Goals for today:

1. Write pseudocode for breadth-first search (BFS) and depth-first-search (DFS) algorithms,
2. Build a spanning tree using DFS and BFS,
3. Build a minimum spanning tree (MST) using Prim's algorithm.

1) relate this maze to graphs? vertices? edges?
2) how to find a path through maze?
3) how to find shortest path through maze?

click on "game" in the row for today's class at go/cs200
Depth-First Search ("backtracking").

Main idea: Keep traversing edges until you "hit a wall," then go back to parent.
→ maintain a tree: connected and acyclic
Depth-First Search in **pseudocode**.

```
depthFirstSearch(G)
    input: connected graph G = (V_G, E_G)
    output: spanning tree T
    1. u ← arbitrary vertex in V_G
    2. T ← ({u}, Ø)
    3. visit(u, G, T)

visit(u, G, T)
    input: starting vertex u, connected graph G = (V_G, E_G),
           current spanning tree T = (V_T, E_T)
    output: updated spanning tree T = (V_T, E_T)
    1. for v ∈ neighbors(u, G) do
       2. if v ∈ V_T then
       3. continue
       4. E_T ← append({u, v})
       5. V_T ← append(v)
       6. visit(v, G, T)
```

*avoid cycle!* [root]
Exercise 1: Build spanning tree of this graph using DFS.

- Start at vertex e.
- Visit neighboring vertices in alphabetical order.
- List order of vertices visited.
Breadth-First Search ("flooding").

Main idea: Visit neighbors one "level" at a time.
→ maintain a tree: connected and acyclic
Depth-First Search in pseudocode.

```
breadthFirstSearch(G)

    input: connected graph G = (V_G, E_G)
    output: spanning tree T = (V_T, E_T)

1    u ← arbitrary vertex in G
2    T ← ({u}, ∅)
3    L ← {u}   # unprocessed vertices
4    while L ≠ ∅
5        v ← pop(L)  # remove first vertex from L
6        for w ∈ neighbor(v, G)
7            if w ∈ L ∨ w ∈ V_T
8                continue
9        L ← L ∪ w
10       V_T ← append(w)
11       E_T ← append({v, w})
```
Exercise 2: Build spanning tree of this graph using BFS.

- Start at vertex e.
- Visit neighboring vertices in *alphabetical* order.
- List order of vertices visited.
Prim's algorithm for constructing a minimum spanning tree (MST).

Minimum spanning tree: Spanning tree of a graph with minimum **sum** of **edge weights**.

Main idea: Add minimum weight edge that is (1) connected to current tree and (2) does not form a cycle.

weight = 22