SCHOOL OF ENGINEERING

VANDERBILT UNIVERSITY

CS 3250
Algorithms
Graphs: Depth-First Search

Topological Sort

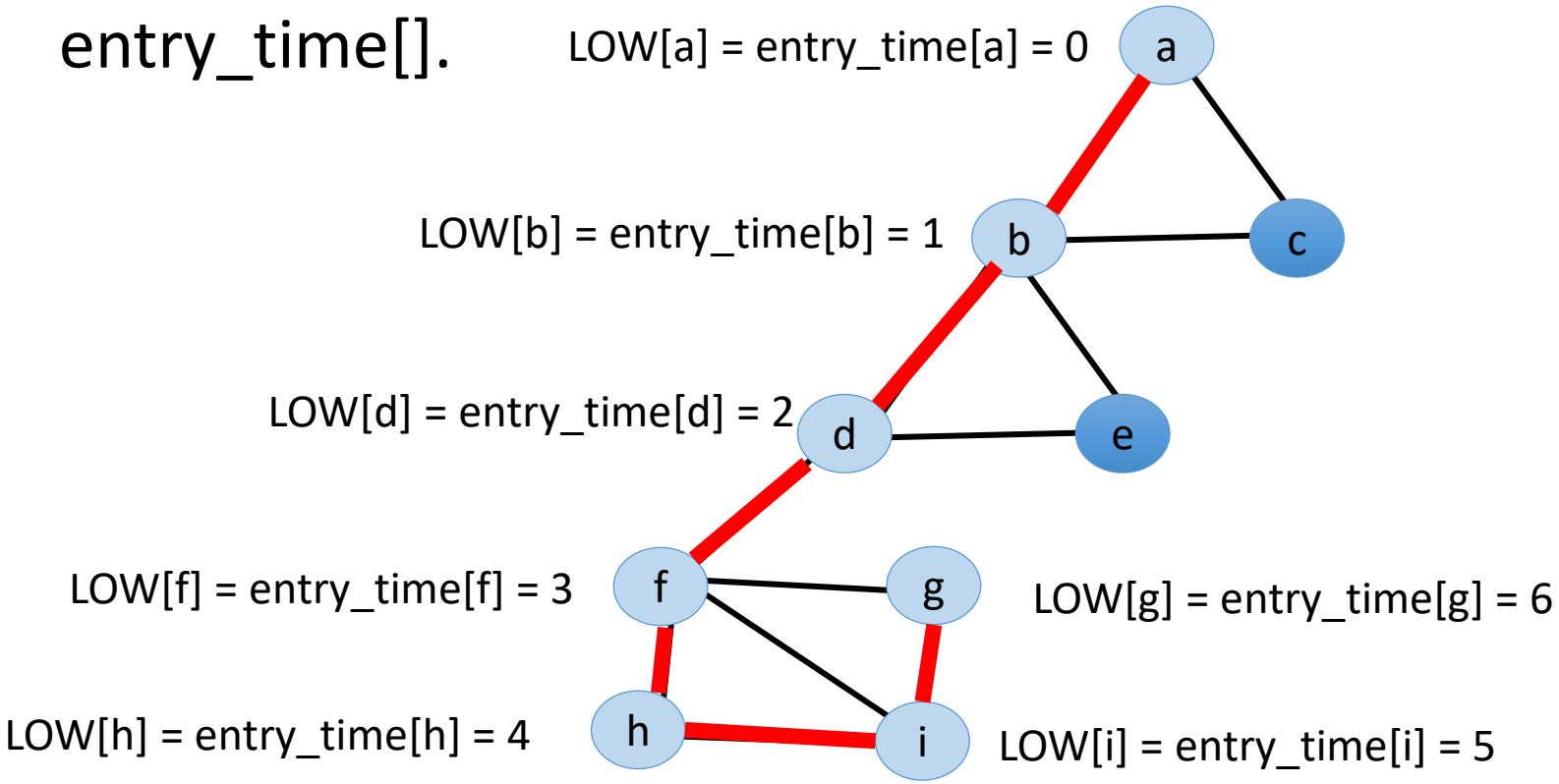
Announcements

- **HW2** is due Wednesday, February 7th by 9 AM.
- There are two parts:
  1. **Brightspace Hashing Quiz**. Formative assessment. Graded but not timed.
  2. **Gradescope Questions.**
     - For the first Gradescope question, you will need to reference the graph generator quiz on Brightspace to generate your random graph for Exercise #1.
     - If you haven't started the HW yet, get going.

