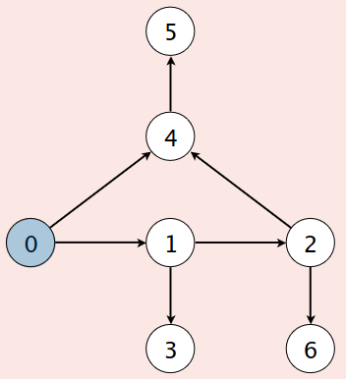
Precept Outline
- Review of Lectures 15 and 16:
  - Graphs and Digraphs
  - Graph search: DFS and BFS

Relevant Book Sections
- Book chapters: 4.1 and 4.2

A. Review: Graphs, Digraphs and Graph Search

Your preceptor will briefly review key points of this week's lectures. They may use the following graph to trace examples:



B. Rooted DAGs

Suppose G is a digraph (i.e. a directed graph). G is *acyclic* if it doesn't contain any directed cycles (i.e. a directed path whose first and last vertices are the same). Describe an algorithm that runs in $O(V + E)$-time to detect if G is acyclic.

Note. A directed acyclic graph is often known by its acronym *DAG*.

In a DAG, a vertex v is an ancestor of a vertex w if there is a directed path from w to v.

A **rooted DAG** is a DAG that has one vertex – the root – that is an ancestor of every other vertex. Describe an algorithm that runs in $O(V + E)$-time to detect if a given DAG G is rooted.

C. Tag Game on a Graph

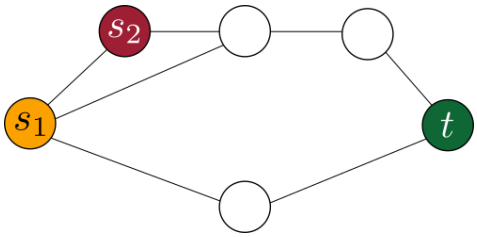
Two players compete on a game of tag on a (undirected) graph G. Here are the rules of the game.

- The first player starts at vertex s_1 and wins if it reaches a certain vertex t.
- The second player starts at vertex s_2 and wins if it tags the first player before they reach vertex t. The second player tags the first if they are ever both on the same vertex. (we assume that if both players are in t, then it counts as a tag, so the second player wins)
- The players move simultaneously by taking a step to a vertex adjacent to the one they are currently on, or they can choose to stay in the same vertex they are on.
- The second player cannot be more than k steps away from s_2.
- The second player is omniscient, which means that they can always find their optimal path and they can know the path taken by the first player in advance. In other words, you can assume that the game is played in the following way:
  - The first player picks some path from s_1 to t.
  - Then, the second player sees what the first player picked, and based on that picks some path.
  - The second player wins if it shares a vertex with the first player at any point, and otherwise the first player wins.

Given G, s_1, s_2, t, and k, your task is to design an algorithm to decide whether the first player can win the game.

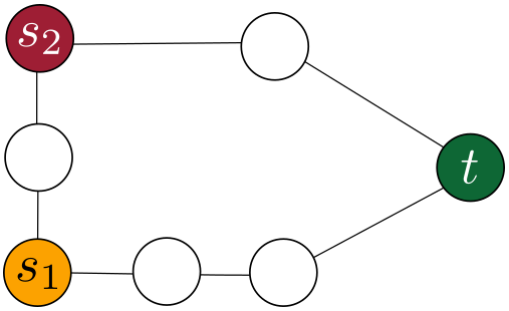
Example:

Suppose the following graph is the one the players are using, labeled accordingly.



Then, the first player can win by moving to the bottom vertex and then to t, regardless of what k is. It's easy to see that the second player can never catch the first if they follow this path.

Now, suppose the following graph is the one the players are using, labeled accordingly.



If k is 3, then the first player cannot win. If the first player decides to take the bottom path, then the second player can use the top path to go around and tag it just before they reach t. If the first player decides to take the top path, then the second player can just wait in s_2.

However, if k is 1, then the first player can win by taking the bottom path.

Performance requirements. For full credit, your algorithm must take $\Theta(E + V)$ time, where V and E are the number of vertices and edges in G, respectively.

To help you think about this problem, let's start by considering a simpler version of the problem. Suppose G is a *cycle graph*. A cycle graph with V vertices has V edges and follows the pattern: vertex 1 is adjacent to V and 2, vertex 2 is adjacent to 3, …, vertex $V - 1$ is adjacent to V. So it looks like a cycle.

Describe a solution to the tag game on a graph when the graph is a cycle graph.

Now, suppose G is a *path graph*. A path graph with V vertices has $V - 1$ edges and follows the pattern: vertex 1 is adjacent to 2, vertex 2 is adjacent to 3, ..., vertex $V - 1$ is adjacent to V. So it looks like a path.

Describe a solution to the tag game on a graph when the graph is a path graph.

With the intuition developed before, describe a solution to the tag game on an arbitrary graph.