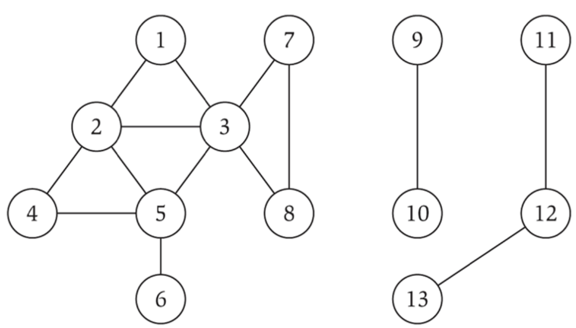
CS311 Data Structures
Lecture 15 — Graph

Jyh-Ming Lien

November 20, 2017

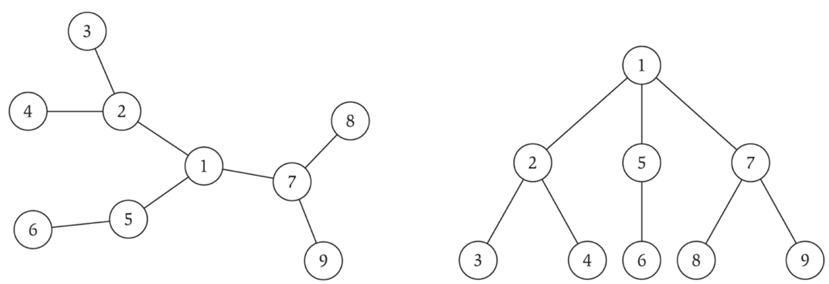
Graph Representation

- Terminology $G = (V, E)$
  - V = nodes or vertices $\{v\}$
  - E = edges between pairs of nodes, $\{e = (u, v)\}$, where u and v are called **ends** of e
  - For directed edge $e = (u, v)$ is an ordered list where u is the **tail** and v is the **head** and e **leaves** u and **enters** v.
  - A path is a sequence of vertices $v_1, v_2, \cdots, v_{k-1}, v_k$. A path is called **simple** if $v_i \neq v_j \forall i \neq j$
  - A cycle is a path $v_1, v_2, \cdots, v_{k-1}, v_k$ in which $v_1 = v_k, for k > 2$, and the first $k-1$ nodes are all distinct
  - An undirected graph is **connected** if for every pair of nodes u and v, there is a path between u and v.



Graph and Tree

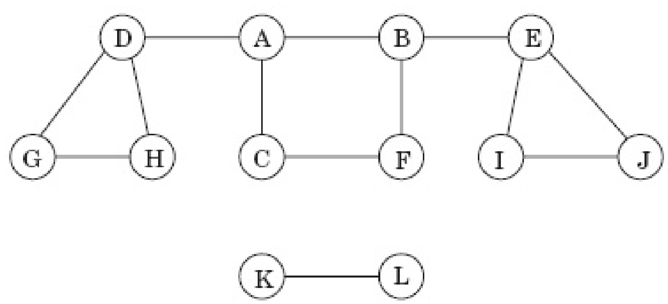
- An undirected graph G is a tree if
  - G is connected
  - G does not contain a cycle
  - G has $n-1$ edges, where n is the number of nodes in G

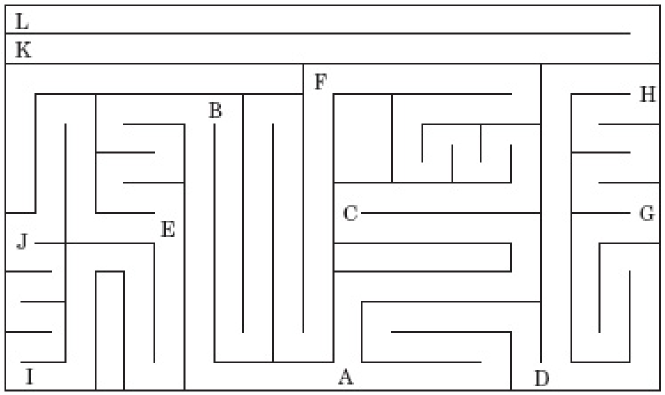


- Many algorithms work by converting a graph to a tree (the simplest representation of the graph)
  - shortest path tree
  - spanning tree
  - exploring tree (BFS, DFS, ...)
  - ...
  -

Graph Search

- What parts of the graph are reachable from a given vertex? (i.e., connected components)
- Many problems require processing all graph vertices (and edges) in systematic fashion
- Basic tools to *safely* explore an unknown environment
    -
    -





Graph Search

- Basic exploration algorithm

Algorithm 2.1: $\text{EXPLORE}(G = \{V, E\}, v \in V)$

- Can the algorithm always work?
  - *proof*

Graph Search

- Example: EXPLORE(B)