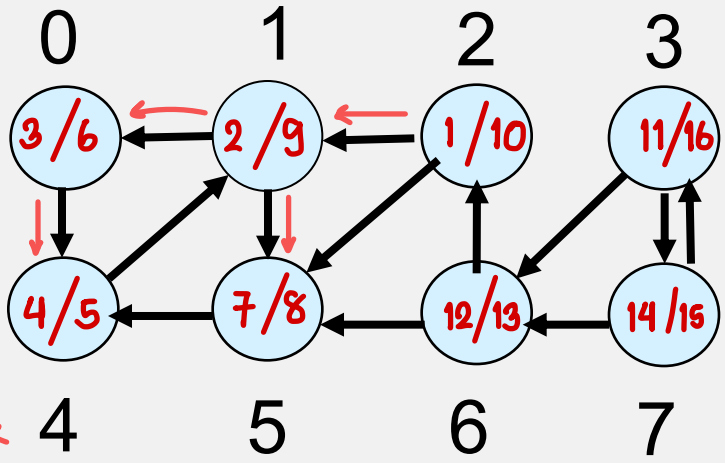


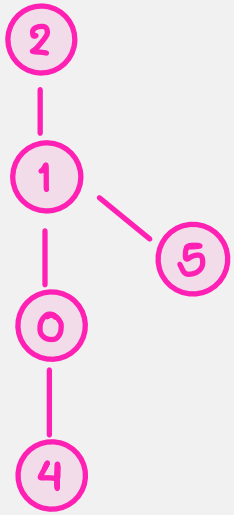

0
1
2
1/8
2/7
9/12
4/5
3/6
10/11
3
4
5

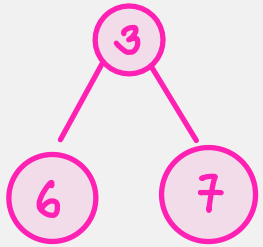
S= 4



S= 2



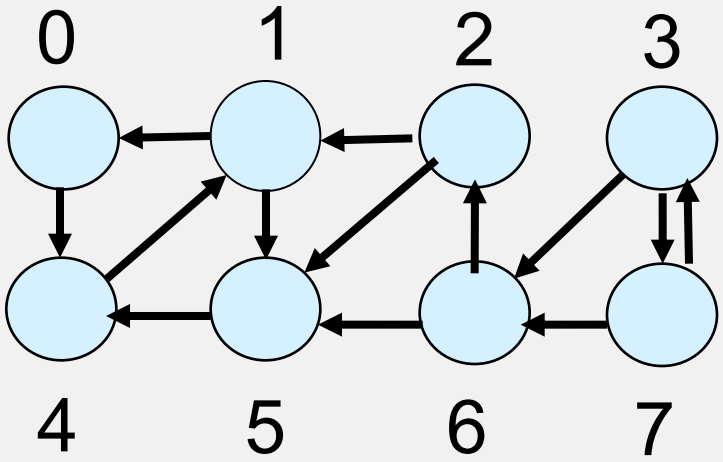


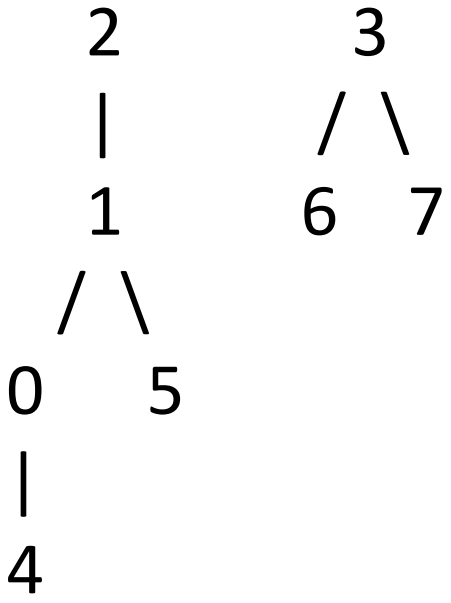


มีทางแยกให้เดินลงให้ครบก่อน

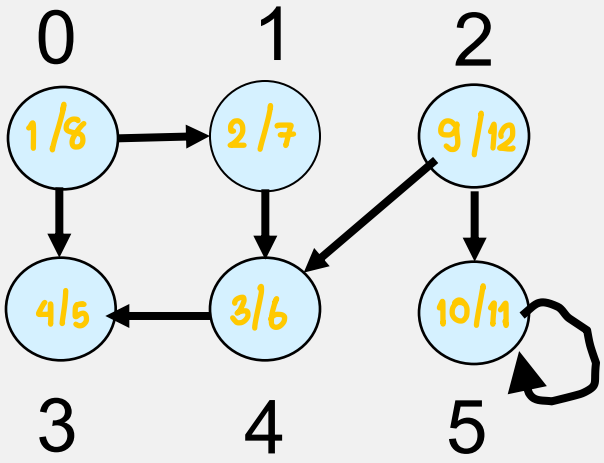
ก่อนจะวนถึงตัวเอง

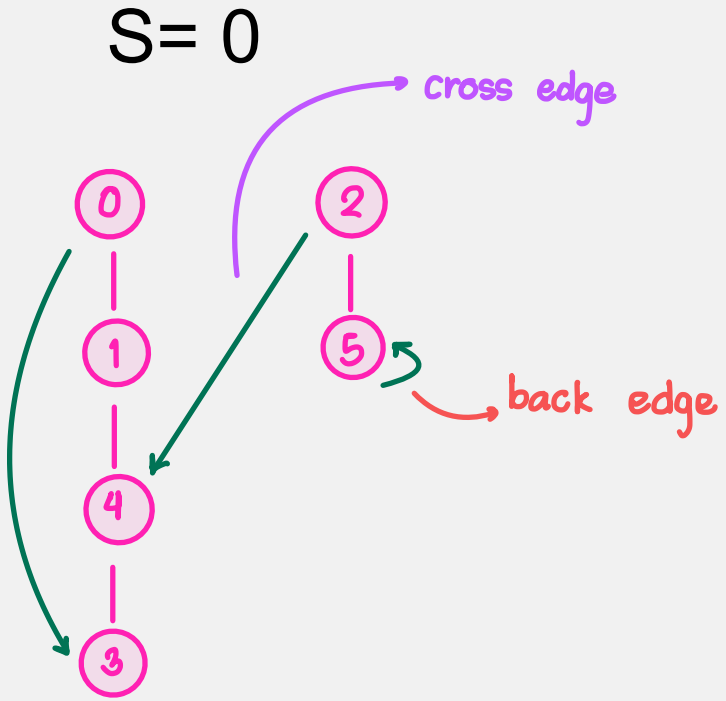
