review session 3/10 from
10:30 – 11:45

A <u>graph</u> $G = (V, E)$

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 $u \to v$

this notation mostly works just for <u>simple</u> graphs

(no loops or muti-edges)
$u \to u$ or uu

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 <u>neighbor</u> of u
- if $u \to v$ is a directed edge, 1) u is a <u>predecessor</u> of v

  2) v is a <u>successor</u> of u
- <u>in-degree</u> of u is # of pred.s
- out-degree of u is # of successors

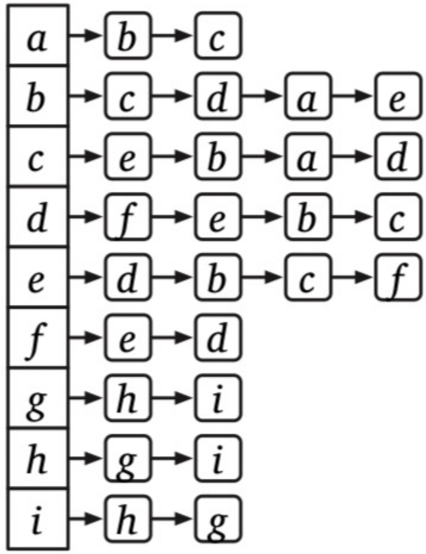
may use V or E to denote the <u>number</u> of vertices or edges

(depth-first seach takes $O(V+E)$ time)

Representations / Data structures

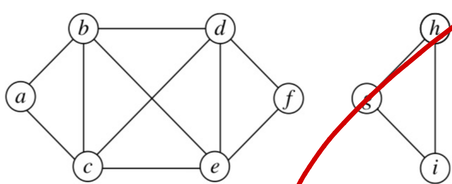
adjacency

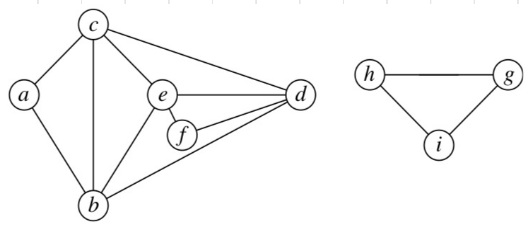
	a	b	c	d	e	f	g	h	i
a	0	1	1	0	0	0	0	0	0
b	1	0	1	1	1	0	0	0	0
c	1	1	0	1	1	0	0	0	0
d	0	1	1	0	1	1	0	0	0
e	0	1	1	1	0	1	0	0	0
f	0	0	0	1	1	0	0	0	0
g	0	0	0	0	0	0	0	1	0
h	0	0	0	0	0	0	1	0	1
i	0	0	0	0	0	0	1	1	0



adjacency matrix $\Theta(V^2)$ space

adjacency list





size $\Theta(V+E)$

directed graphs use lists of successors

degree

iterate over neighbors of u in $O(1+\deg(u))$

BFS

Given $G=(V,E)$.

?: Is vertex v <u>reachible</u> from s (is there a path from s to v).

Breadth-first search (BFS)

visits vertices in increasing order of distance from s

edges

mark visited vertices so we don't repeat work

initially, all vertices unmarked

BFS(s): search from any $s \in V$ (FIFO queue)

put $(\emptyset, s)$ in a queue

while the queue is not empty

take (p, v) from the queue "potential parent" of v

if v is unmarked

mark v

parent(v) $\leftarrow$ p

for each edge vw

put (v, w) in the queue

Can prove:

1) 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 $\le V$ times

$\Rightarrow$ Each edge added to queue twice. So $\le 2E+1$ enqueues

$\Rightarrow$ $\le 2E+1$ dequeues

So... total time of...

O(E) time

- if graph is directed, loop over successors of v.

- now spanning tree over vertices reachable from s (with s as root)

- USE BFS TO FIND UNWEIGHTED SHORTEST PATHS!

(not 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 $O(V+E)$.