Outline: Search II
- breadth-first search
- shortest paths
- depth-first search
- edge classification

Reading: CLRS 22.2-22.3

Recall:
- **graph search**: explore a graph 
  e.g. find a path from start vertex s to a desired vertex
- **adjacency lists**: array Adj of |V| linked lists 
  - for each vertex u∈V, Adj[u] stores u’s neighbors, i.e. {v∈V | (u,v)∈E^2} 
  just outgoing edges if directed

E.g.
- Diagram of a graph and its adjacency list representation.
Breadth-first search (BFS):
- explore graph level by level from s
  - level $\emptyset = \{s\}$
  - level $i$ = vertices reachable by path of $i$ edges but not fewer
- build level $i > \emptyset$ from level $i-1$ by trying all outgoing edges, but ignoring vertices from previous levels

BFS($V, Adj, s$):
- level = $\{s: \emptyset\}$
- parent = $\{s: None\}$
- $i = 1$
- frontier = $\{s\}$
- while frontier:
  - next = $\[]$
  - for $u$ in frontier:
    - for $v$ in $Adj[u]$:
      - if $v$ not in level:
        - level[$v$] = $i$
        - parent[$v$] = $u$
        - next. append($v$)
  - frontier = next
- $i += 1$
Example:

```
Example:  

```

Analysis:
- vertex \( v \) enters next (\& then frontier) only once (because \( \text{level}[v] \) then set)
  - base case: \( v = s \)
  \( \Rightarrow \text{Adj}[v] \) looped through only once
- time = \( \sum_{v \in V} |\text{Adj}[v]| = |E| \) for directed graphs
  \( \leq |E| \) for undirected graphs
  \( \Rightarrow O(E) \) time
  \( O(V+E) \) to also list vertices unreachable from \( v \) (those still not assigned level)

"LINEAR TIME"

Shortest paths:
- for every vertex \( v \), fewest edges to get from \( s \) to \( v \) is \( \text{level}[v] \) if \( v \) assigned level \( \infty \) else (no path)
- parent pointers form \underline{shortest-path tree} = union of such a shortest path for each \( v \)
  \( \Rightarrow \) to find shortest path, take \( v, \text{parent}[v], \text{parent}[\text{parent}[v]], \text{etc.}, \) until \( s \) (or None)
Depth-first search (DFS): like exploring a maze

- follow path until you get stuck
- backtrack along breadcrumbs until reach unexplored neighbor
- recursively explore

parent = {s: None}

\[ \text{DFS-visit}(V, \text{Adj}, s): \]
for v in Adj[s]:
    if v not in parent:
        parent[v] = s
        DFS-visit(V, Adj, v)

\[ \text{DFS}(V, \text{Adj}): \]
parent = {}
for s in V:
    if s not in parent:
        parent[s] = None
        DFS-visit(V, Adj, s)

search from start vertex s
(only see stuff reachable from s)

explore entire graph
(could do same to extend BFS)
**Example:**

- **forward edge**
- **back edge**
- **cross edge**

**Edge classification:**
- **tree edges** (formed by parent)
- **nontree edges**

- **back edge:** to ancestor
- **forward edge:** to descendant
- **cross edge** (to another subtree)

- To compute this classification, keep global time counter & store time interval during which each vertex is on recursion stack

**Analysis:**
- DFS-visit gets called with a vertex s only once (because then parent[s] set)
  \[ \Rightarrow \text{time in DFS-visit} = \sum_{s \in V} |\text{Adj}[s]| = O(E) \]
- DFS outer loop adds just O(V)
  \[ \Rightarrow O(V+E) \text{ time (linear time)} \]