DFS trees



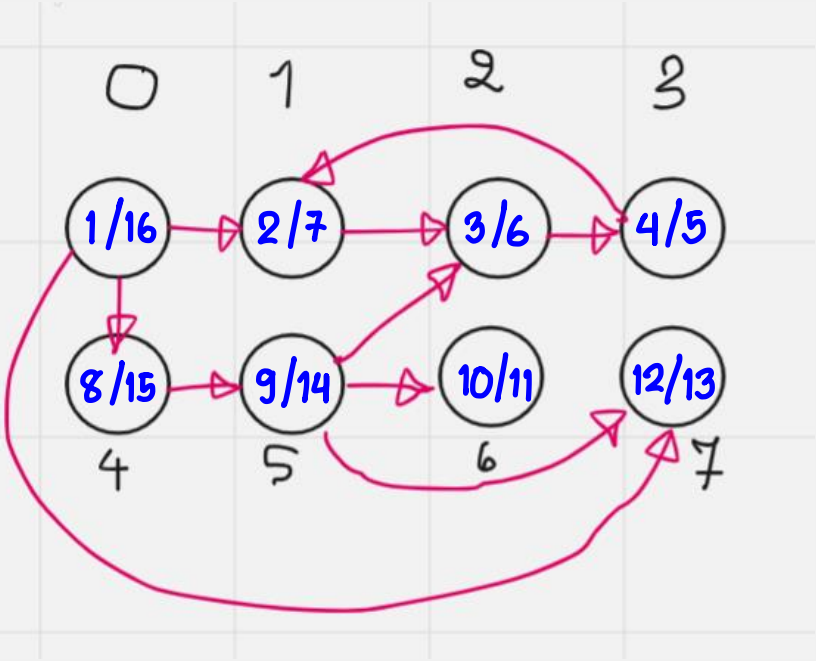


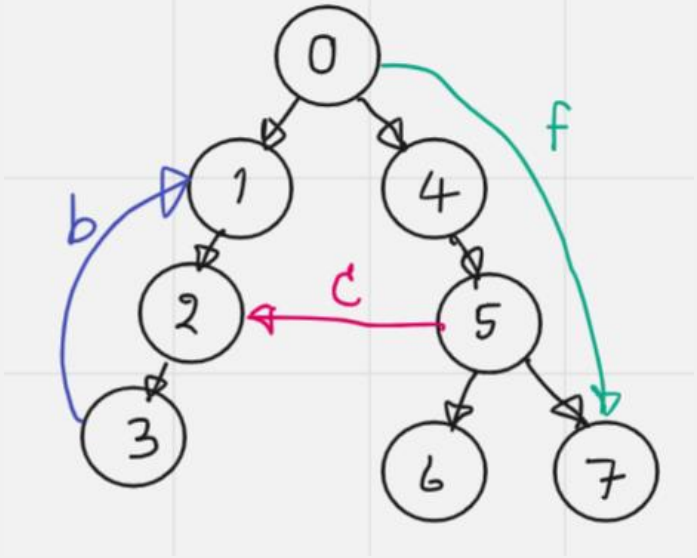
Edge Classification





Edge Classification





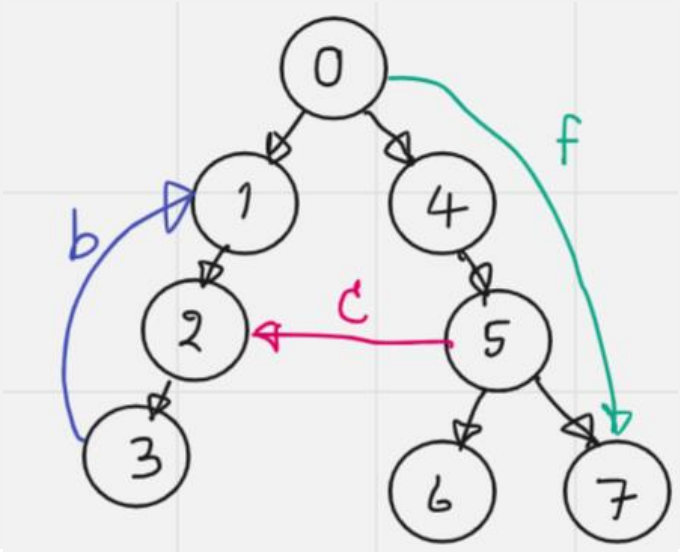
Edge Classification

1. **Tree edges** : are edges in the dfs forest G''. Edge (u,v) is a tree edge if v was discovered by exploring edge(u,v).

ลูกหลานไปบรรพบุรุษ

2. **Back edge** : are those edges (u,v) connecting a vertex u to an ancestor v in a dfs tree. Self-loop are considered to be back edges.

which point from a node to one of its ancestors



มี back edge ตรงไหน
เป็น cycle #

Classification of Edge

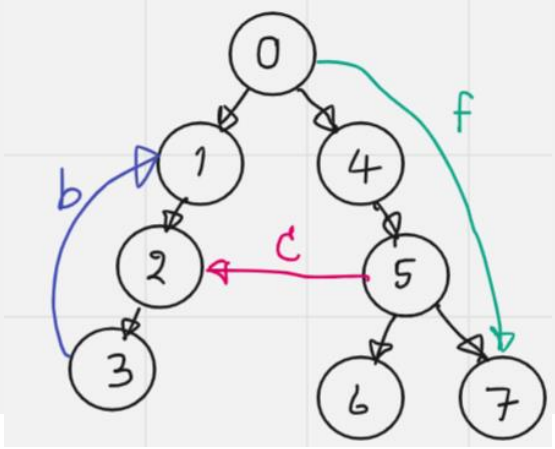
บรรพบุรุษไปลูกหลาน

3. **Forward edges** : are those non tree edge (u,v) connecting a vertex u to a descendent v in a dfs tree.

