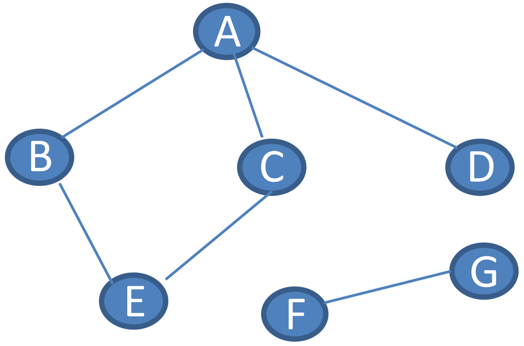
COMP20007 Workshop Week 6

1	**Topic 1: More on DFS and Topological Sort, Q 6.1**
2	**Topic 2: the Prim's Algorithm for MST, a quick exercise**
3	**Topic 3:** Binary Trees & BST ***Group/Individual Exercises:*** Q 6.2, 6.4, 6.3 **Lab:** • Review for MST and ask questions, or/and • Implement BST

DFS revisited: stack mechanism of recursion, time complexity

```
function DFS(G=(V,E))
  for each v in V do
    mark v with 0
  for each v in V do
    if v is marked with 0 then
      DfsExplore(v)

function DfsExplore(v)
  // implicit push(v) at the start
  // start visiting v
  mark v with 1
  for each edge (v,w) in E do
    if w is marked with 0 then
      DfsExplore(w)
  // end visiting v
  // implicit pop out v at the end
```



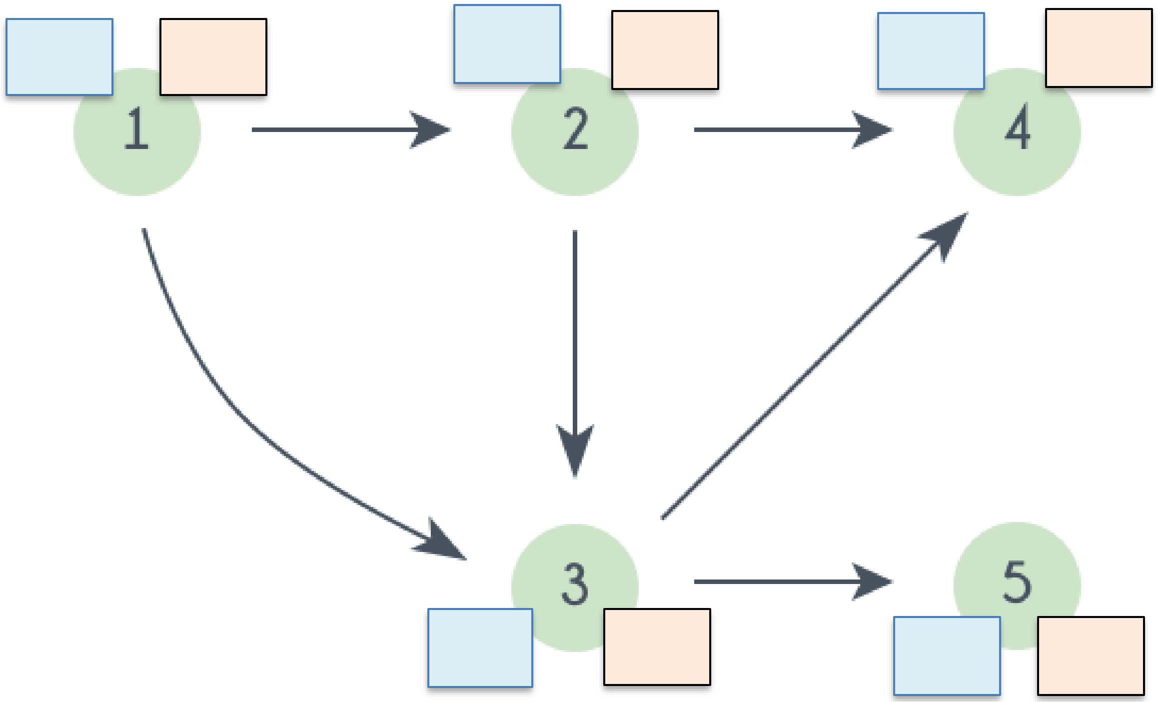
Work out Time Complexity

- for adjacency lists
- for adjacency matrix

DFS exercise: push- and pop-order (aka. pre- and post-order)

Problem: *For the graph below, write the push and pop order for DFS, starting from node 1*

Method 1: *Fill in the timestamp in blue boxes for push-orders, pink boxes for pop-orders.*	***Method 2:*** *Show the stack content and operations, then collect the push and pop order.*



