Week 8

Assignments:
h7 available _____ and due before 10pm on after spring break
p3b available and due before 10pm on 3/28
x4 available and due before 10pm on 4/1
x5 available and due before 10pm on 4/1

Read: Week 8 (start on week and 9 for week after Spring Break)

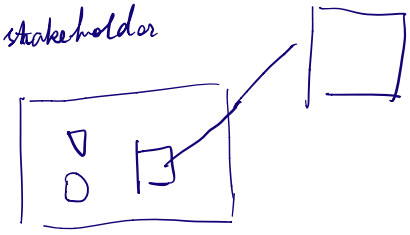
THIS WEEK (in addition to the Midterm Exam results):

- Graphs
  - edge representations
  - traversals
  - Spanning Trees
  - ~~topological ordering~~
  -

← x4 intro

TA will simulate stakeholder

Diagram home



- Exam Results (Tuesday or Wednesday in class)

NEXT WEEK (SPRING BREAK – No classes, lab or office hours)
Next WEEK (after Spring Break)

- Topological Ordering
- Dijkstra's Shortest Path
- Spanning Trees

Compute degree of a Vertex

→ Write the code to be added to a `Graph` class that computes the degree of a given node in an undirected graph.

* degree:
"out" degree for direction.

Adjacency list:

```
public int degree( Graphnode<T> n) {
    return n.getAdj().size();
}
```

O(1)

Adjacency matrix:

AM
AM.length // row
AM[i].length // col.

```
public int degree( Graphnode<T> n) {
    int degree = 0;
    int index = n.getAmIndex();
    for (int col = 0; col < AM[index].length; col++) {
        if (AM[index][col])
            degree++;
    }
    return degree;
}
```

O(1)
O(1)
O(V)

O(V)
↑ # of vertices

Comparison of Edge Representations

Ease of Implementation

* All are easy

Space *(memory)* V # vertices, E # edges

AM $O(V^2)$

AL $O(V+E)$

Time *(complexity of ops)*

Depend on implementation

node's degree?

AM $O(V)$ linear

AL $O(1)$ constant.

edge exists between two given nodes?

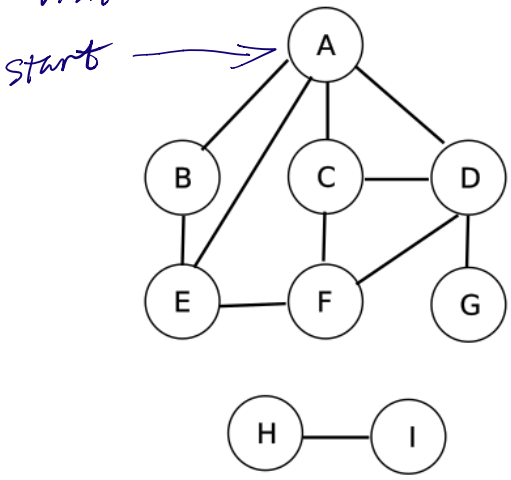
AM $O(1)$ constant AM[IndexOfFrom][IndexOfTo].

AL $O(N)$ linear ← must iterate through AL.
& adjacent nodes for vertex.

Searches and Traversals

Search Given a "start vertex"
look through a collection to find something
stop when found or reach end of selection

Traversal
visit each vertex exactly one
specify a "start" vertex
visit vertices that are reachable from start vertex.



connective component #1

connective component #2

Same graph.

→ Which connected component in the graph above can produce the longest path?

Problem: How do we avoid cycles?

Solution: traverse, mark visited vertices
and when reach a marked vertices, that's a cycle

****CS400 Convention: pick next unvisited vertex in increasing alphanumeric order**

Depth-First Search

Assume: all vertices are unvisited at start.

Relies on: stack, use call stack + recursion

recursive

Algorithm:

```
DFS(V)
    mark V as visited
    for each unvisited successor (s) of V
        DFS(s)
```

Equivalent to: pre order tree traversal.

Breadth-First Search

Assume: all vertices are unvisited at start.

Relies on: queue; need to creat queue

x recursive.