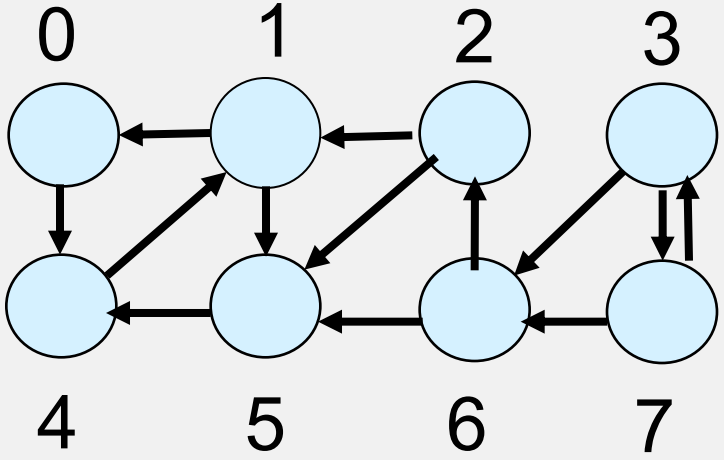
which point from a node of the tree to one of its descendants

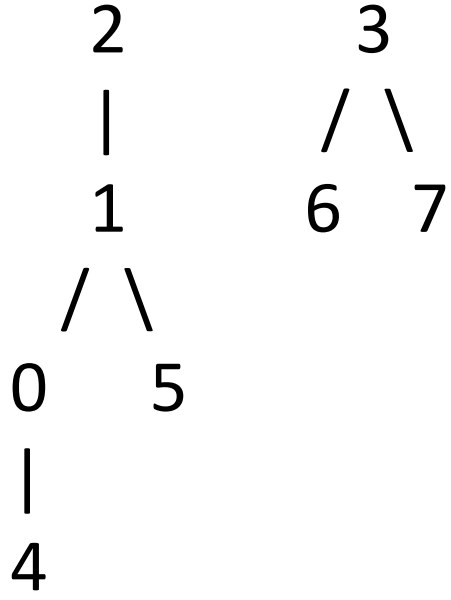
ข้ามระหว่าง Trees

4. **Cross edge :** are all other edges. They can go between vertices in the same depth-first search tree, as long as one vertex is not ancestor of the other, or they can go between vertices in difference dfs tree.



DFS trees





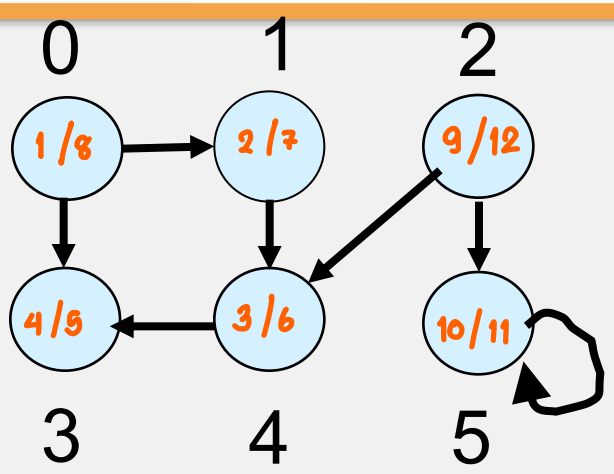
1. Tree edges
2. Back edges
3. Forward edges
4. Cross edges

