Announcements

Programming Assignment #8 ⇒ Breadth-first search (May 16th)

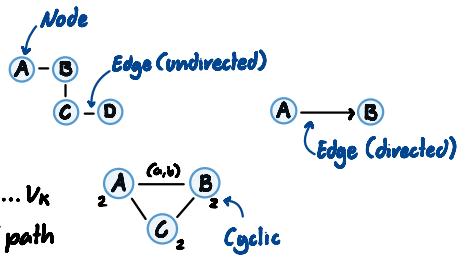
- Run breadth first search on the mazes and find the path

Programming Assignment #9 ⇒ Extra credit

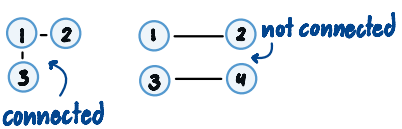
- Due after the final

Graphs

Definitions



- A graph is a collection of nodes (vertices) and edges
- Graphs may be directed (arrows) or undirected (lines)
- The degree of a node is equal the number of edges insident
- A path (of length k) from vertex a to b is a sequence $V_0, V_1, V_2 \dots V_k$ where $a = V_0$ and $b = V_k$ and $(V_i, V_{i+1}) \in E$ for all $i = 0 \dots k-1$ of path length # edges in the path
- If there is a path from a to b, b is reachable from a
- A path is simple if all nodes on the path are unique
- A graph with no cycles is acyclic
- (a, b) edge leaves a and arrives at b
- A "regular graph" is one in which one edge exists between any pair of nodes and each edge only connects a pair of nodes



Representing a graph ⇒ Two common ways + custom

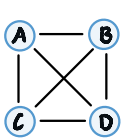
1. Adjacency list (often prefered especially for sparce graphs)
2. Adjacency matrix

A graph with 4 nodes

Adjacency list ⇒ Good for most use cases

- One list (vector) for each node. List includes adjacent nodes
- Good for asking what nodes are adjacent to some node
- Memory usage O(E) where E is the number of edges

A) B, C, D
B) A, C, D
C) A, B, D
D) A, B, C



Adjacency matrix ⇒ Good for dense graphs

- Good for asking if a and b are connected
- Memory usage $O(N^2)$