CS180 - Graphs

Note Title 12/8/2011

Announcements

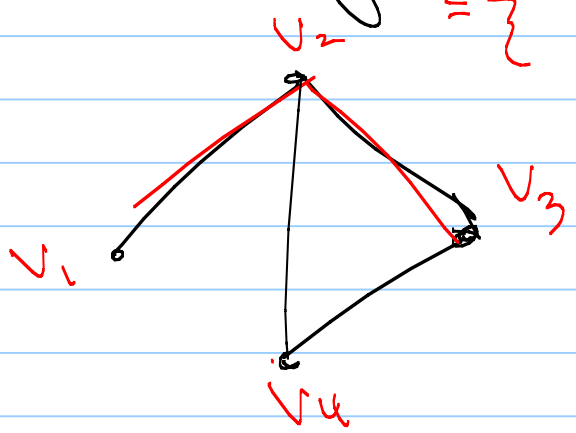
- HW due today
- Final HW - due last day of class
- Review session Thurs or Fri. of finals (bring your finals schedule to next class)

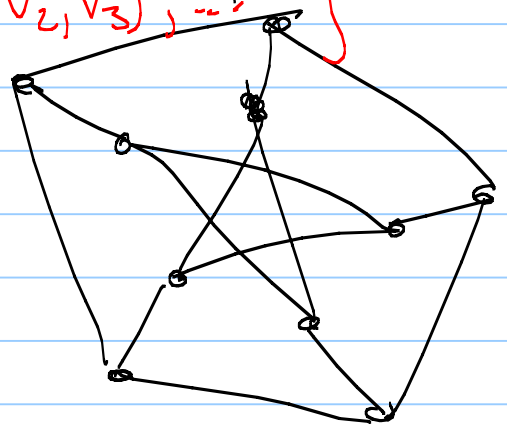
Graphs

A graph $G = (V, E)$ is a set of 2 sets V & E.

V = vertices $= \{v_1, v_2, v_3, v_4\}$

E = edges (which are pairs of vertices) $= \{ \{v_1, v_2\}, \{v_2, v_3\}, \ldots \}$





Why use graphs?

They can model anything!

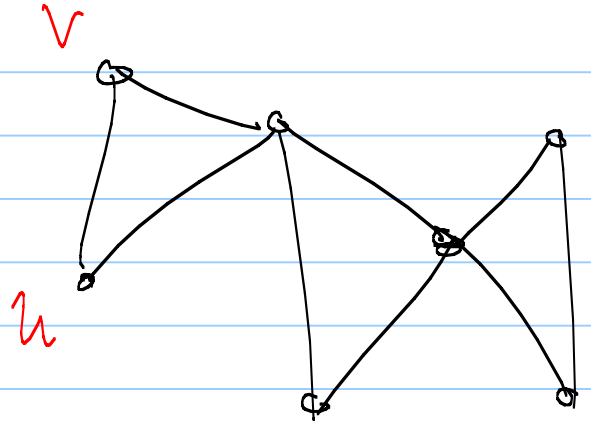
Examples:

- Maps
- Relationships - facebook
- Travel (other methods)
- Internet
- ⋮

Definitions

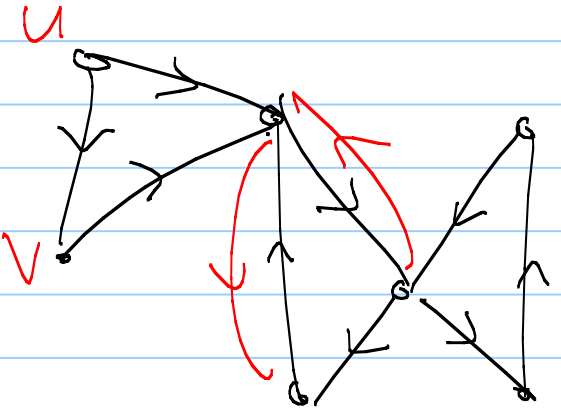
- G is undirected if every edge is an unordered pair

