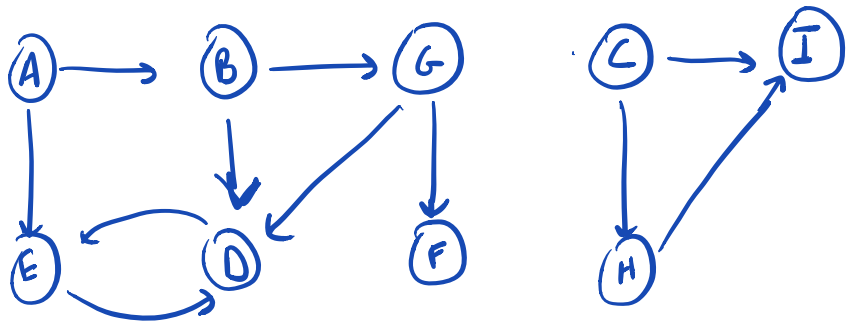
CS170

DISCUSSION 3!

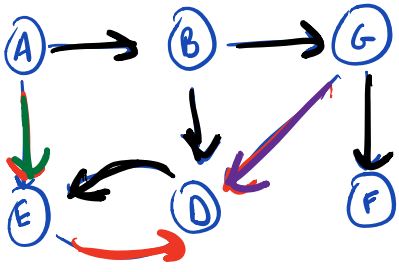
AGENDA

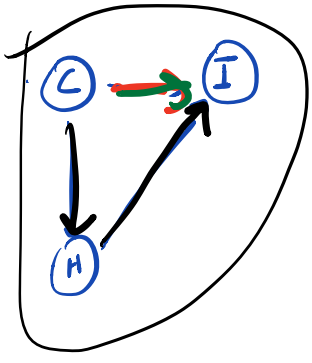
1. Graph algorithms!
   DFS, Djikstras, SCC algorithm
   - Walkthrough, practice
2. Administrivia
3. Some graph proofs

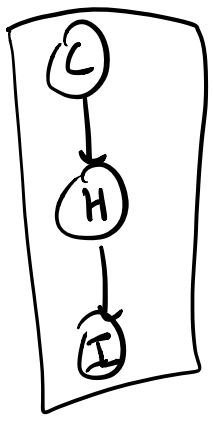
DFS Example



node	pre-visit	post-visit
A	1	12
B	2	11
D	3	6
E	4	5
G	7	10
F	8	9
C	13	18
H	14	17
I	15	16







Tree edges:

Forward edges

Back edges:

Cross - edge

Types of edges in DFS traversal

1. Tree: In the DFS traversal. (u, v)

2. Forward: Edge to non child descendant.

3. Back edge: Edge to ancestor. (u, v)

4. Cross-edge: Already post-visited (before encountering u) (u, v)

Things to note about DFS

(u, v)

1. [u] u [v] v is not possible! If (u, v) in graph

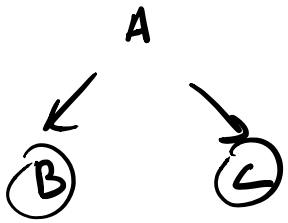
2. In an undirected graph, there are only tree and back/ forward edges.

2 Short Answer

For each of the following, either prove the statement is true or give a counterexample to show it is false.

(a) If (u, v) is an edge in an undirected graph and during DFS, $\text{post}(v) < \text{post}(u)$, then u is an ancestor of v in the DFS tree.