VANDERBILT UNIVERSITY

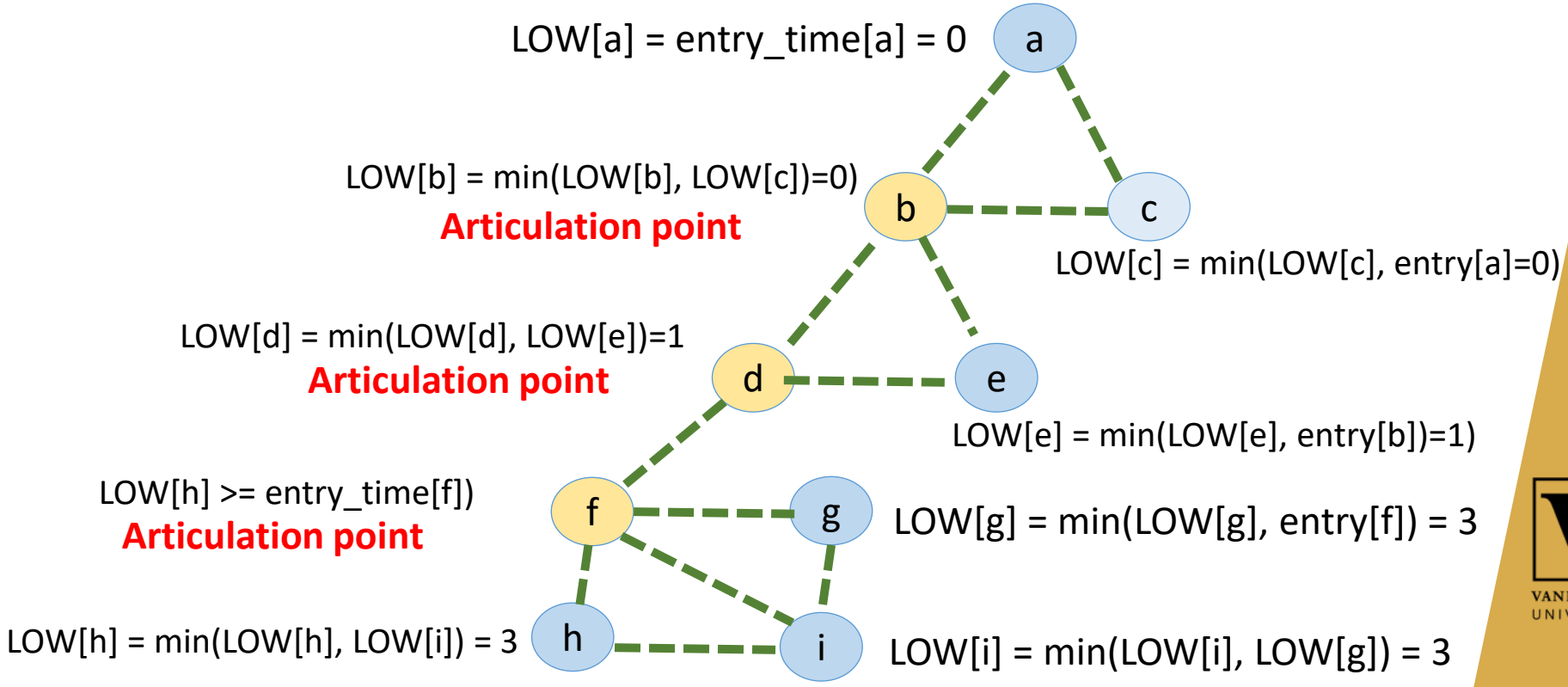
Recap: Tarjan's Articulation Point Algorithm

- **Walkthrough Step 1:** During DFS, let's pretend we discover the vertices in this order: a, b, d, f, h, i, g (I know, it's not lexicographic). As we **discover** each of those vertices, we initialize the LOW of each newly discovered vertex to its entry_time[].



Recap: Tarjan's Articulation Point Algorithm

Walkthrough Step 12: Backtrack from b and recognize that vertex a is the root of the DFS tree of discovery with only one child. Therefore vertex "a" is not an articulation point. DFS now complete and we have located all articulation points.



