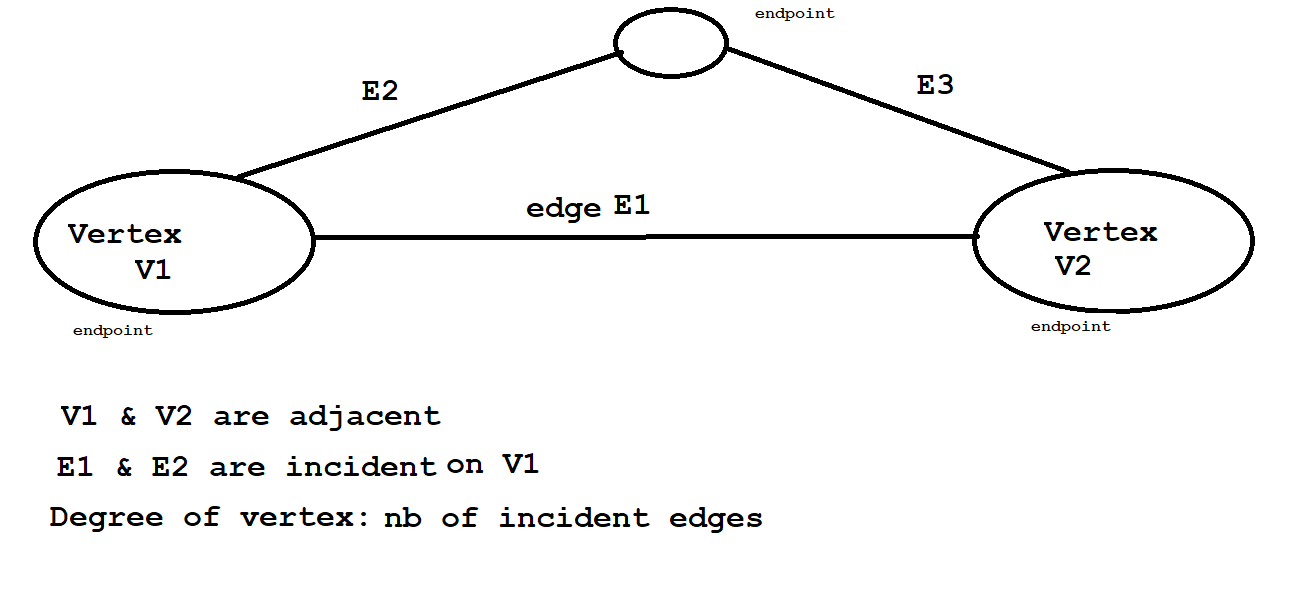
Conclusion 12

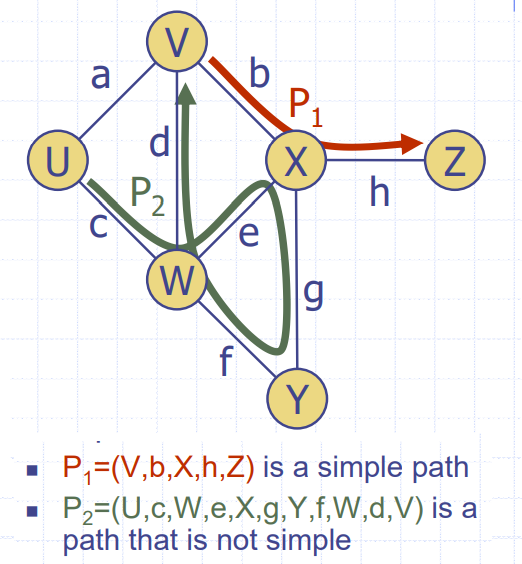
Graph, Depth-First Search

1. Graph :

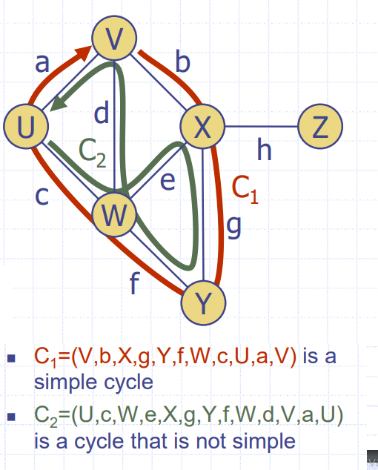
* Basics



* Simple and not simple path (simple: all its vertices and edges are distinct)