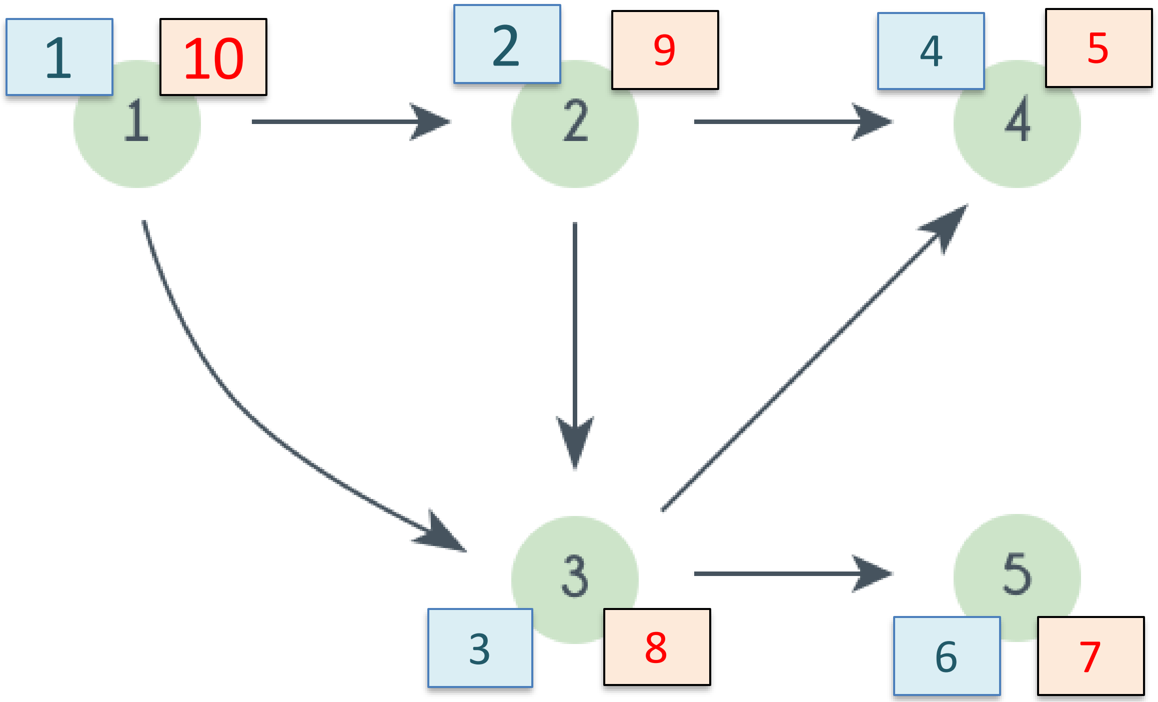
The PUSH and POP operations, using {a, b, c} to denote a stack of 3 elements where c is at the top of the stack.

ops **stack content**
init {}
push(1) {1}
...

Check soln DFS exercise: push- and pop-order (pre- and post-order)

Problem: *For the graph below, write the push and pop order for DFS, starting from node 1*

Method 1: Fill in the timestamp in yellow boxes for push-orders, pink boxes for pop-orders.	*Method 2:* Show the stack content and operations. Then collect the push and pop order.