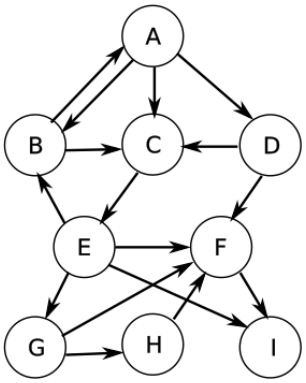
Algorithm:

```
BFS(V)
    q = new queue();
    mark V as visited first.
    q.enqueue(V)   // add to queue
    while (!queue.isEmpty())
        C = q.dequeue; // remove
        for each unvisited success(s) of C;
            mark as visited
            q.enqueue(s)
```

Equivalent to: level-order traversal

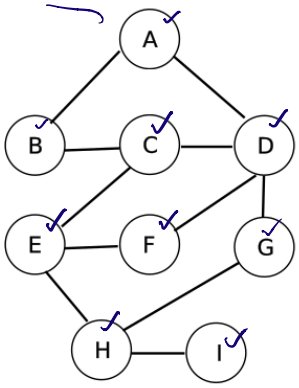
DFS Practice

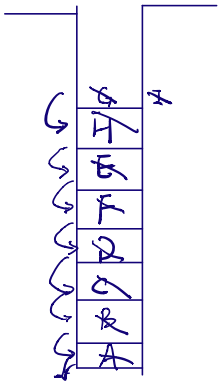
Give the order that vertexes are visited for depth-first search (DFS) starting at A for each graph



Graph 1:

start at A

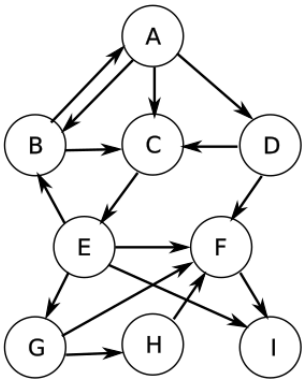




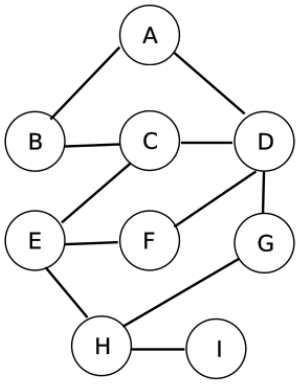
Graph 2:

A B C D F E H G I

Give the order that vertexes are visited for depth-first search (DFS) starting at C for each graph



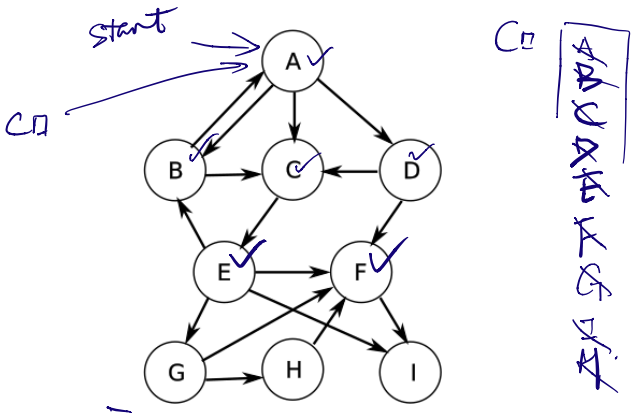
Graph 1:



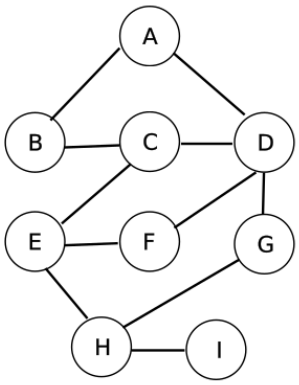
Graph 2:

BFS Practice

Give the order that vertexes are visited for breadth-first search (BFS) starting at A for each graph

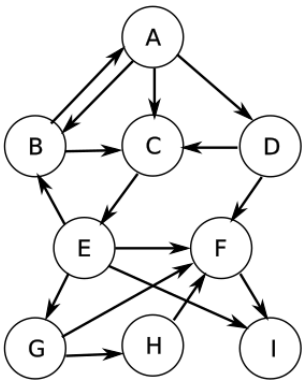


Graph 1:

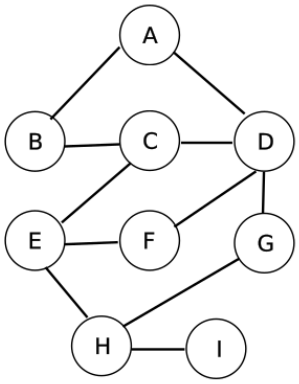


Graph 2:

Give the order that vertexes are visited for breadth-first search (BFS) starting at C for each graph



Graph 1:



Graph 2:

traversal. works.