* Simple and not simple Cycle:



* A path in a graph is a sequence of alternating vertices and edges, starting with a vertex and ending with a vertex. A path is simple if all its vertices and edges are distinct.
* Properties

n number of vertices

m number of edges

deg(v) degree of vertex v

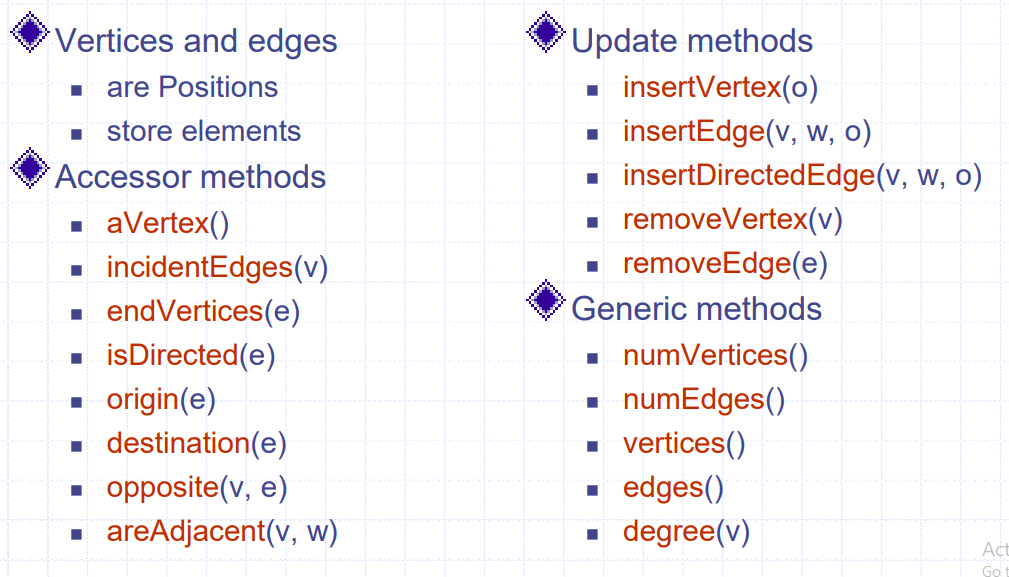
Σ deg(v) = 2m

m ≤ n (n − 1)/2 //direct graph = x2

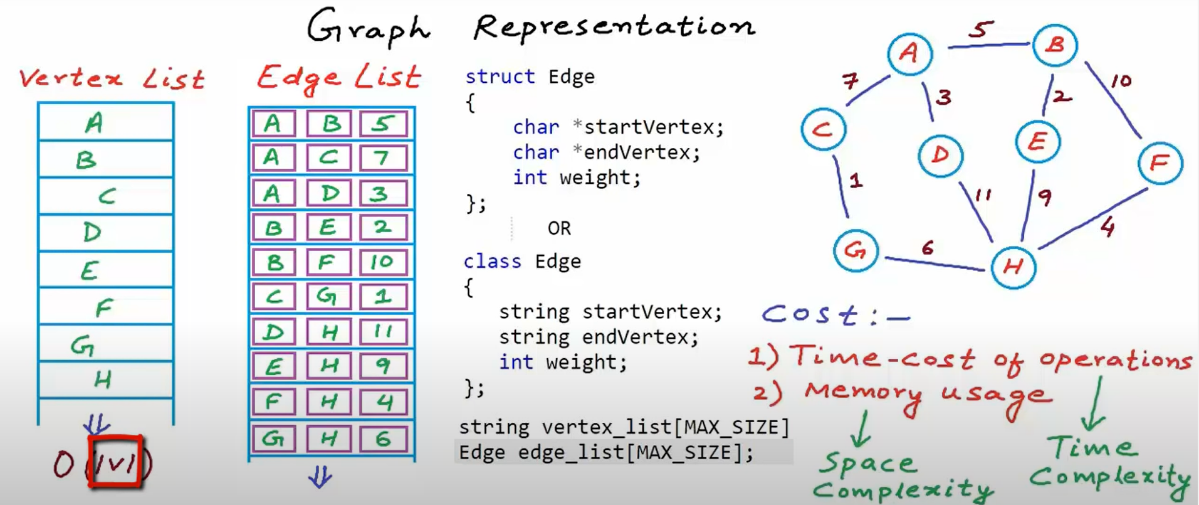
spanning tree m=n-1 (connected no cycle)