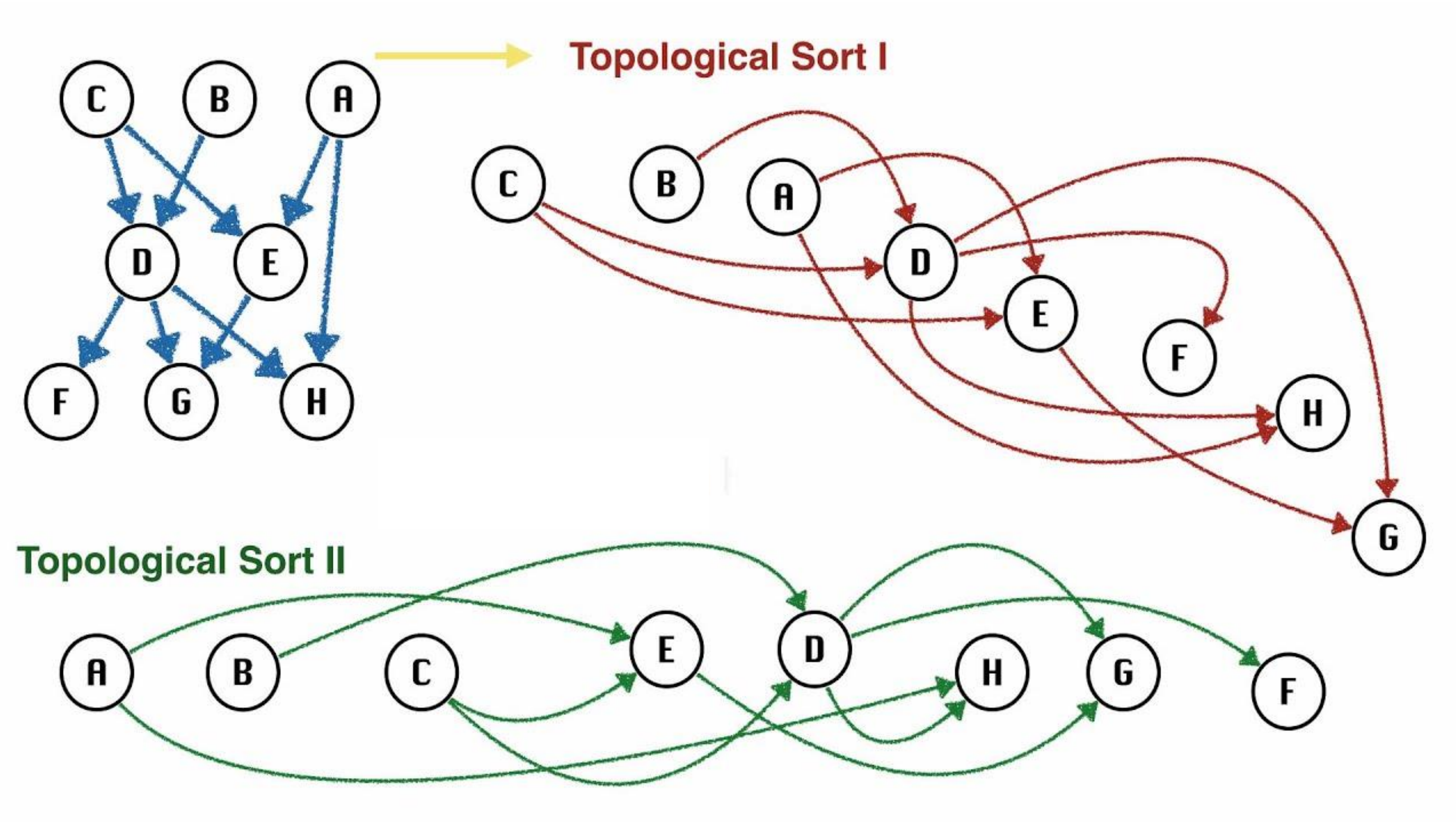


ops	**stack content**
init	{}
push(1)	{1}
push(2)	{1,2}
push(3)	{1,2,3}
push(4)	{1,2,3,4}
pop 4	{1,2,3}
push(5)	{1,2,3,5}
pop 5	{1,2,3}
pop 3	{1,2}
pop 2	{1}
pop 1	{}

push order: 1,2,3,4,5
pop order: 4,5,3,2,1

Topological Sorting (for DAG only!)

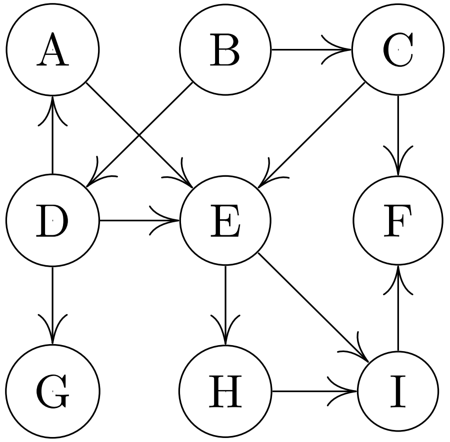
A *topological ordering:* sorting the nodes of the graph such that all edges point in one direction, to nodes later in the ordering.



Group Work: Q 6.1

Q6.1: *One of the algorithms discussed in lectures involves running a DFS on the DAG and keeping track of the order in which the vertices are popped from the stack. The topological ordering will be the reverse of this pop order.*

➔ *Finding a topological order for the graph by running a DFS.*



operation	content	operation	content
init stack	{}		
push(A)	{A}		
…			

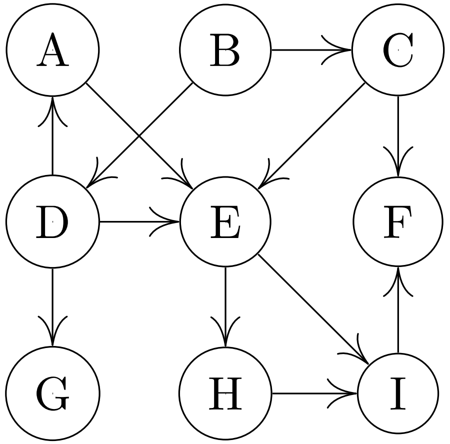
Notes: This algorithm for topological sorting is correct for the recursive implementation of DFS. Some variants of stack implementation of DFS might result in a different pop order (unlikely usable for topological sort)

The topological order resulted from the above DFS run =
???

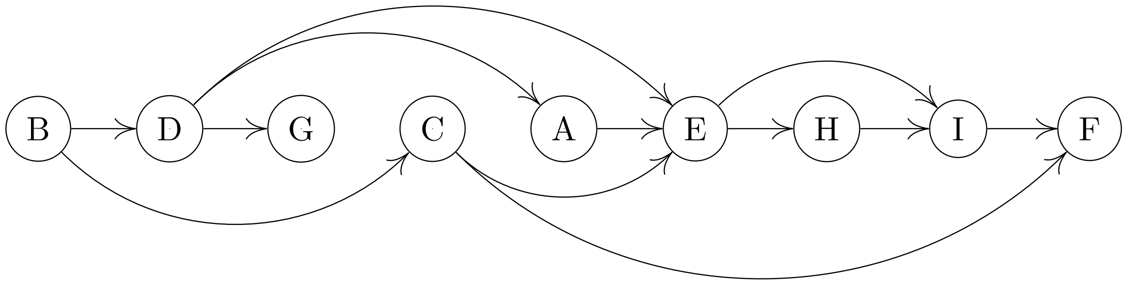
Check Soln: Q6.1

***T1:** Finding a topological order for the graph by running a DFS.*

operation	stack_content	operation	stack_ content
init stack	{}	push (B)	{B}
push (A)	{A}	push (C)	{B,C}
push (E)	{A,E}	pop C	{B}
push (H)	{A,E,H}	push (D)	{B,D}
push (I)	{A,E,H,I}	push (G)	{B,D,G}
push (F)	{A,E,H,I,F}	pop G	{B,D}
pop F	{A,E,H,I}	pop D	{B}
pop I	{A,E,H}	pop B	{}
pop H	{A,E}		
pop E	{A}		
pop A	{}		



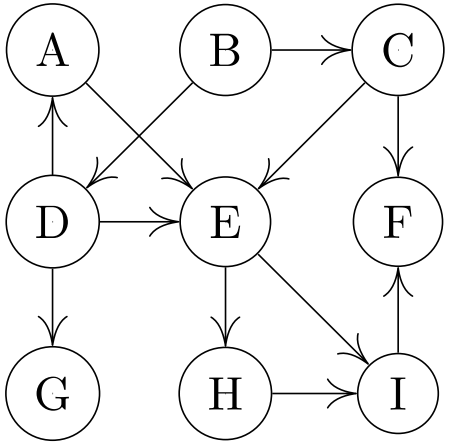
The topological order resulted from the above DFS run (= reversed pop order): B D G C A E H I F , you can check by re-draw the graph in the above linear order.



