Lecture 12:
• Bipartite Graphs
• Topological Sorting

Announcements
• PS-3 due
• HW-2 out, due Thurs
• PS-4 out, due Tues, Oct 18

Implementing DFS using Iteration

Basic Idea:
• similar to BFS, but use a stack instead of a queue
• note this is what the recursion does “for us” ... plus call overhead

Check In: Trace example below using iteration (i.e., a stack)

Note: Must implement DFS iteratively for HW-3
Bipartite Graphs

Definition:
- Let $G = (V, E)$ be an undirected graph
- $G$ is bipartite iff $V$ can be partitioned into 2 sets ...
- ... such that for $(u, v) \in E$, $u$ and $v$ are in different partitions
- Thus, all edges “go between” the two sets

Check-In: Which of these are bipartite? What are the partitions?

![Graphs](image)

2-Colorability

A bipartite graph is 2-colorable ... decision problem: is $G$ 2-colorable?
- Nodes assigned 1 of 2 colors, adjacent nodes must have different colors
- Note can also have 3-colorable, 4-colorable, etc., graphs

![Coloring](image)

Note: Undirected graphs without odd-length cycles are bipartite (2-colorable)
Bipartite Algorithm

HW-3: Check if a graph is bipartite by finding a 2-coloring

**Basic Idea:** Re-implement DFS to find a 2-coloring

- assign each node as either BLUE, GREEN, or NEITHER
- as \((u, v)\) edges traversed, check that \(u\) and \(v\) have opposite colors
- if \(u\) and \(v\) have the same color, not 2-colorable
- if \(v\) not assigned color yet, assign \(v\) opposite color of \(u\)
- repeat process over all nodes in graph (since could be disconnected)

*Note:* we ignore edge direction (colorability independent of direction)

Could we use BFS instead of DFS? ... Yes!

- Note cost is also the same: \(O(V + E)\)

---

Topological Sorting

**Basic Idea:**

- view directed edges as "precedence" constraints (e.g., \(u\) precedes \(v\))
- topological sort orders vertices according to precedence
- examples: course plan (prerequisites), task scheduling (task dependencies)

**Check In:** Give a topological sort of the graph:
Topological Sorting

Check In: What if $G$ is disconnected?

- can do each component separately (in any order) or intermixed
- as long as precedence constraints are satisfied

Check In: What happens if $G$ has a cycle? ... i.e., is not a DAG

- then there is not a topological sort for $G$
- since vertices in the cycle depend on (precede) each other