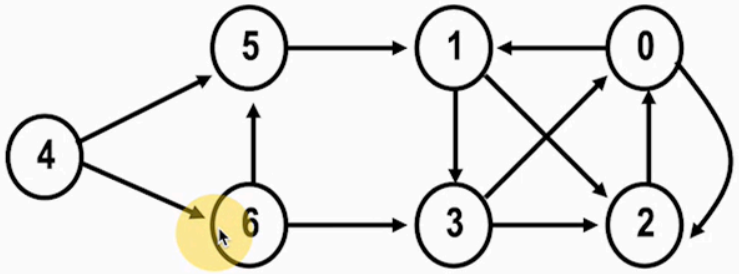
Graph Traversal

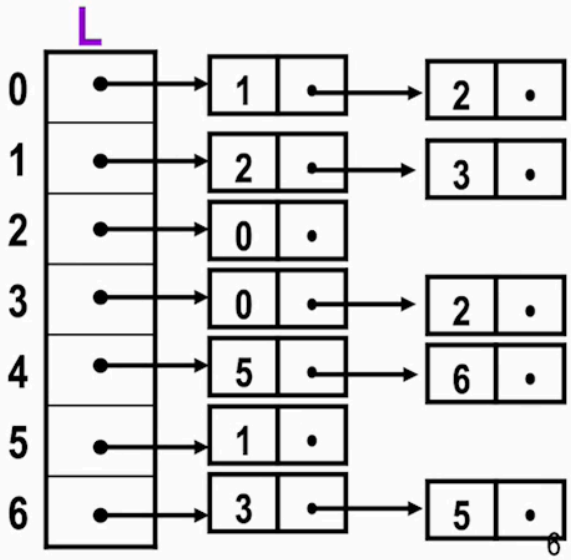
Depth First Search (DFS)

- a systematic way of visiting vertices and arcs
- This technique is called **depth-first search** because it continues searching in the forward (deeper) direction as long as possible
- a generalization of Preorder Traversals

Recursive dfs()

```
function dfs(vertex v)
{
    vertex w;
    mark[v] = visited;
    for each vertex w on L[v] do
        if (mark[w] == unvisited)
            dfs(w)
}
```





mark

U	U	U	U	U	U	U
0	1	2	3	4	5	6

Simulation
(1)

L[v]
0: 1,2
1: 2,3
2: 0
3: 0,2
4: 5,6
5: 1
6: 3, 5

```
function dfs(vertex v)
{
    vertex w;
    mark[v] = visited;
    for each vertex w on L[v] do
        if (mark[w] == unvisited)
            dfs(w)
}
```

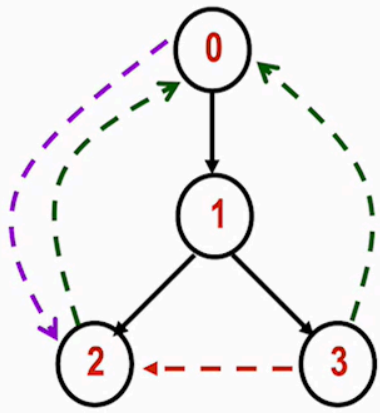
mark	V	V	V	V	U	U	U
	0	1	2	3	4	5	6

dfs(0)
v = 0
w_1 = 1 → dfs(1)
w_2 = 2
End of dfs(0)

dfs(1)
v = 1
w_1 = 2 → dfs(2)
w_2 = 3
End of dfs(1)

dfs(2)
v = 2
w_1 = 0
End of dfs(2)

dfs(3)
v = 3
w_1 = 0
w_2 = 2
End of dfs(3)



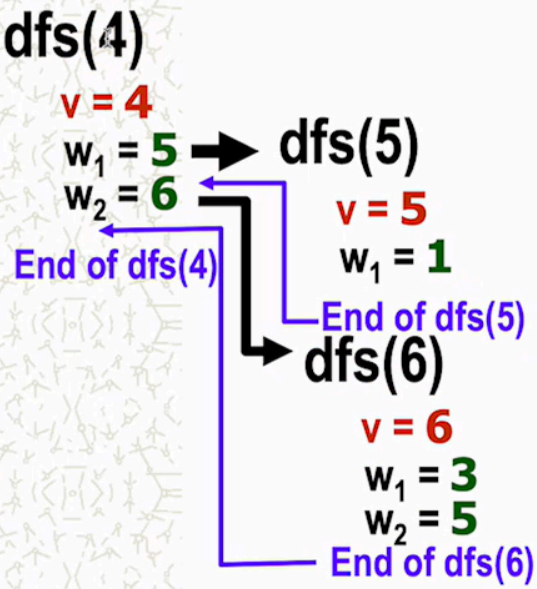
Not all vertices are visited!! Call dfs again!!