Notes:

- Alternative to showing stack content, we can run DFS manually and mark post order with time stamp

Q 6.1: Discuss other methods for toposort?



Additional Concept for di-graph nodes:

- a *source node* is a node that have no incoming edge,
- a *sink node* is a node that have no outgoing edge.

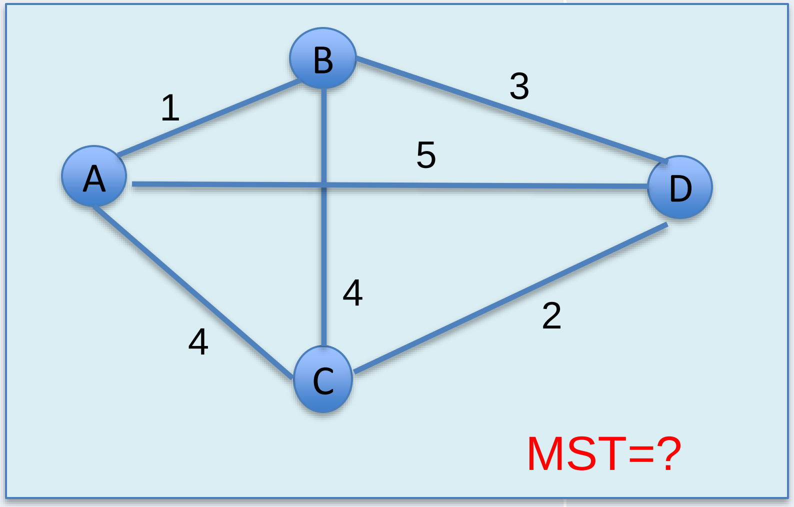
Describe Alternative method(s) *for Finding a topological order:*

Prim’s Algorithm vs Dijkstra’s Algorithm. Discuss concepts

	Prim’s	Dijkstra’s
Aim	find a MST	find SSSP from a vertex s
Applied to	connected weighted graphs with weights ≥ 0	weighted graphs with weights ≥ 0
Works on directed graphs?	?	
Works on unweighted graph?	?	

Related concepts for Prim’s

- spanning trees = ?
- MST = ?
- is MST unique?



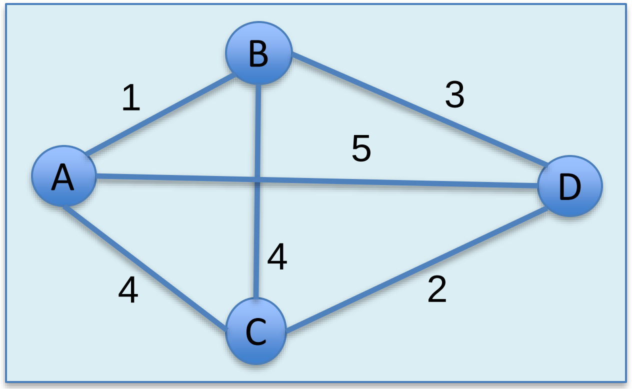
Dijkstra(G=(V,E),S) versus	Prim(G=(V,E))
Task: Find SSSP from S (that involves all nodes of a *connected* graph)	**Task: MST (that involves all nodes of a *connected* graph)**

for each v ∈V do
  cost[v]:= ∞
  prev[v]:= nil
 	
cost[S]= 0
PQ:= create_priority_queue(V,cost) with cost[v] as priority of v∈V
	for each v ∈V do
	  cost[v]:= ∞
	  prev[v]:= nil
	pick initial S
	cost[S]:=0
	PQ:= create_priority_queue(V, cost) with cost[v] as priority of v∈V

while (PQ is not empty) do u := ejectMin(PQ)	while (PQ is not empty) do u := ejectMin(PQ)

for each neighbour v of u do
 	
  if **cost[u] + w(u,v)** < cost[v] then
   cost[v]:= **cost[u] + w(u,v)**
   update (v, cost[v]) in PQ
   prev[v]:= u
	for each neighbour v of u do
	
	  if **w(u,v)** < cost[v] then
	   cost[v]:= **w(u,v)**
	   update (v, cost[v]) in PQ
	   prev[v]:= u

Aim to minimise the distance from S to any node, that's why **cost[u] + w(u,v)** is considered	Aim to minimise the contribution of individual edge weight to the MST, that's why only **w(u,v)** is considered



Running Prim's Algorithm to find a MST

At each step, we add a node to MST.
We choose the node with **minimal edge cost**.
On paper: We start with A according to the alphabetical order.

step	node added to MST	A	B	C	D
0		0,nil	∞,nil	∞,nil	∞,nil
1					
2					
3					
4					

Still unsure how to trace the Dijkstra and Prim Algorithms?