not connected m=n-2 (can be cycle)

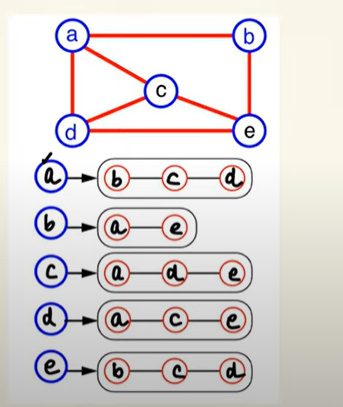
ADT



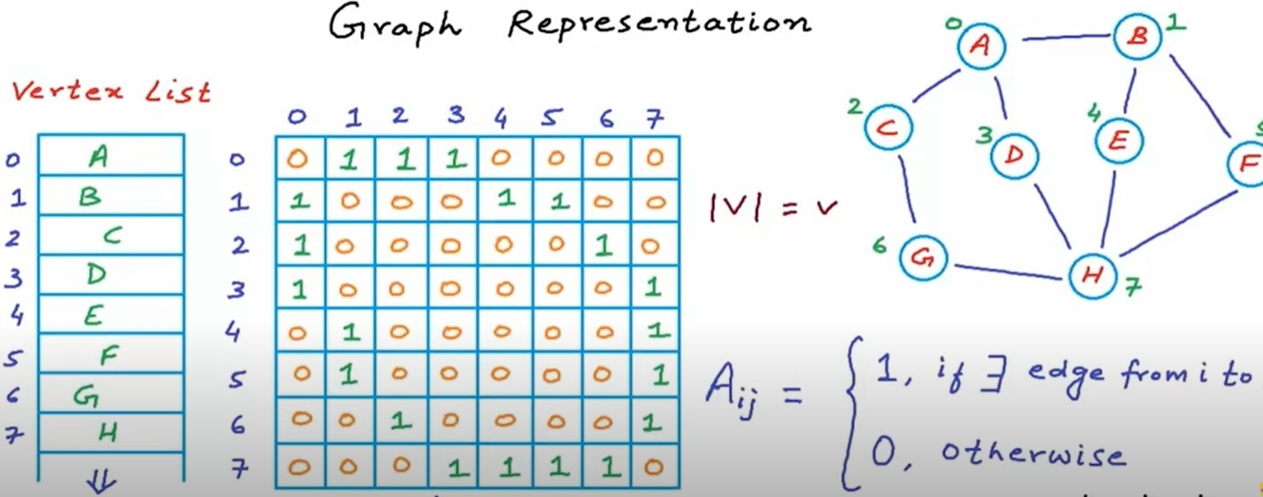
* Edge List Structure:

