Lecture 9:
• Breadth-First Search

Announcements
• HW-1 out, due Sept 29
• Exam 1, Tues, Oct 4
• PS-3 out, due Thur, Oct 6

Graph Search

Basic Idea: Iterating through (edges of) a graph
• graph version of “looping” through a list or “traversing” a tree

Example: determine if \( v \) is reachable from \( u \)
• start “searching” from \( u \)
• following edges to see if we arrive at \( v \)

Like list iteration, graph search is a “building block” in many graph algorithms
Graph Search

Consider the graph:

(1). What are possible “traversals” starting at $u$? … many!

(2). What are ways we can represent a traversal?
   • order vertices according to when discovered (e.g., $u \ v \ t \ s$)
   • as “search trees” to also capture edges traversed

(3). What if the graph was undirected?
   • more paths and a cycle $(u \ v \ s)$ … can lead to infinite loops if not careful

Breadth-First Search

Two main types of graph traversals:
   • “Breadth” first search (BFS) … we’ll start here
   • “Depth” first search (DFS) … next week

Basic Idea of BFS:

(1). start at some vertex $s$
(2). visit all of $s$’s adjacent vertices
(3). visit all of $s$’s adjacent vertices’ adjacent vertices
(4). and so on until all ancestors searched
Breadth-First Search (cont)

As we search, keep track of vertices visited ... How?
• as a list of visited vertices ... inefficient to check if \( u \) visited
• using an array indexed on vertex number ... much faster!!!

As we search, build up a search tree
• store parent edges using a dictionary (map) indexed on vertex numbers
• given \( u \), \( \text{parent}[u] \) gives parent vertex

Algorithm: BFS
Input: A graph \( G = (V, E) \) and a source vertex \( s \in V \).
Result: Dictionary mapping discovered vertices to their parent vertices.

1 \textbf{begin} \\
2 \hspace{1em} \text{mark all vertices } v \in V \text{ as not visited} \\
3 \hspace{1em} \text{mark vertex } s \text{ as visited} \\
4 \hspace{1em} \text{set parent of } s \text{ to -1 (denoting the root)} \\
5 \hspace{1em} Q \leftarrow \text{a queue data structure initialized with } s \\
6 \hspace{1em} \textbf{while } Q \text{ is not empty } \textbf{do} \\
7 \hspace{2em} \text{\( u \leftarrow \) the vertex at the front of } Q \text{ (dequeue)} \\
8 \hspace{2em} \textbf{for } v \text{ adjacent to } u \text{ in } E \textbf{ do} \\
9 \hspace{3em} \text{\textbf{if } v \text{ is not yet visited } \textbf{then}} \\
10 \hspace{4em} \text{mark } v \text{ as visited} \\
11 \hspace{4em} \text{set } v \text{'s parent to } u \\
12 \hspace{4em} \text{add } v \text{ to the end of } Q