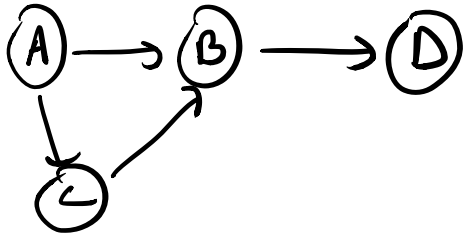
(b) In a directed graph, if there is a path from u to v and $\text{pre}(u) < \text{pre}(v)$ then u is an ancestor of v in the DFS tree.



node	pre	post
A	1	6
B	2	3
C	4	5

DAG

1 A directed graph that contains no cycles.

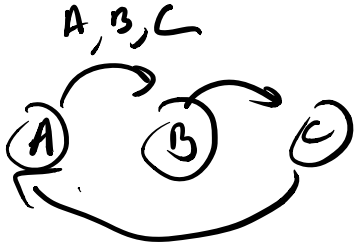


> Property: DAGs have no back edges

(can't have an edge going to an already seen vertex, cycle)

> Property: If (u, v) is an edge in a DAG, then post(u) > post(v).

(no back edges)

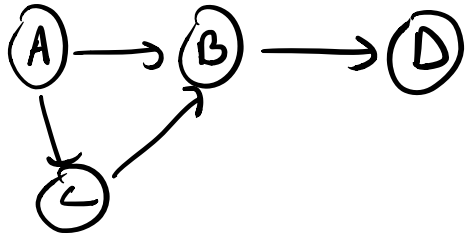


Topological sort algorithm (linearization).

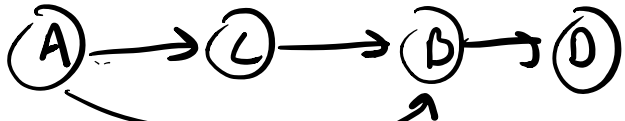
1 Run DFS (G)

2. Output vertices in decreasing post-order.

Since DFS runs in O(V + E) time, linearization runs in O(V+E) time.