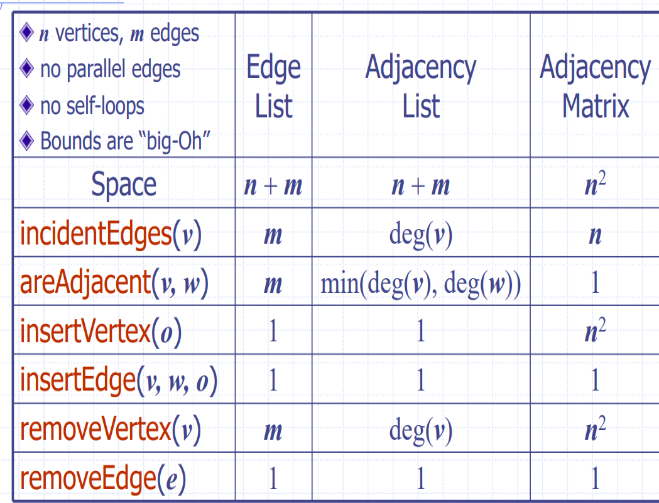
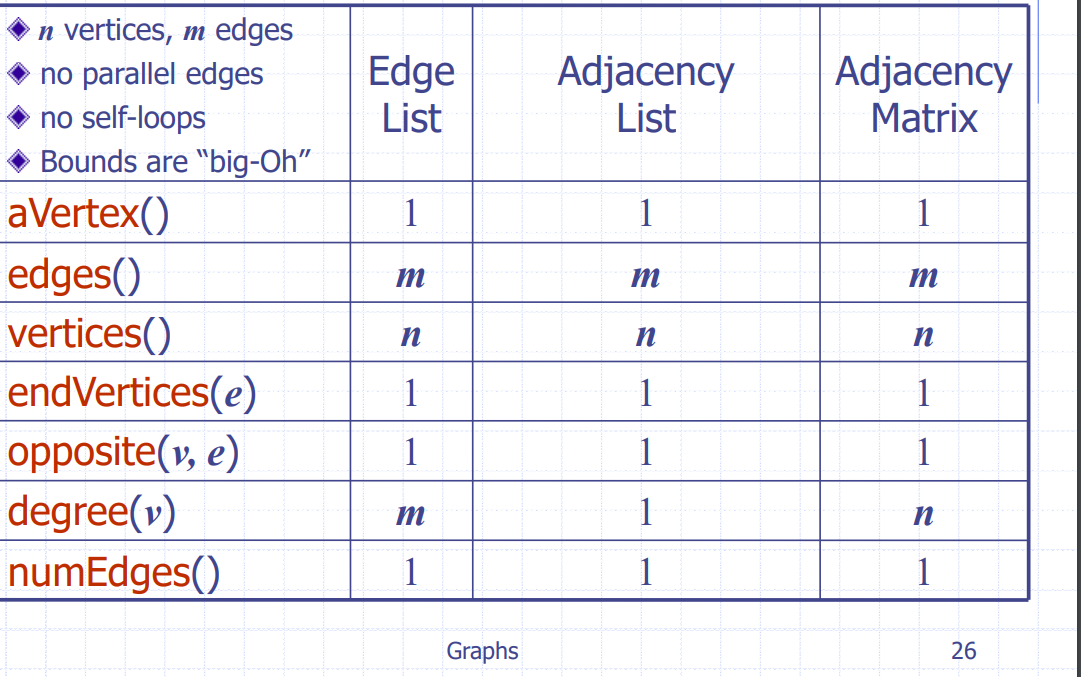