- see pages 25-28 of last week workshop slides for details about tracing Dijkstra
- Do Q5.8 (question 8 last week) and compare with solution for the Prim algorithm

Peer Activity: Adapting Prim's Algorithm

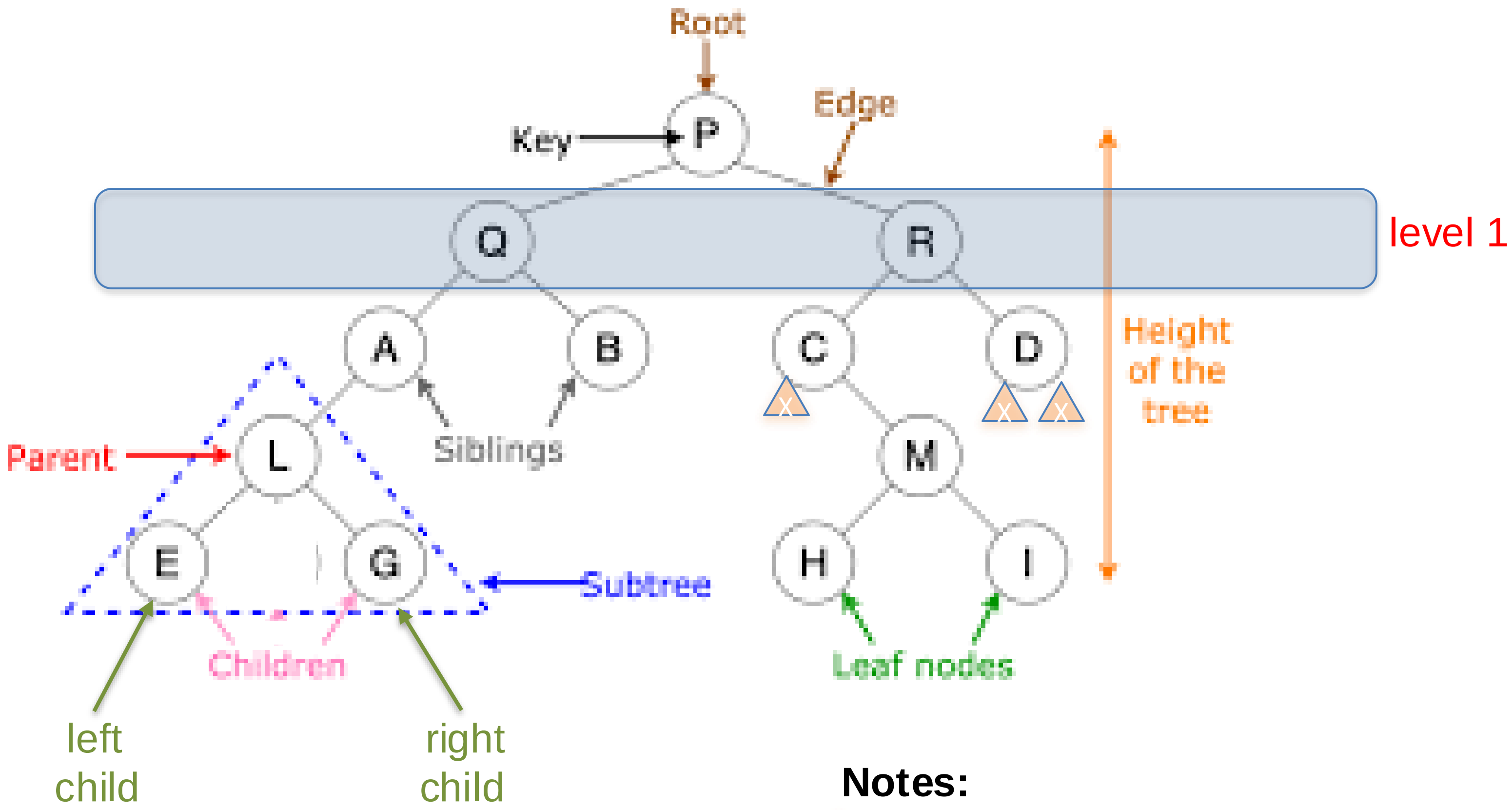
Can Prim's algorithm be modified to meet these requirements?

a. Yes, if infinite-weight 'light' edges are accounted for.
b. Yes, if it is repeatedly run on non-MST vertices until exhaustion.
c. No, prior component analysis is required.
d. No, it fails to halt on unconnected graphs.

We want **an algorithm** that:

- works on both connected and unconnected graphs
- finds a set of MSTs spanning each connected component
- doesn't require separate component analysis before its commencement

Binary Trees as Special Graphs



Notes:

 denotes a NULL node, aka. *external node*, only a few of them drawn here. If the tree has n internal (non-NULL) nodes, it has n+1 external nodes