VANDERBILT UNIVERSITY

Recap: Tarjan's Articulation Point Algorithm

Execute DFS(v)

When vertex v is discovered, LOW[v] = entry_time[v] = time

time = time + 1

For each of v's neighbors, neigh

Total = O(V+E)

More informative: Θ(V+E)

    if neigh is undiscovered

        execute DFS(neigh)

        LOW[v] = min of {LOW[v], LOW[neigh]} (backing up)

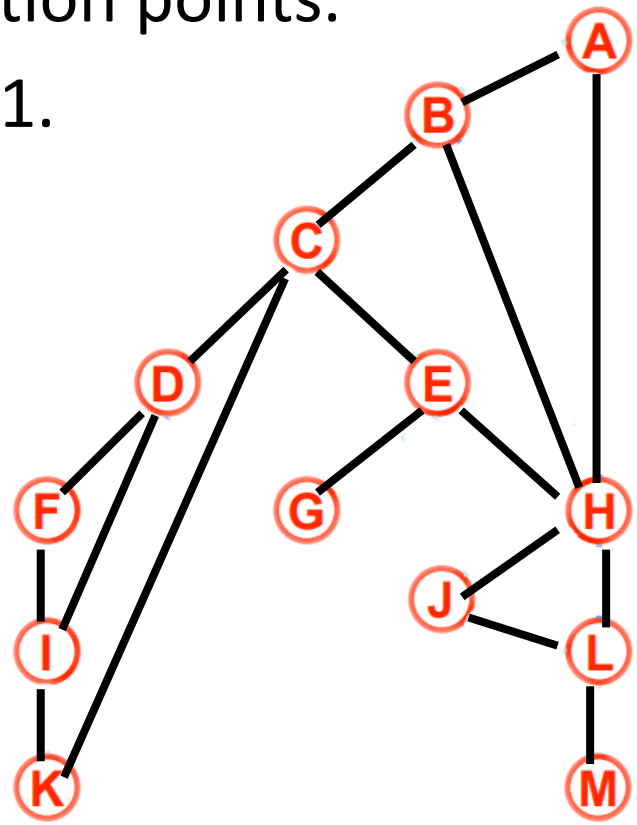
        if LOW[neigh] ≥ entry_time[v], v is articulation point

    else if (neigh is not v's parent but has been discovered)

        LOW[v] = min of {LOW[v] and entry_time[neigh]}

On Your Own: Tarjan's Algorithm

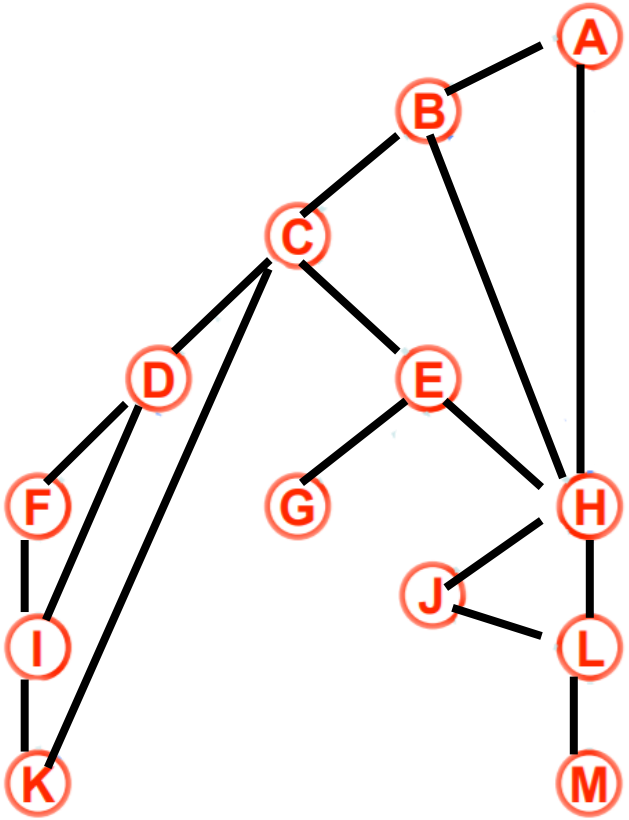
- Walk through and explain Tarjan's algorithm using the graph below starting at vertex A. When given a choice between 2 paths, choose the one that comes first lexicographically. Identify all articulation points.
- Start your clock at 1.

