A graph \( G = (V, E) \)

- **Vertices** (a set of abstract objects. Choose whatever objects are useful for your application!)
- **Edges** (undirected: unordered pairs of vertices. Directed: ordered pairs)

*Undirected edges* \( uv \)

*Directed edge* \( u \rightarrow v \)

This notation mostly works just for simple graphs
(no loops or multi-edges)
\[ \hat{\mu} \rightarrow \mu \text{ or } \mu \mu \]

we'll assume (usually) graphs are simple

most algorithms generalize cleanly to not simple graphs

- if \( uv \) is an edge, we say \( v \) is a neighbor of \( u \)

- if \( u \rightarrow v \) is a directed edge, \( u \) is a predecessor of \( v \)

\[ \text{in-degree of } u \text{ is } \# \text{ of predecessors} \]

\[ \text{out-degree of } u \text{ is } \# \text{ of successors} \]
may use V or E to denote the number of vertices or edges (depth-first search takes $O(V+E)$ time)

Representations/Data structures
adjacency \( \Theta(v^2) \) matrix space

directed degree graphs use adjacency lists

size \( \Theta(V+E) \) iterate over neighbors of \( u \) in \( O(1 + \text{deg}(u)) \)
BFS

Given $G = (V, E)$.

? Is vertex $v$ reachable from $s$ (is there a path from $s$ to $v$).

Breadth-first search (BFS) visits vertices in increasing order of distance from $s$.

Mark visited vertices so we don’t repeat work.

Initially, all vertices unmarked.
BFS(s):
put (s, s) in a queue
while the queue is not empty
  take (p, v) from the queue
  if v is unmarked
    mark v
    parent (v) ← p
    for each edge vw
      put (v, w) in the queue

(can prove:)
I marks every vertex reachable from s exactly once
2) Set of pairs \((v, \text{parent}(v))\) form a spanning tree on the component of \(G\) containing \(s\) (all those reachable vertices).

3) Paths from \(s\) within spanning tree are shortest paths to each marked vertex (unit edge weights).

Analysis: we run the for loop \(\leq V\) times \(\Rightarrow\) Each edge added to queue twice, \(\Rightarrow\) \(\leq 2E+1\) enqueues \(\Rightarrow\) \(\leq 2E+1\) dequeues
So... total time of...

$O(E)$ time

- if graph is directed, loop over successors at $v$
- how spanning tree over vertices reachable from $s$ (with $s$ as root)

USE BFS TO FIND UNWEIGHTED SHORTEST PATHS!

(ort Dijkstra)
Depth-first search (DFS):

**DFS(v):**
- mark v
- **PREVISIT(v)**
  - for each edge vw
    - if w is unmarked
      - `parent(w) ← v`
      - DFS(w)
- **POSTVISIT(v)**

**DFSALL(G):**
- **PREPROCESS(G)**
  - for all vertices v
    - unmark v
  - for all vertices v
    - if v is unmarked
      - DFS(v)

**DFSALL** marks each vertex once. Sees each edge twice (or once if G is directed). Runs in $O(V + E)$. 