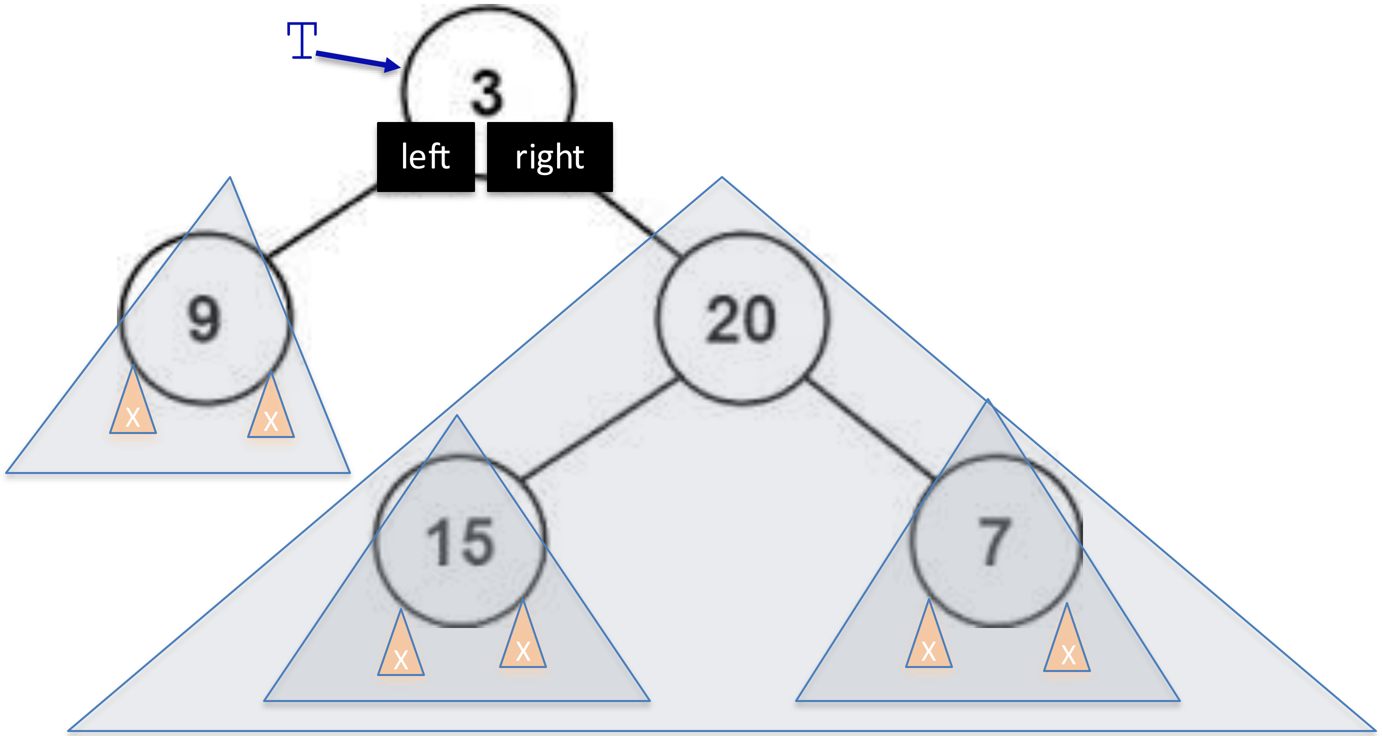
Make differences between:

- *leaf nodes* and *external nodes*
- *none-leaf nodes* and *internal nodes*

Binary Tree: Recursive Definition

A binary tree T is:

- NULL, or
- a node T, called the tree's *root node*, that contains:
  - some data,
  - a link to another binary tree called the root's *left child*, and
  - a link to another binary tree called the root's *right child*



***Side note:** a non-empty tree is fully defined by its root. For pseudocode simplicity we use:*

tree T = its root node

= pointer to its root node.

For convenience, we also use T.key *(or* T.data*),* T.left, T.right *for the* data, left*, and* right *childrens*

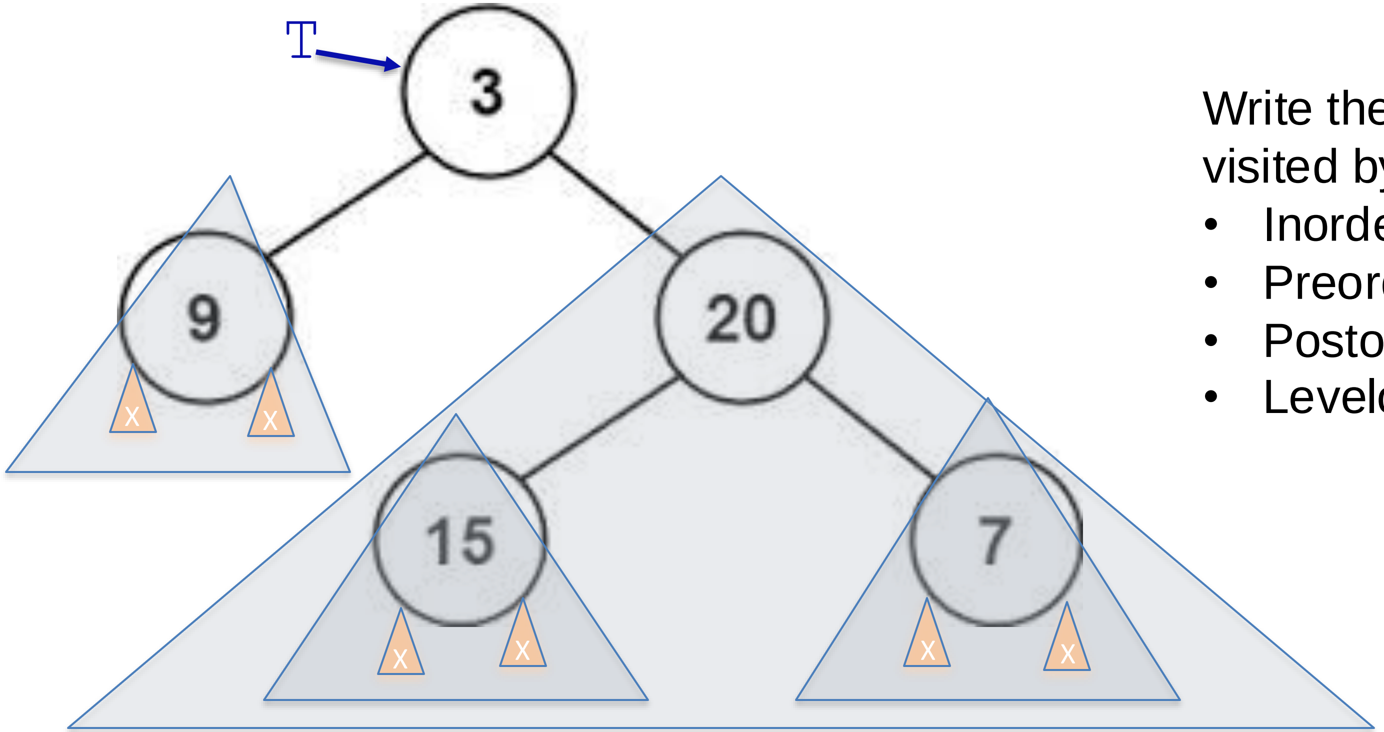
Binary tree traversal

Understand:

tree traversal?

inorder, *preorder*, *postorder* traversal? Are they BFS or DFS?