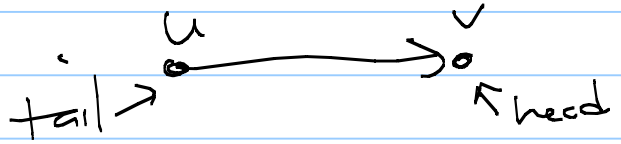
  so $\{u, v\} = \{v, u\}$



- G is directed is every edge is an ordered pair

  $e = (u, v) \neq (v, u)$





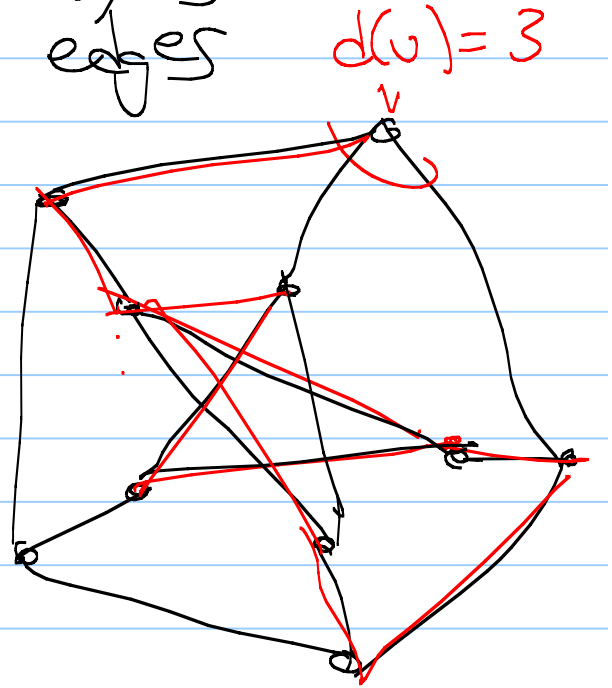
Dfns

- The degree of a vertex, $d(v)$, is the number of adjacent edges

- A path $P = v_1 \dots v_k$ is a set of vertices with $\{v_i, v_{i+1}\} \in E$

- A path is simple if all vertices are distinct

- A path is a cycle if it is simple except $v_1 = v_k$



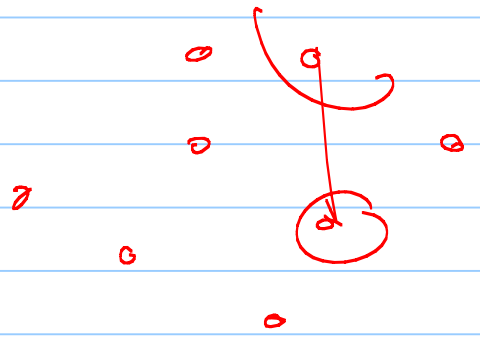
$|V| = n$

Lemma: (degree-sum formula)

$$\sum_{v \in V} d(v) = 2|E|$$

$|E| = m$

Why?



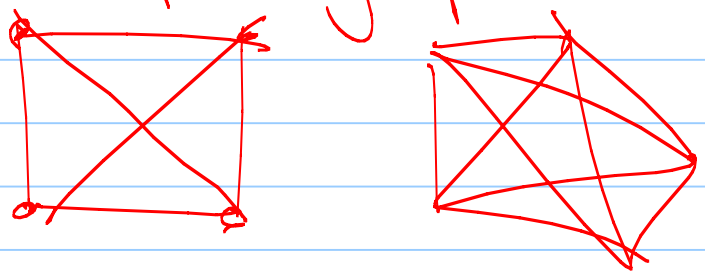
Sizes of $|V|$ & $|E|$

We usually let $n = |V|$ and $m = |E|$.

How big can m be?