Applications of DFS & BFS

- Is the graph connected – is one connected component.
- what vertices are reachable from start?

Path Detection

Is there a path from S (start) to T (target)?

what is the path. (DFS)

What is the shortest path? (fewer # edges)

lowest "cost".

work if graph is unweighted. use, bfs

Cycle Detection

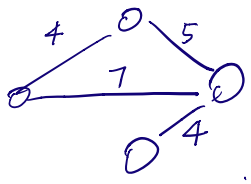
Is there a cycle = path from start to start.

cycle: have 3 or more vertice

Are there cycles that don't include start vertice

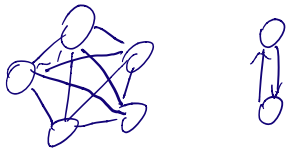
More Graph Terminology

Weighted graph: assign a "cost" to each edge



Network: a weighted directed graph - edge, weights are none negative.

Complete graph: an edge exists between every vertex pair

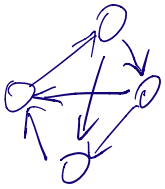


Connected graph (undirected): a path exist between every vertex pair

not has to be complete

Connected graph (directed):

weakly connected: a path exists between every Vertex Pair if direction ignored



strongly connected: ---------- (strongly → is weakly connected) — not ignored.

Length of a path:

edges on the path

Sub-graph: T is a sub-graph of G if all vertices and edges in T exists in G.

P.S. two graph, same, both sub-graph of the other -

"cost" of the path: sum of weights of edges on the path.

Spanning Trees

Definition

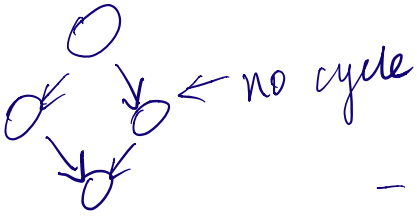
- If T is a sub-graph of G, and contains all vertices of G, it is a spanning Tree. (must pick a root)
- All spanning trees are sub-graphs, have root
- they are tree - have a root, no cycles
- have exactly N-1 edges, N: # vertices (order of the graph)

Easy Spanning Tree: remove edges until there is no cycles. and pick a root.

Cycle detection

DFS(s) - recognize cycle when we visit an already visited nodes successor

Can BFS(s) detect cycle?



- must compare path to know for sure.

DFS and BFS Spanning Trees

DFS algorithm

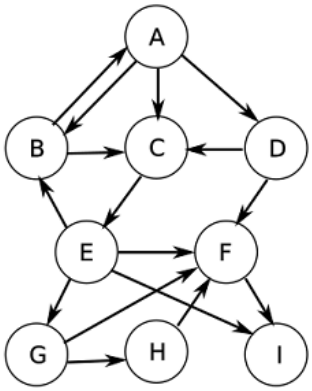
```
add vertex V to T
mark vertex V as visited
push V to stack
while stack is not empty
     C is vertex on top of stack (not popped)
     if no unvisited successor of C
          pop C from the stack
     else
          S is next unvisited successor of C
          add S to T
          add edge (C,S) to T
          mark S as visited
          push S to stack
```

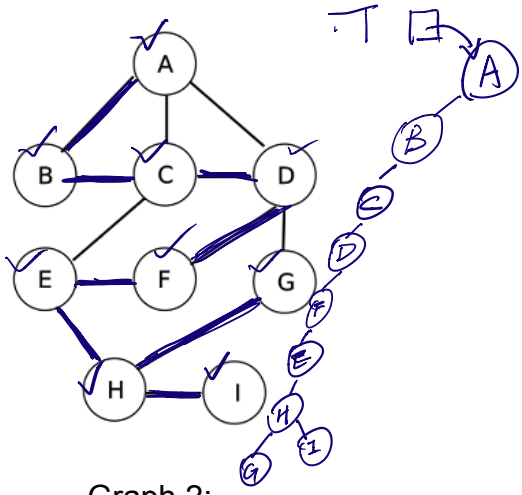
BFS algorithm

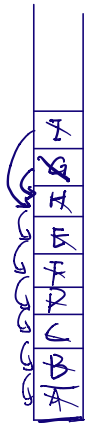
```
Follow the DFS or BFS algorithms as for the traversal
Add V to T
Mark V as visited
Add V to queue
while queue is not empty:
     C = queue.dequeue()
     for each unvisited successor (S) of C
          add S to T
          add edge (C,S) to T
          mark S as visited
          add S to queue
```