What is *level-order* traversal? Is it BFS or DFS?

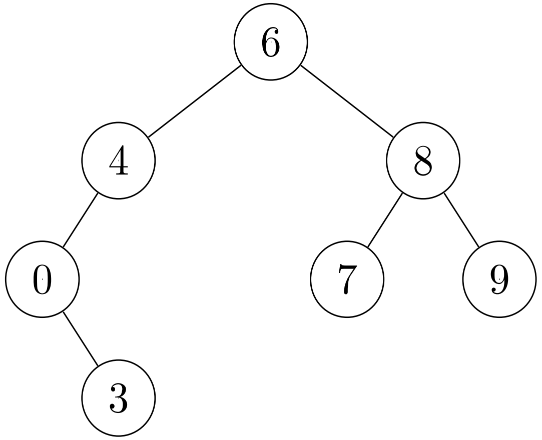


Write the nodes in order visited by:

- Inorder :
- Preorder :
- Postorder :
- Levelorder:

Q 6.4: Binary Tree Sum

Write an algorithm to calculate the sum of a binary tree where each node contains a number.



YOUR ANSWER: The pseudocode:

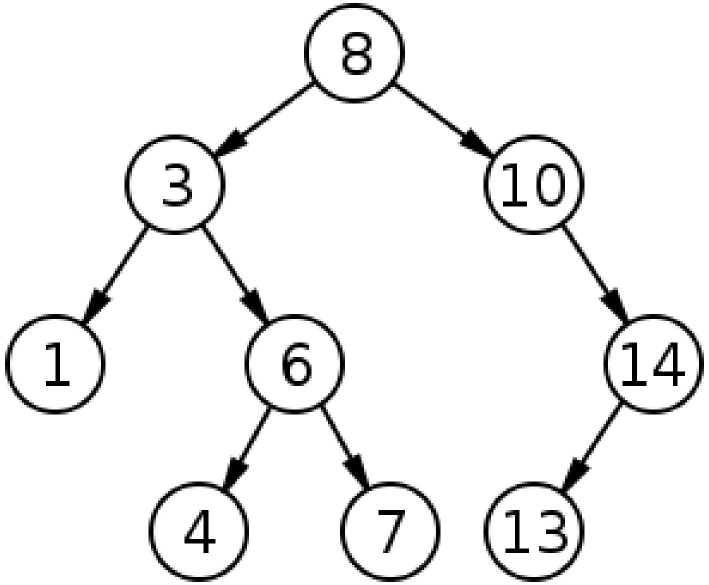
function ???

Binary Search Tree

Review: What's a BST?

How to:
1. print the keys in increasing order?
2. print in decreasing order?
3. copy the tree?
4. free the tree?

Your answers:
1.
2.
3.
4.

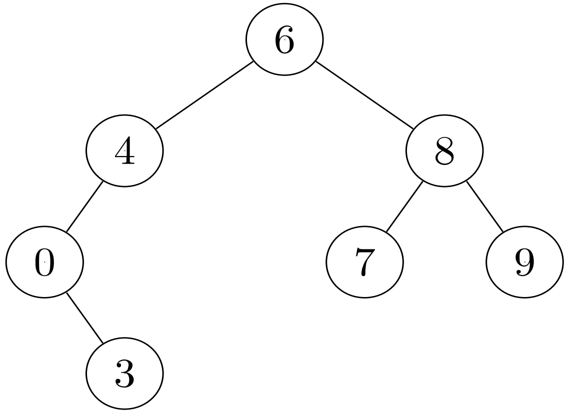


Small Exercise:

- Write a function for searching in BST
- What's the complexity of that function?

Q 6.2: conventional traversal

Write the *inorder*, *preorder* and *postorder* traversals of the following binary tree:



YOUR ANSWER:

In-order:

Pre-order:

Post-order:

Q 6.3: level-order traversal

Level-order: visit level-by-level, left-to-right, starting from the root (which is in 0-th level).

a) For the tree, what's the visited order?

b) Write the level-order pseudo-code.

YOUR ANSWER:

a) Level-order:

b) pseudocode

```
// level order traversal for binary tree T
function LevelOrder(T)
  ?
```

Some topics in the Lectures Week 1-5 (as I remember, likely incomplete)

Problem Solving Techniques

- Brute Force
- Divide-and-Conquer
- Reduce-and-Conquer
- Transform-and-Conquer

Complexity

- Basic Operations and Complexity of Algorithms
- Complexity Classes, comparing complexity functions
- Complexity of Recursive Algorithms: building and solving recurrences
- The Master Theorem for solving recurrences of divide-and-conquer

Basic DS & ADT

- Arrays and Linked Lists
- Stacks & Queues

Graphs

- Concepts, Properties, Representation
- Traversal with DFS and BFS
- Complexity of graph algorithms
- Topological Sort
- Dijkstra's Algorithm for SSSP
- Prim's Algorithm for MST

Binary Trees

- Concepts, Properties, Representation
- DFS Traversal: in-order, pre-order, post-order
- BFS Traversal: level-order

LAB

- Q&A on not-yet-done problems (if any) of previous workshops/lab/lectures
- Q&A on MST

OR/AND:

- Do the BST insert exercise, aim to finish within 30 minutes a runnable version with:
  - data types
  - function insert
  - function printing keys in order
  - main(): a loop to read data and build th
- Questions on Dijkstra's and BFS, DFS, g

Big Advice for MSTs & Exams

(in addition to your "optimal strategy")

Give time to understand the question at hand and address all requirements of the question.