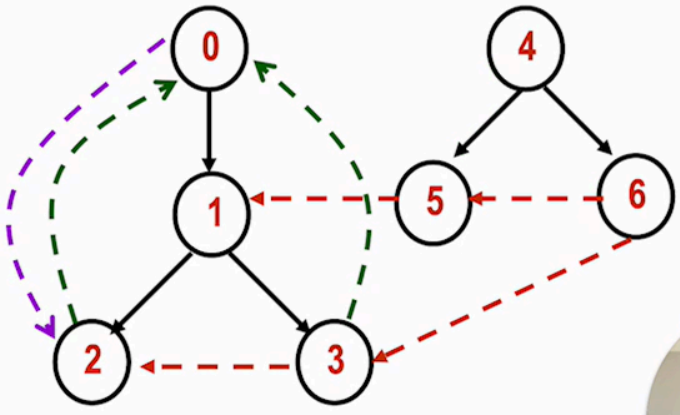
Simulation
(2)

L[v]
0: 1,2
1: 2,3
2: 0
3: 0,2
4: 5,6
5: 1
6: 3, 5

```
function dfs(vertex v)
{
    vertex w;
    mark[v] = visited;
    for each vertex w on L[v] do
        if (mark[w] == unvisited)
            dfs(w)
}
```

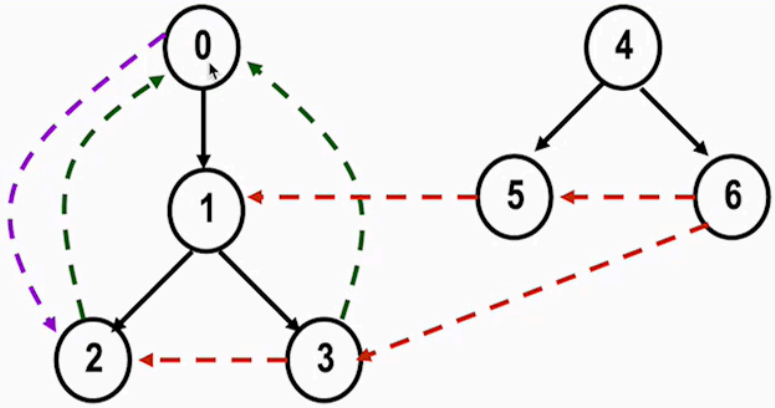
mark	V	V	V	V	V	V	V
	0	1	2	3	4	5	6





Depth-First Search Spanning Forest

- The arcs leading to new unvisited vertices are called **tree arcs.**
- Tree arcs form a **depth-first spanning forest** for the given digraph.

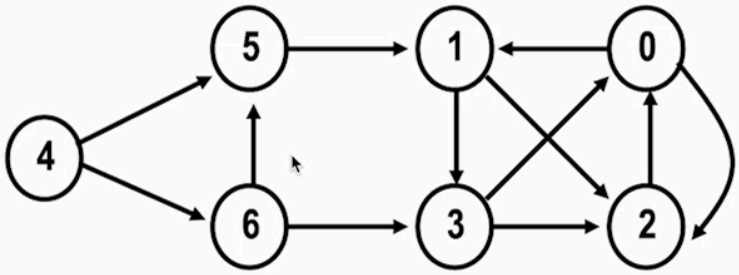


Other types of Arcs:

1. **Back arc** is a non-tree arc which goes from a vertex to one of its ancestors in the spanning forest. **Example**: $3 \rightarrow 0$
2. **Forward arc** is a non-tree arc which goes from a vertex to a proper descendant in the spanning forest. **Example**: $0 \rightarrow 2$
3. **Cross arc** is a non-tree arc which goes from a vertex to another vertex that is neither an ancestor nor a descendant.