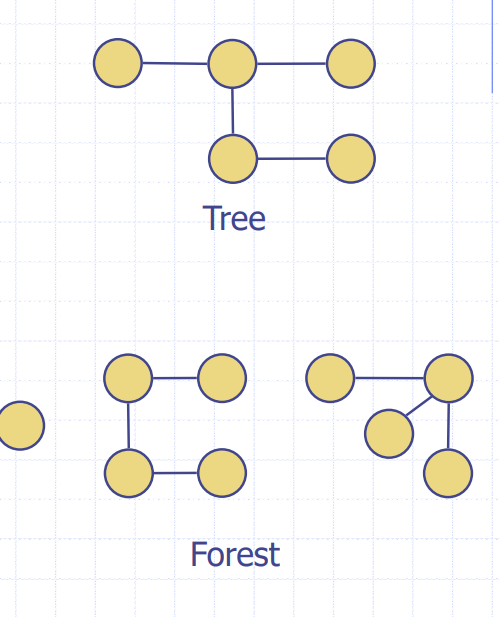
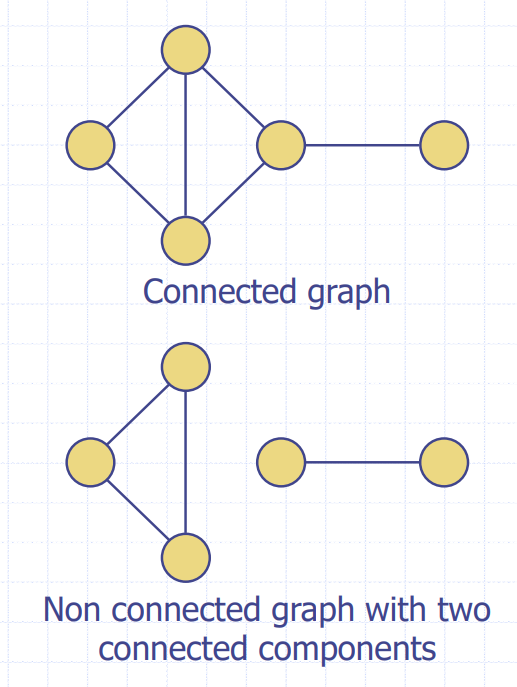
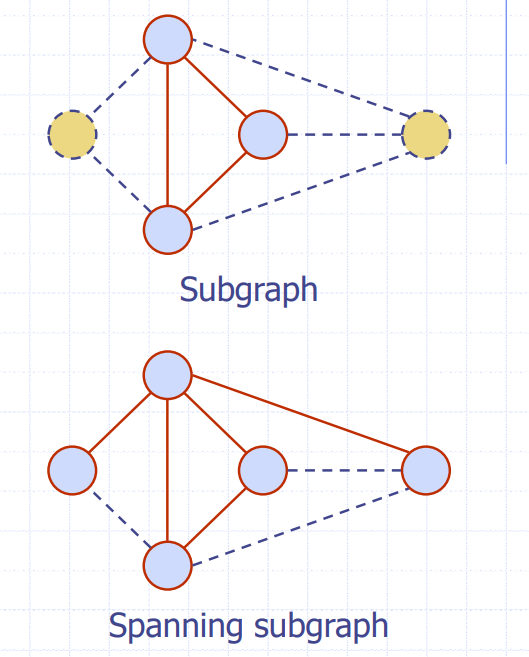
* Adjacency List Structure

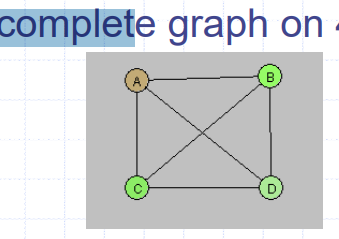
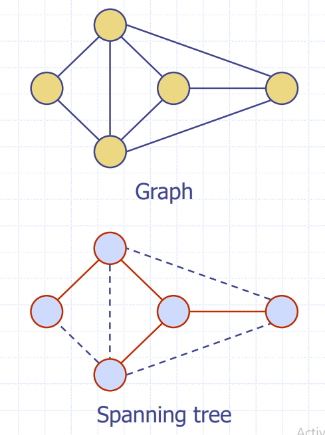
****

Adjacency List Structure:

****

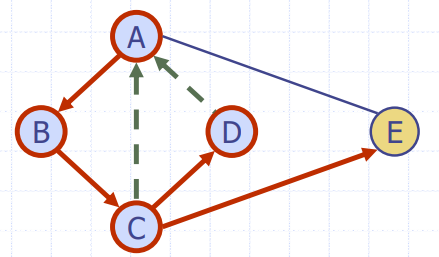
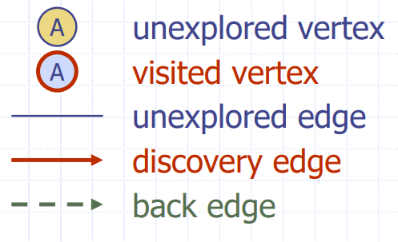
** **

****

****

* A spanning tree connects all vertices of a graph without any cycles. A spanning forest is a subgraph in which each connected component is a spanning tree of the vertices in that component.

