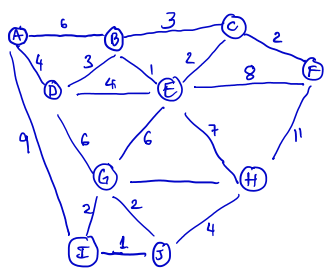
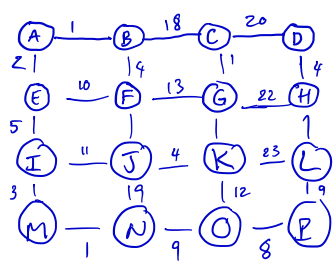
Practice M5

Friday, October 22, 2021 11:08 AM

Phuc Tran (Jennifer)
MSWE - Fall 2021
241R - Applied Data Structures & Algorithms
The Algorithms Design Manual - Second Edition
Homework: 5-1, 5-2, 5-31

5-1 For the following graphs G_1 (left) and G_2 (right):

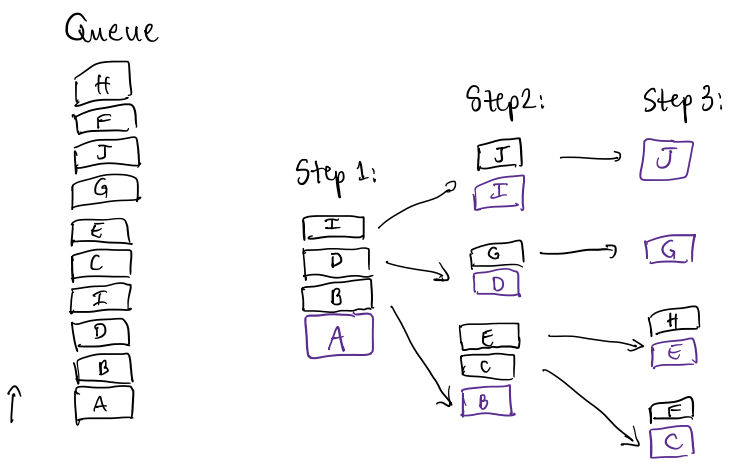




a) Report the order of the vertices encountered on a breadth-first search starting from vertex A. Break all ties by picking the vertices in alphabetical order (i.e. A before Z).

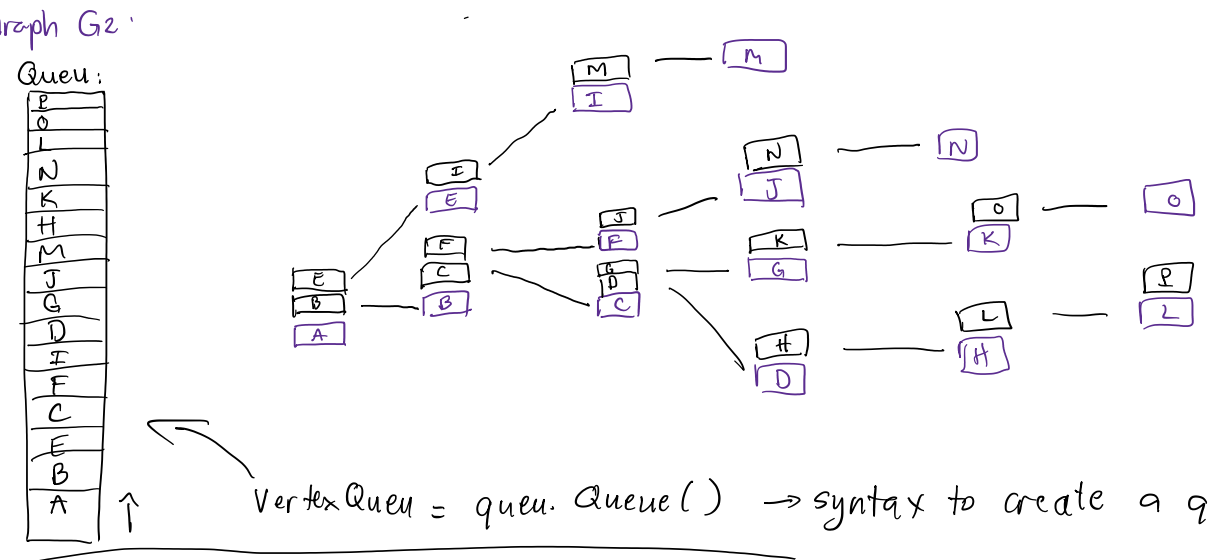
Breadth-first search: starts at a node → explores the neighbors first → move to next level neighbors
$O(V+E)$ = O(vertices + edges)

Graph G1:
Step 1: Start with node A
Visit node A's neighbors: B, D, I
Step 2: Start with one of node A's neighbors, choose node B.
Visit all node B's neighbors
Add all B's unvisited neighbor to the queue.
Repeat for other neighbors of A, which are D & I in this case
Step 3: Visit the unvisited neighbors of the node in step 2.



Ans: A, B, D, I, C, E, G, J, F, H

Graph G2:



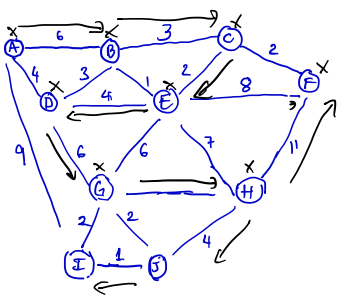
VertexQueu = queu.Queue() → syntax to create a queue class object in Python

Ans: A, B, E, C, F, I, D, G, J, M, H, K, N, L, O, P

b) Report the order of the vertices encountered on a depth-first search starting from vertex A. Break all ties by picking the vertices in alphabetical order.

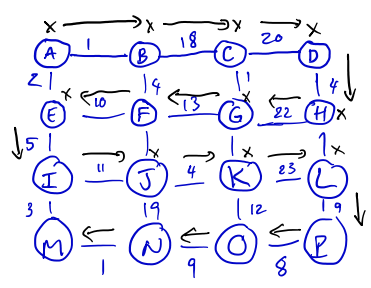
Notes: • O(vertices + edges)
• performs tasks such as count connected components, determine connectivity, find bridges / articulation points
• DFS plunges depth first into a graph w/o regard for which edge it takes next until it cannot go any further, at which point it backtracks and continues.

Graph G1:
A → B → C → E → D → G → H → F trace back to H → J → I



⇒ A, B, C, E, D, G, H, F, J, I

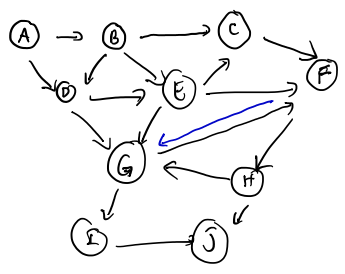
Graph 2:



A→B→C→D→H→G→F→E→I→J→K→L→P→O→N→M

A, B, C, D, H, G, F, E, I, J, K, L, P, O, N, M

5-2 Do a topological sort of the following graph G:



Method 1: Cannot sort b/c G→F→H→G is cyclic.
Directed Acyclic graphs only for top sort.
Method 2:
Reverse G→F to F→G to make the graph acyclic
This can be sorted as
A, B, D, E, C, F, H, G, I, J

TS: linear ordering of vertices such that for every directed edge (u, v), vertex u comes before v in the ordering.
Time complexity: O(vertices + edges)
TS is not unique,

5-31 Which data structures are used in depth-first & breadth-first search?

BFS: queu - first in first out - go adjacent nodes, then adjacent nodes of those
DFS: stack - first in last out - go all the way to the last node