   Examples: $3 \rightarrow 2$ and $5 \rightarrow 1$

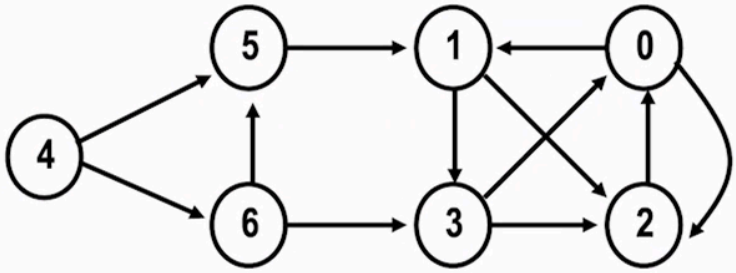
Exercise 2



A. Given the same graph and Adjacency List L, simulate **dfs**() and draw the **dfs** spanning forest and non-tree arcs starting with vertex 4, i.e with function call: **dfs(4)**

Note: If the function call will terminate and not all vertices are visited, call **dfs**() again using the remaining vertex with the lowest value.

Exercise 2 (Cont.)



B. Simulate function **dfs**() and draw the **dfs spanning forest** starting with **dfs(4)** using an Adjacency list L where in the adjacent vertices are arranged in descending order.

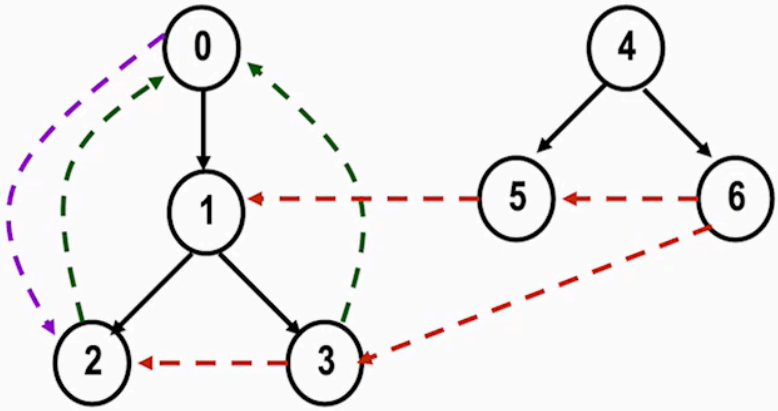
Depth-First Number

A **depth-first number**

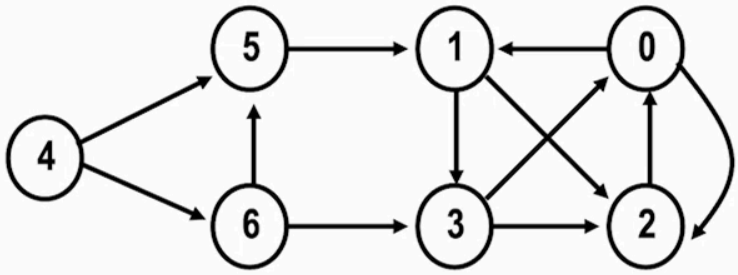
- is a number assigned to a vertex
- according to the order
- in which the vertex was first marked
- visited during a depth-first search

dfnumber

1	2	3	4	5	6	7
0	1	2	3	4	5	6



Exercise 3

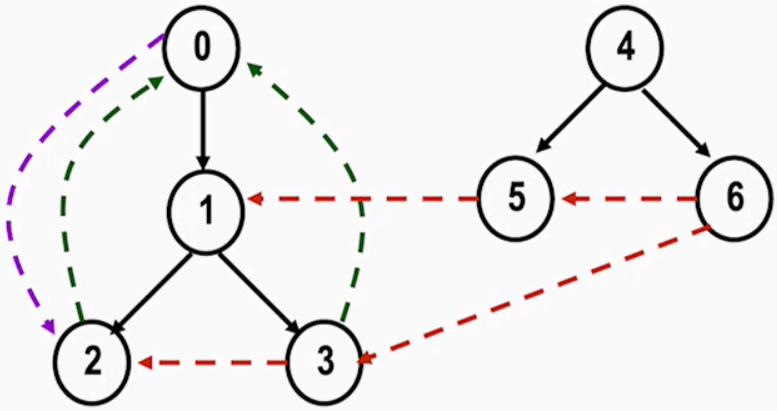


Determine the depth-first search number of each vertex in the dfs(4), i.e. based on Exercise 2A.

Observations

dfnumber

1	2	3	4	5	6	7
0	1	2	3	4	5	6



Observations:

1. w is a descendant of v iff (if and only if):
   dfnumber(v) ≤ dfnumber(w) ≤ dfnumber(v) + number of descendants of v
2. Forward arcs go from low-numbered to high-numbered vertices.
3. Back arcs go from high numbered to low-numbered vertices.
4. Cross arcs go from high numbered vertices to low-numbered vertices