DFS Spanning Tree Practice

Which edges are not in the DFS spanning tree starting at A for each graph



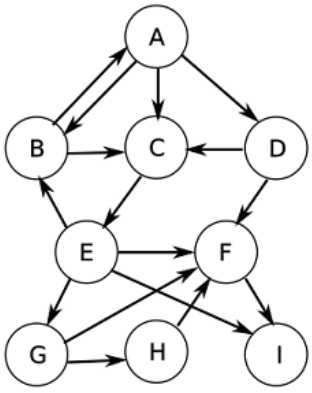


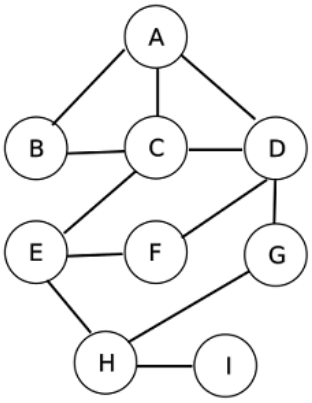


Graph 1:

Graph 2:

Which edges are not in the BFS spanning tree starting at E for each graph





Graph 1:

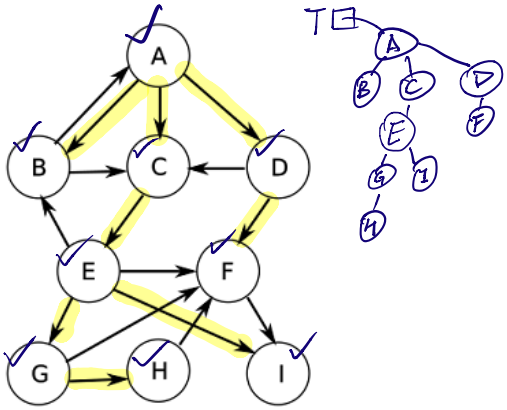
B->A, B->C, D->C
E->B, E->F,
F->I
G->F
H->F

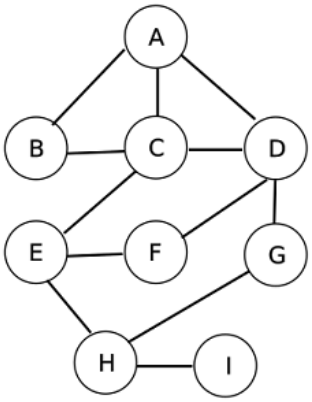
Graph 2:

B-C, C-D, E-F, G-H

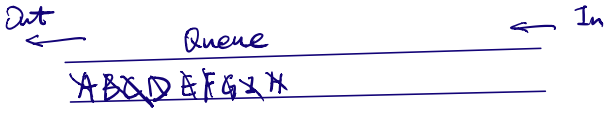
BFS Spanning Tree Practice

Which edges are not in the BFS spanning tree starting at A for each graph

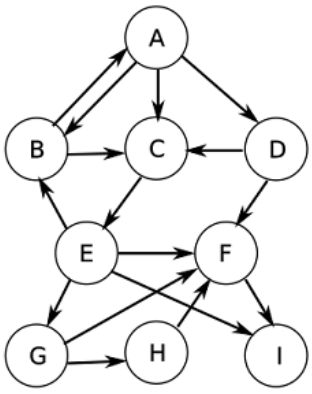


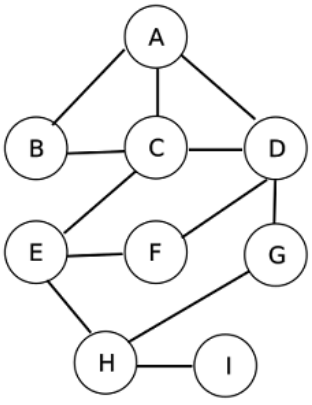


Graph 1: Graph 2:



Which edges are not in the BFS spanning tree starting at D for each graph





Graph 1: Graph 2:

Exam.

(8) If the same element is inserted in to an AVL and Red Black tree, the height of two tree will differ by most one level

False

14. consider full resize.

Load Factor ≥ LoadFactorThreshold

28. copy file from remote repo to local repo. pull

我都错的是什么智障题目.