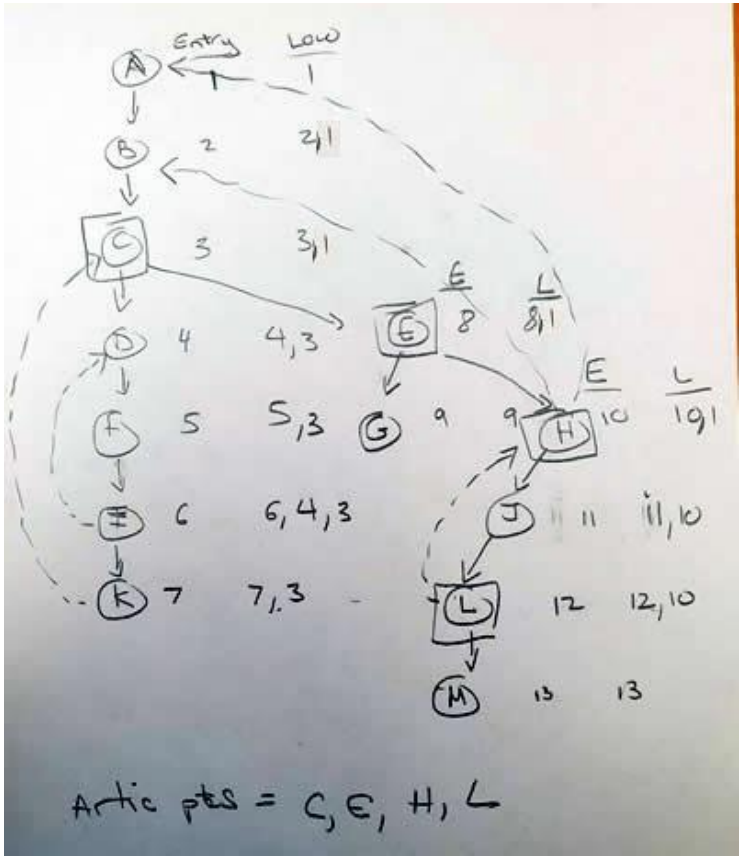


VANDERBILT UNIVERSITY

On Your Own: Tarjan's Algorithm

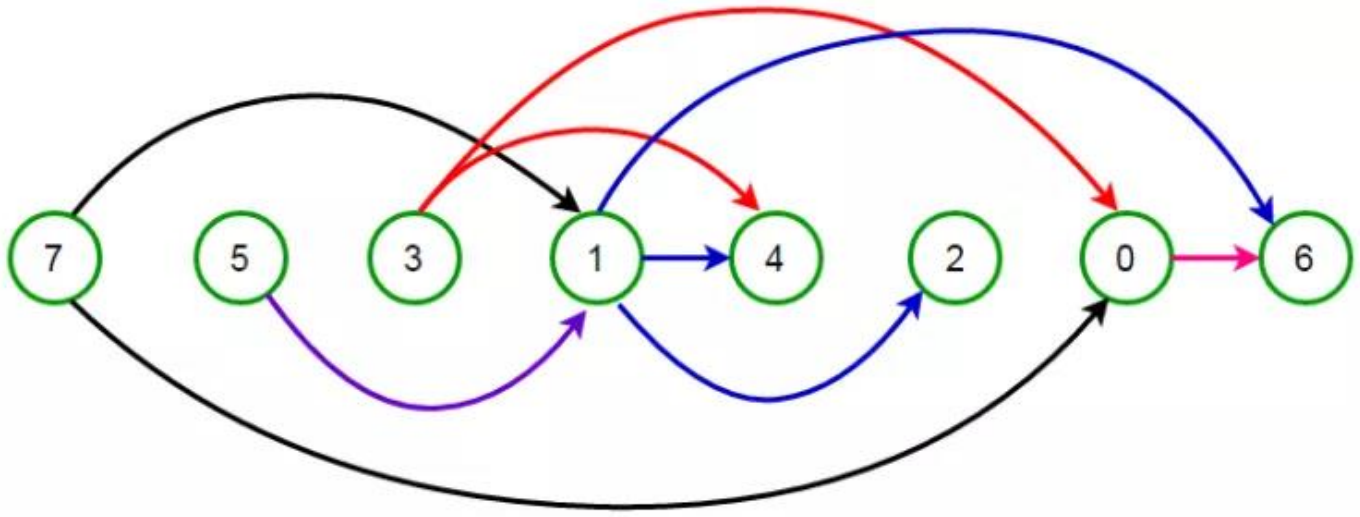
- Walk through and explain Tarjan's algorithm using the graph below starting at vertex A. When given a choice between 2 2 paths, choose the one that comes first lexicographically.





Depth-First Search: Topological Sort

- Another popular use of DFS is Topological Sort which is often used as the basis in task scheduling.
- A **topological sort** is a linear ordering of the vertices in a graph G such that for every **directed** edge uv, vertex u comes before v in the ordering.