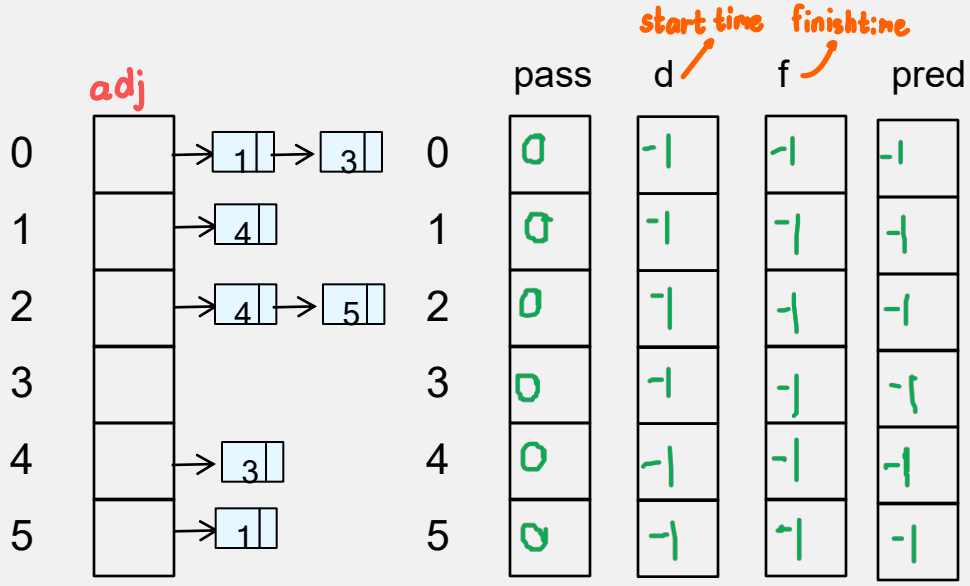
4.4.2 Algorithm DFS (G)

```
DFS(G)
time=0;
for ( u=0;u< 6; u++)
{       pass[u]=0
        pred[u]=d[u]=f[u]=-1;
}
u=S  //สมมุติ s คือ โหนด 0
DFS_Visit(u)
for ( u=0;u<6;u++)
{       if (pass[u]=0)
        {   DFS_Visit(u)
        }
}
```

implement by vector

in BFS d is distance (ระยะทาง)





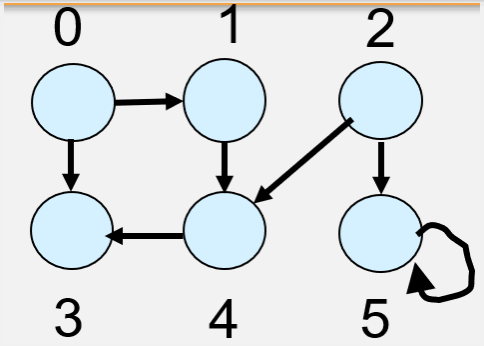
0 , 1 , 2

```
// time=0 is global
 DFS_Visit(u)
 pass[u]=1
 d[u]= time++;
 for each v ∈ adj[u]
 {       if (pass[v] == 0)
         {       pred[v] = u
                 DFS_Visit(v)
         }
 }
 pass[u]=1
 f[u]=++time;
```

