Graph Q's?

- Path between 2 vertices? ← today
- Shortest path? ← Monday
- Acyclic? ← next weds.
- Connected? ← HW

  (Acyclic? and Connected? ⟩ Tree?)

- Number of edges?
- Number of leaves?
- Number of cycles? ← !!?

Depth-First Search (DFS)

Idea: explore as far as we can in one "direction", then backtrack + try other directions.

Use "markers" to record where we've been, so we don't get stuck in a loop.

{list/array of booleans
dictionary vertices → booleans
set} Θ(1) to mark and check

Mark all vertices UNVISITED
[parent ← empty dictionary]

DFS(G, u):
    Mark u VISITED   Θ(1)
    for each neighbor v of u: assume it takes O(deg(u)) time to list neighbors of u.   degree of u.
        if v is UNVISITED: Θ(1)
            [parent[v] ← u.]   Θ(1)
            DFS(G, v) ←

We look at each vertex at most once.
We look at each edge at most twice.

To check if s, t are connected: run DFS(G, s), check if t was VISITED.

To find a path s → t, run DFS(G, s), then trace a path backwards from t to s: t, parent[t], parent[parent[t]], etc ... s.

Time complexity?

Time complexity of DFS is $O(\underset{\text{\# of vertices}}{V} + \underset{\text{\# of edges.}}{E})$.