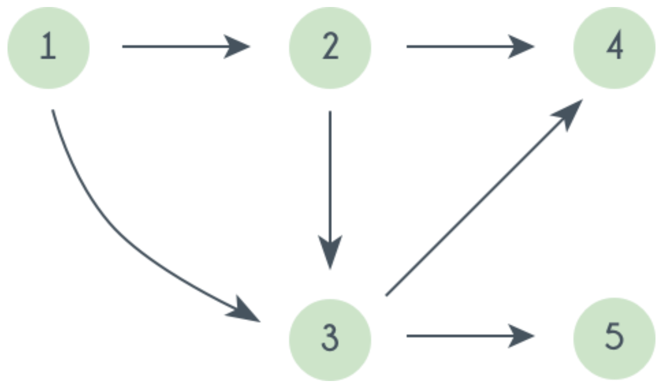
- Do Not answer a wrong question.
- Do Not miss a requirement if you can do.

Additional Slides for Convenience

DFS exercise: generating push- and pop-order

Problem: *Modify the DFS algorithm so that it also builds the arrays* `push[V]` *and* `pop[V]` *to store the push- and the pop-order of the vertices.*

Example graph:



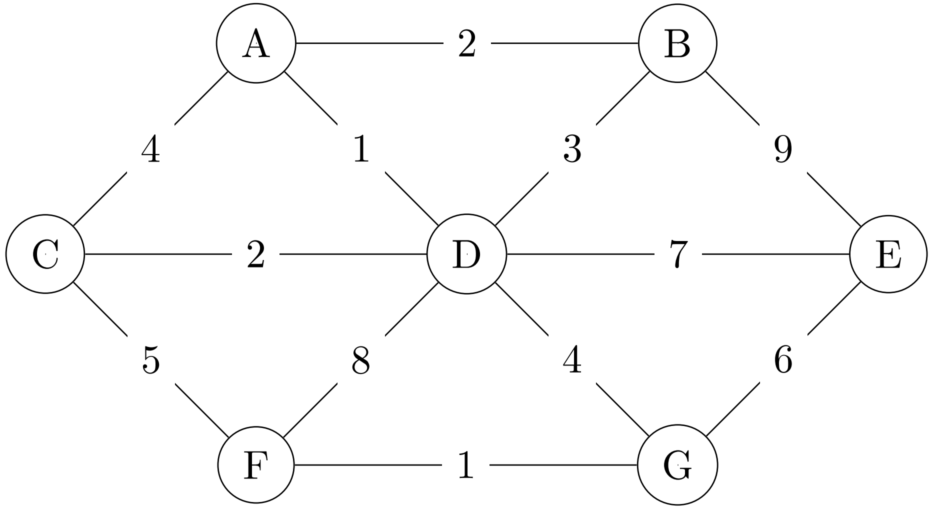

```
// building push[V] and pop[V]
function DFS(G=(V,E))
  for each v in V do
    mark v with 0
  for each v in V do
    if v is marked with 0 then
      DfsExplore(v)

function DfsExplore(v)
  mark v with 1
  for each edge (v,w) in E do
    if w is marked with 0 then
      DfsExplore(w)
```

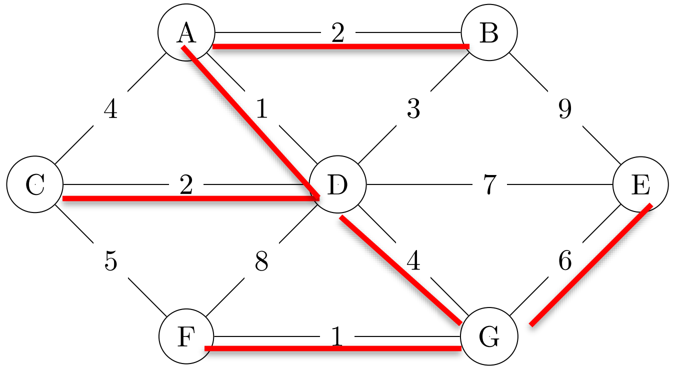
Question 5.8 (last week): Minimum Spanning Tree with Prim's Algorithm

Prim's algorithm finds a minimum spanning tree for a weighted graph. Discuss what is meant by the terms 'tree', 'spanning tree', and 'minimum spanning tree'.

Run Prim's algorithm on the graph below, using A as the starting node. What is the resulting minimum spanning tree for this graph? What is the cost of this minimum spanning tree?



step	node done	A	B	C	D	E	F	G
0		0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
1								
2								
3								
4								
5								
6								
7								



Check: Soln to Q5.8 : Tracing Prim's Alg

What's the resulted MST: all nodes and the red edges

What's the cost of that MST?

cost= **0**+**2**+**2**+**1**+**6**+**1**+**4**= 16

step	node ejected	A	B	C	D	E	F	G
0		**0/nil**	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
1	A		2,A	4,A	**1,A**	∞/nil	∞/nil	∞/nil
2	D		**2,A**	2,D		7,D	8,D	4,D
3	B			**2,D**		7,D	8,D	4,D
4	C					7,D	5,C	**4,D**
5	G					6,G	**1,G**	
6	F					**6,G**		
7	G							