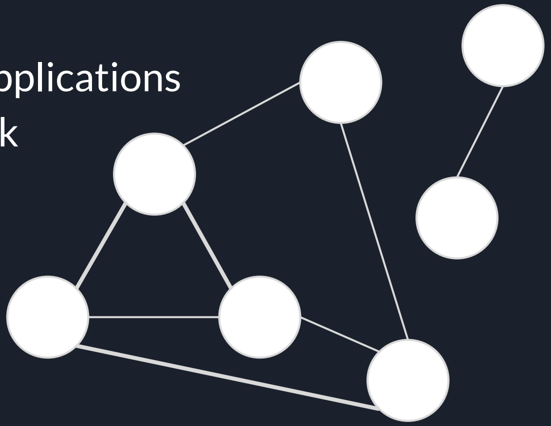
ProgTeam: Week 4

Introduction To Graphs

What is a graph?

- Graph is a set of vertices and edges
- Edges are a pair of vertices (u,v)
    - Edges can be ordered or unordered; called *directed* and *undirected*
    - Edges can have an additional parameter, a *weight*
        - Some graphs are *unweighted*
- Graphs can be *connected* or *unconnected*
- Graphs can simulate several real world applications
    - Connections on an internet network
    - Plumbing in a city
    - Roads in a country
    - Social media friends
    - … and much, much more



How can we work with graphs?

- Most common representation is an *adjacency list*
  - For each vertex v, store a list of all vertices directly reachable from v
  - Works with directed graphs too; only store one edge
- Another common representation is *adjacency matrix*
  - For N vertices, store an NxN matrix, where each entry represents whether an edge is present
  - Can be useful, but takes N^2 memory
- Some graphs don’t need to have edges stored
  - Called *implicit graphs*
  - Example: a 2D grid where squares are neighbors

Breadth-First Search

- On an undirected graph, we can calculate the distance from a source in linear time

- Use a queue, and add vertices in the order we reach them
    - Avoid adding the same vertex to the queue

- Distance of newly found vertex will be distance of old vertex + 1

Example Problem: Knight Jump