1. Depth-First Search

****

* Algorithm:

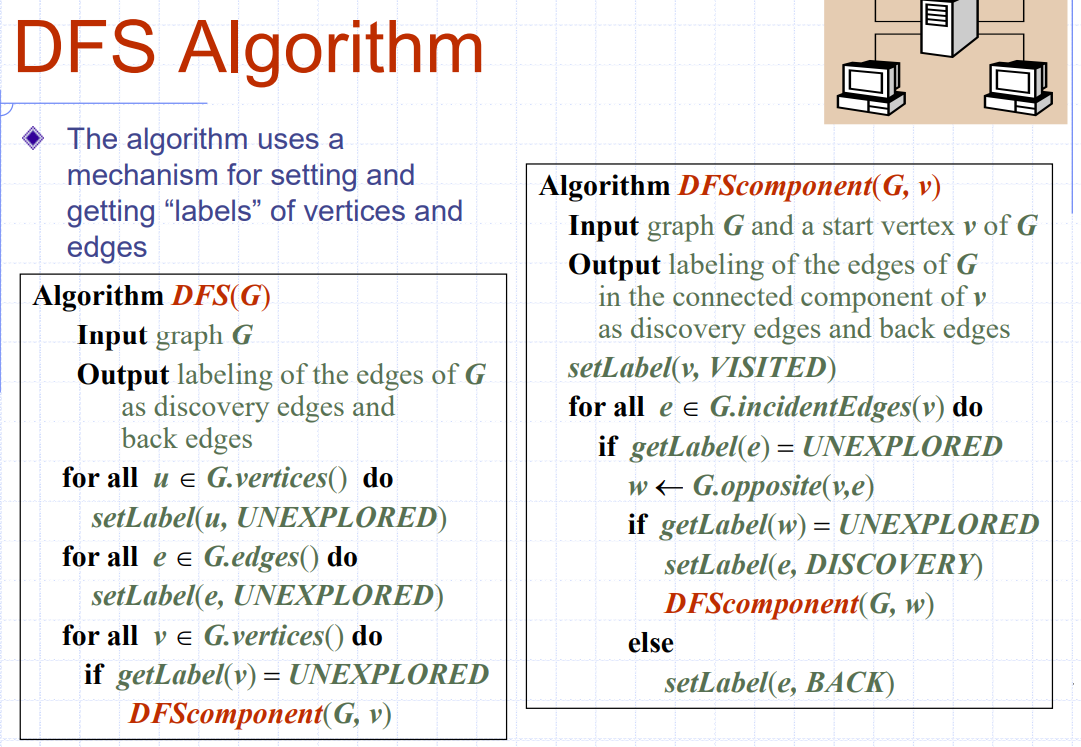
A DFS traversal of a graph G

◼ Visits all the vertices and edges of G

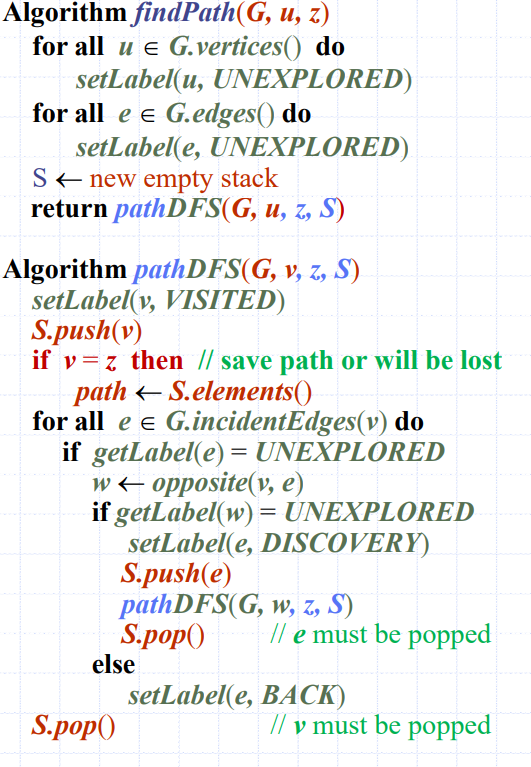
◼ Determines whether G is connected

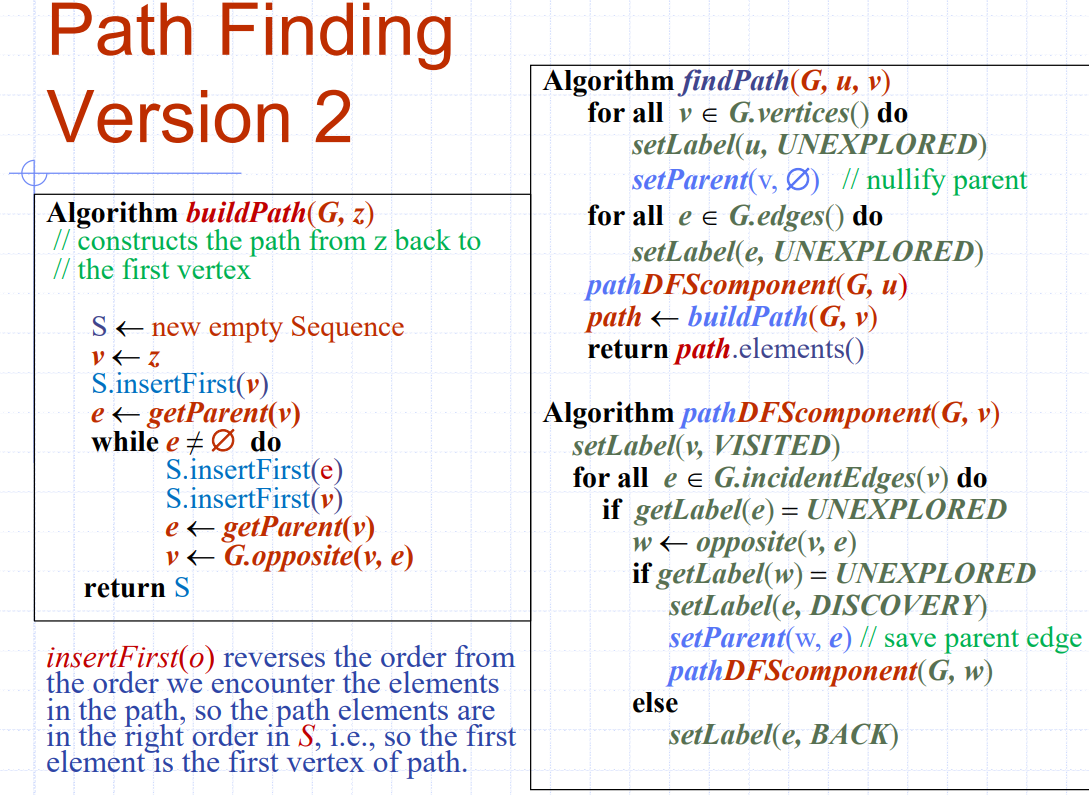
◼ Computes the connected components of G