Topological Order

Depth-First Search: Topological Sort

- **Analogy:** What is an ordering in which you can take all the CS core classes for the CS degree?
  - CS1101 → CS 2201 → CS 2212 → CS 3250 → CS 3270
- Can you think of another valid ordering?
  - CS1101 → CS 2212 → CS 2201 → CS 3250 → CS 3270
- What about this ordering?
  - CS2201 → CS 2212 → CS 1101 → CS 3250 → CS 3270

Depth-First Search: Topological Sort

- **Which of the following is true about a graph that contains a topological ordering?**

A. There must be a vertex with no incoming edges.

B. The graph must be sparse.

C. The graph must be a DAG.

D. Both A and C

E. Both A and B

VANDERBILT UNIVERSITY

Depth-First Search: Topological Sort

- **Which of the following is true about a graph that contains a topological ordering?**

A. There must be a vertex with no incoming edges.

B. The graph must be sparse.

C. The graph must be a DAG.

D. Both A and C

E. Both A and B

VANDERBILT UNIVERSITY

Depth-First Search: Topological Sort

- **Is it true that every DAG must have at least one valid topological ordering?**

A. No. It's only true that if there is a topological ordering the graph is a DAG. It's not necessarily true that every DAG has a topological order.

B. Yes. Every topological ordering must be a DAG and every DAG has a topological order.

C. Maybe. Some topological orderings might be on an undirected graph.

VANDERBILT UNIVERSITY

Depth-First Search: Topological Sort

- **Is it true that every DAG must have at least one valid topological ordering?**

A. No. It's only true that if there is a topological ordering the graph is a DAG. It's not necessarily true that every DAG has a topological order.

B. Yes. Every topological ordering must be a DAG and every DAG has a topological order.

C. Maybe. Some topological orderings might be on an undirected graph.

VANDERBILT UNIVERSITY

Topological Sort: Observations

- Every DAG has a topological ordering.
- **Think about it…**
- This must be true. After all, let's say you were trying to complete the CS core requirements and Data Structures was a pre-requisite to Discrete Structures which was a pre-requisite to Algorithms.
- What happens if the CS department passes a new requirement that you cannot not take Data Structures unless you first complete Algorithms?

Topological Sort: Observations

Some observations:

1. A graph has a topological ordering IFF the graph is a DAG (there cannot be any cycles in the graph).
2. There must be at least one "source vertex" with **no incoming edges**
3. There must be at least one one "sink vertex" with **no outgoing edges**.
4. There can be more than one valid topological ordering for a given graph G.

Topological Sort: Observations

- **Claim:** There must be at least one "sink vertex" with no outgoing edges in any DAG.
- **Proof (BWOC):** Suppose there isn't. This means in an arbitrary graph with n vertices, I can pick any arbitrary vertex B and begin exploring from it (I must be able to explore from it since it's not a sink vertex). Let's follow an edge from vertex B to another vertex C. C can't be a sink vertex either so let's explore again to another vertex D. I can continue this process for all n vertices. But if I follow n arcs, I will have seen n+1 vertices. This means via the **pigeon-hole principle** I must have seen some vertex twice, which in turn means this graph has a cycle and is not a DAG. In other words, there must be at least one sink vertex in any DAG.■

Topological Sort: Observations

Proof (Inductive): Every DAG has a topological ordering

- **Base case:** n = 2; n is number of vertices in Graph G. Since the edge is directed, the graph must have a topological ordering from target vertex to source vertex.
- **Inductive hypothesis:** Assume our base case holds for k vertices. In other words, a DAG on k vertices has a topological ordering. We will prove that the hypothesis holds for k+1 vertices. That is, a DAG G on k+1 nodes has a topological ordering.
- Given an arbitrary graph G with k+1 vertices that is a DAG, find an arbitrary vertex v in G where v is a node with no outgoing edges. Remove v to create G', a graph on k vertices.

Topological Sort: Observations

Proof (con't): Every DAG has a topological ordering

- It must be the case that G' is a DAG as well. After all, if you have a DAG and you **remove** a vertex, you cannot create a cycle. So, G' is a DAG on k vertices and it has a topological ordering by the inductive hypothesis
- We can concatenate Topological ordering (G') + v to get a topological ordering of G.
- QED

That’s All For Now…

- Coming to a Slideshow Near You Soon…
  1. Kosaraju’s Algorithm
  2. Minimal Spanning Trees

That’s All For Now