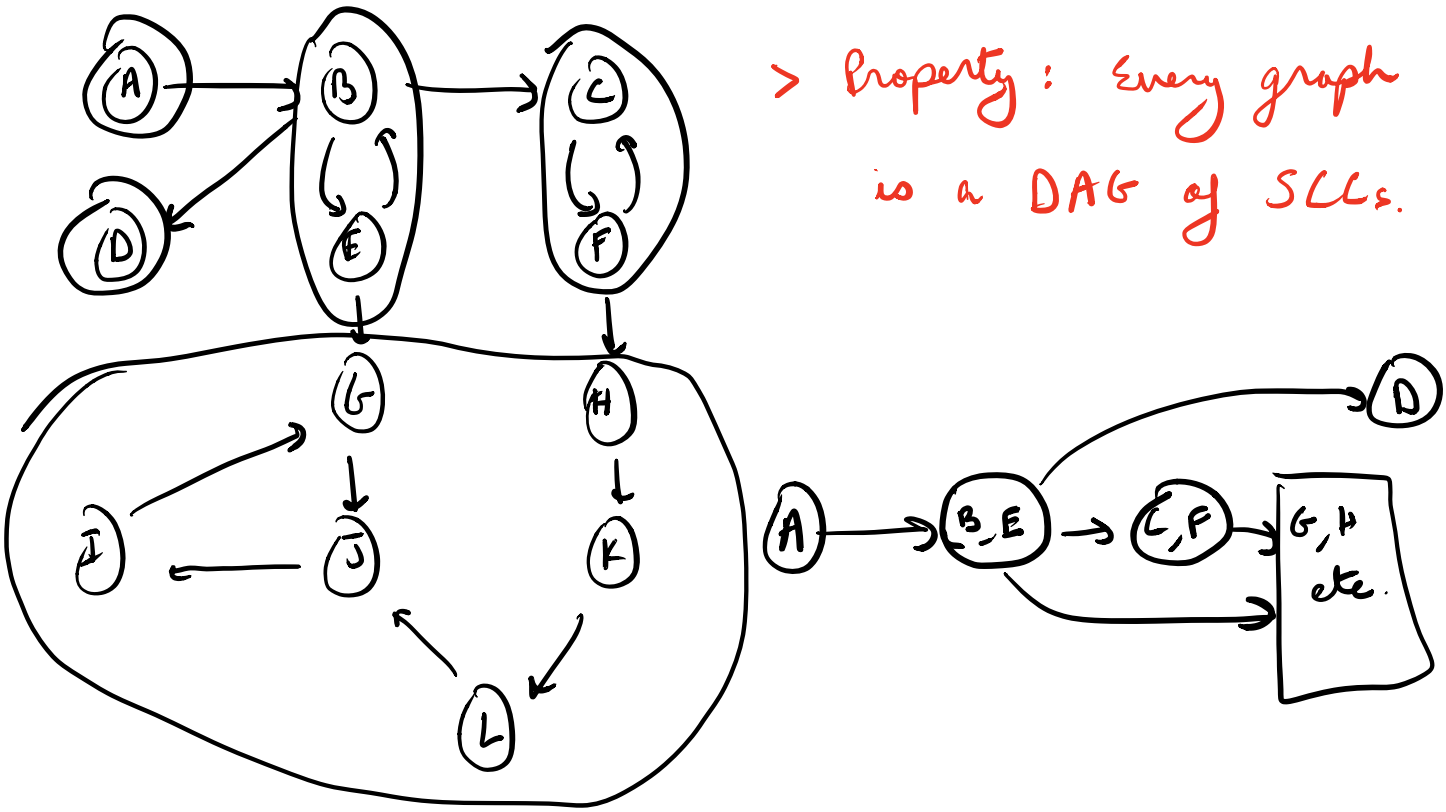


node	pre	post
B	1	4
D	2	3
A	5	8
C	6	7



SCC

- Strongly connected component
- For every pair of vertices (u, v) in the SCC G', there must exist a path from u to v and from v to u.



> Property: Every graph is a DAG of SCCs.

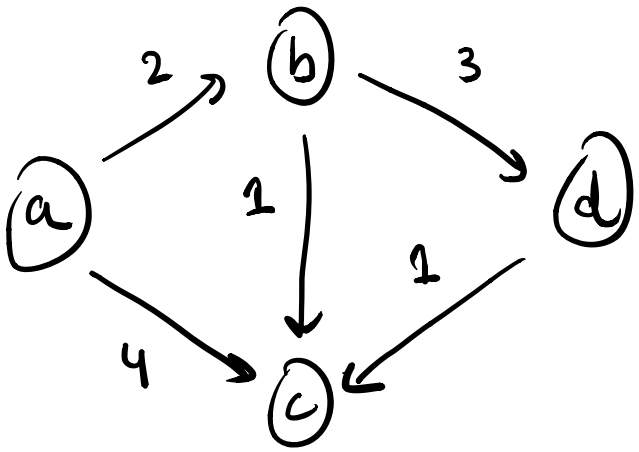
SCC Algorithm

1. Reverse edges in graph to get G^R
2. Do DFS on G^R, find vertex v with largest post-number
   - This is a source in G^R, so a sink in G!
3. Run explore from v on G to get the sink SCC.
4. Remove the sink SCC and go back to 3!

Dijkstra's

- shortest paths, with a priority queue

Example



v	dist [v]

---

4 Dijkstra's Algorithm Fails on Negative Edges

Draw a graph with five vertices or fewer, and indicate the source where Dijkstra's algorithm will be started from.

1. Draw a graph with no negative cycles for which Dijkstra's algorithm produces the wrong answer.
2. Draw a graph with at least two negative weight edge for which Dijkstra's algorithm produces the correct answer.

5 Fixing Dijsktra's Algorithm with Negative Weights

Dijkstra's algorithm doesn't work on graphs with negative edge weights. Here is one attempt to fix it:

1. Add a large number M to every edge so that there are no negative weights left.
2. Run Dijkstra's to find the shortest path in the new graph.
3. Return the path found by Dijkstra's, but with the old edge weights (i.e. subtract M from the weight of each edge).

Show that this algorithm doesn't work by finding a graph for which it must give the wrong answer.