◼ Computes a spanning forest of G

****

* Finding a Path using DFS

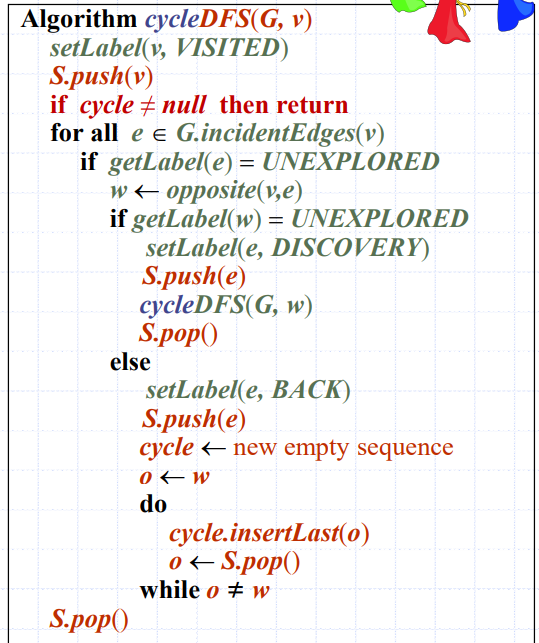
****

****

* Cycle Finding

We can use the DFS algorithm to find a simple cycle

We use a stack S to keep track of the path between the start vertex and the current vertex

****