recursive

u 0 < 1

u 1 < 4



deg

0	2
1	1
2	2
3	0
4	1
5	1

adj

0	→ 1 → 3	0
1	→ 4	1
2	→ 4 → 5	2
3		3
4	→ 3	4
5	→ 1	5

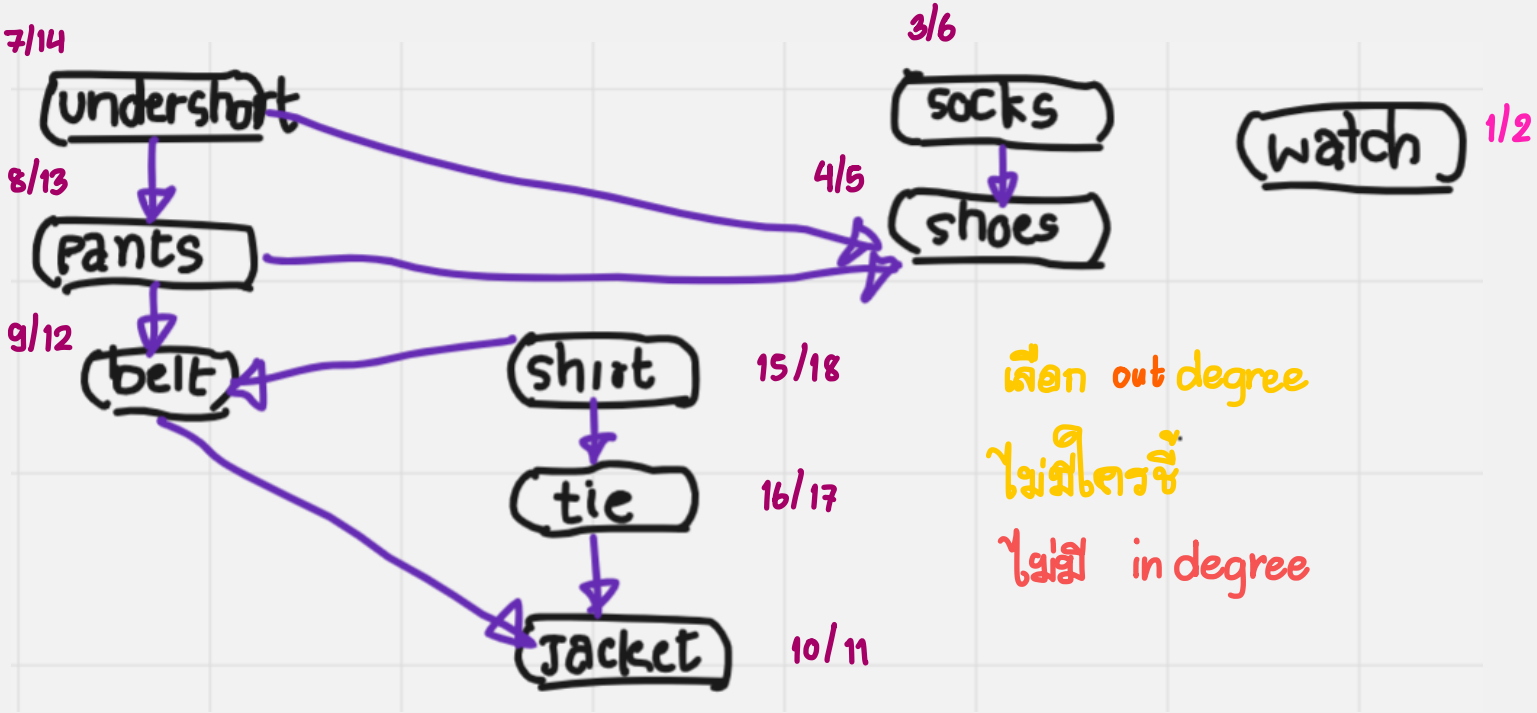
	pass	d	f	pred
0	~~0~~ 1	-1 ~~1~~	-1	-1
1	~~0~~ 1	-1 ~~2~~	-1	-1 ~~0~~
2	0	-1	-1	-1
3	0	-1	-1	-1
4	~~0~~ 1	-1	-1	-1
5	0	-1	-1	-1

4.4.3 Topological sort

กราฟไม่มี cycle ไม่มี back edge
Direct Acyclic Graph

Topological sort of a DAG G=(V,E) is a linear ordering of all its vertices such that if G contains an edge(u,v) , then u appears before v in the ordering. (If the graph is not acyclic, then no linear ordering is possible.)

Topological sort คือการนำเวอร์ทิซของกราฟกราฟหนึ่งมา
- เรียงเป็นเส้นตรงแบบมีลำดับ
- มีข้อกำหนดว่ากราฟนั้นจะต้องมีคุณสมบัติเป็น directed acyclic graph หรือ DAG (คือกราฟที่ไม่มีไซเคิล)



shirt → tie → undershort → pants → belt → jacket → socks → shoes → watch

4.4.4 Algorithm Topological_sort(G)

Topological_sort(G)

1) call DFS(G) to computer finishing times f[v] for each vertex v.

- ทำ DFS กราฟ

2) as each vertex is finished, insert it onto the front of a linked list

- เมื่อ DFS แล้ว ให้นำ vertex ที่มี finish time สูงสุดใส่ไว้ตำแหน่งแรกของ linklist

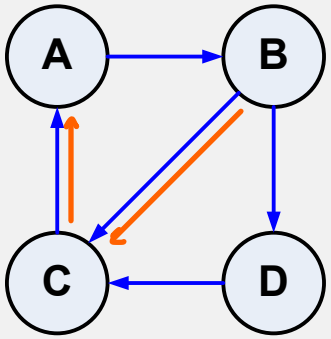
- finish time ที่มีค่ารองลงมาให้นำไปใส่ไว้ตำแหน่งที่ 2 ของ linklist และทำเช่นนี้โดยเรียงจากมากไปน้อย

3) return the link list of vertices.

ส่วนประกอบที่เชื่อมต่ออย่างแข็งแรง

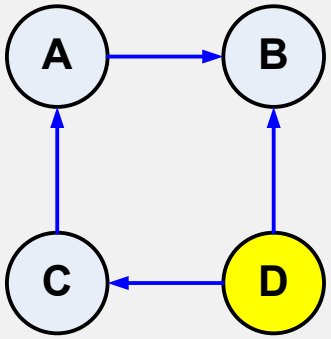
4.4.5 Strongly connected components

A digraph is ***strongly connected*** if every two vertices are reachable from each other. The ***strongly connected components*** of a graph are the equivalence classes of vertices under the "are ***mutually reachable***" relation.

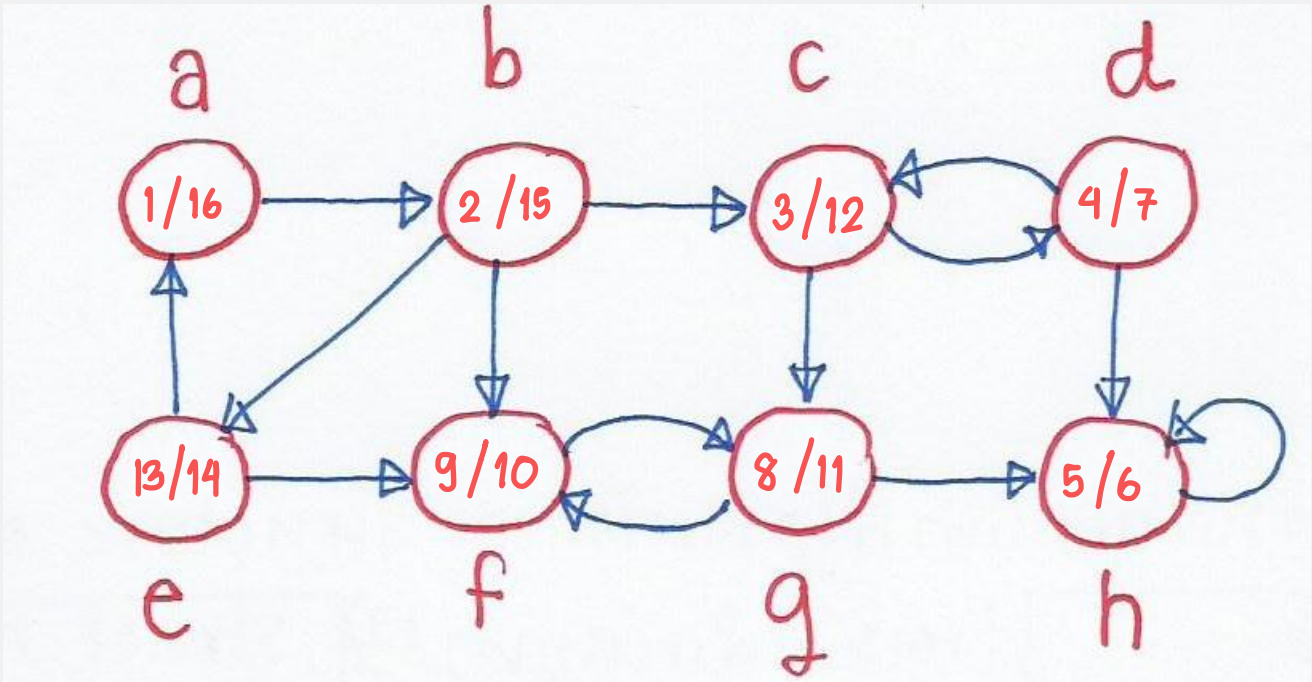


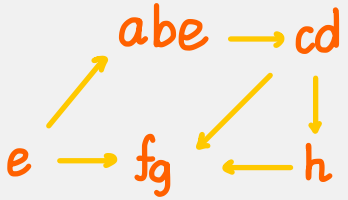
ไปหากันได้ every node

A ไป B
B ไป A



Not mutually reachable





G

	0	1	2
0	0	1	1
1	0	0	0
2	0	0	0

G^T

	0	1	2
0	0	0	0
1	1	0	0
2	1	0	0

4.4.6 Strongly-Connected-Component Algorithm

1) call DFS(G) to compute finishing times f[u] for each vertex u decompose แยกองค์ประกอบ

2) compute GT (transpose graph)

3) call DES(GT), but the main loop of DFS, consider the vertices in order of decreasing f[u] (as computed in line 1)

4) output the vertices of each tree