$v_1, \ldots, v_n$

Complete graphs



$$\binom{n}{2} = \frac{n!}{2!\,(n-2)!}$$

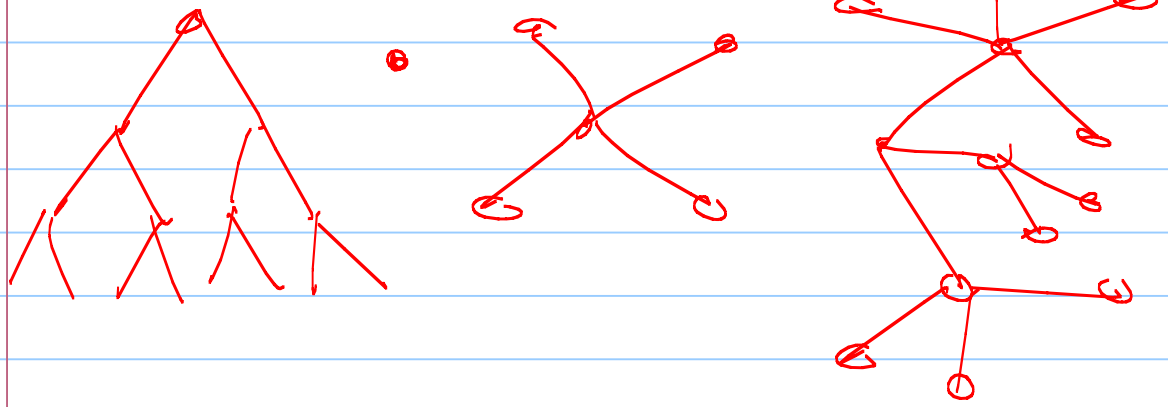
$$= \frac{n(n-1)}{2} \geq m$$

$$m = O(n^2)$$

Tree: A connected graph with no cycles.

(Note: No root in this definition!)

How many edges?



$m = n - 1$

↳ sparse graph

Graphs on a computer

How can we construct this data structure?

$G = (V, E)$

Vertex Lists

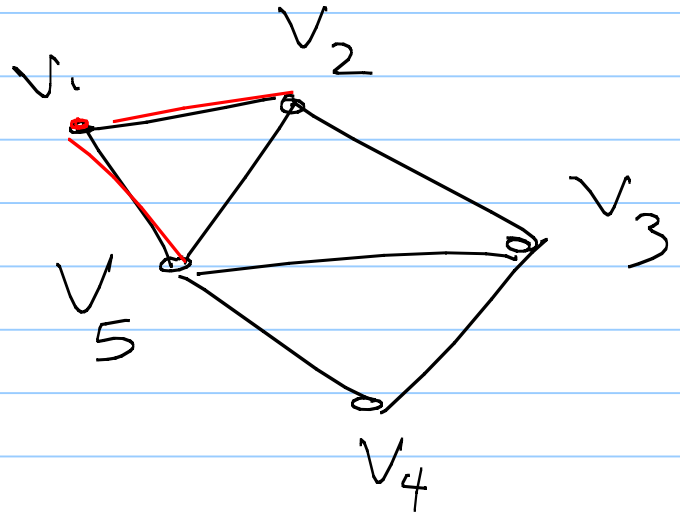
v_1 : v_2, v_5

v_2 : v_1, v_3, v_5

v_3 : v_2, v_4, v_5

v_4 : ⋮

v_5 :



size : $O(n+m)$

Check if v_i is neighbor of v_j : $O(n)$

Implementation

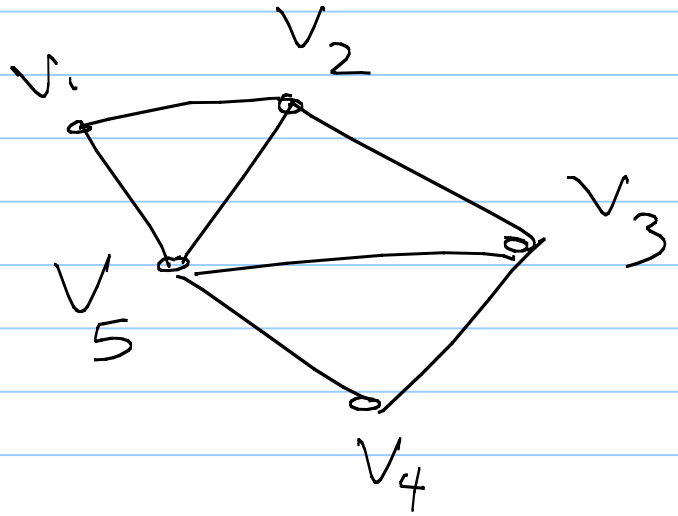
We call these vertex lists, but don't actually need lists.

Options:
- lists
- vectors
- balanced BST

Tradeoffs: insert versus lookup

Adjacency Matrix

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	1
v_2	1	0	1	0	1
v_3	0	1	0		
v_4	0	0		0	
v_5	1	1			0



Space: $O(n^2)$

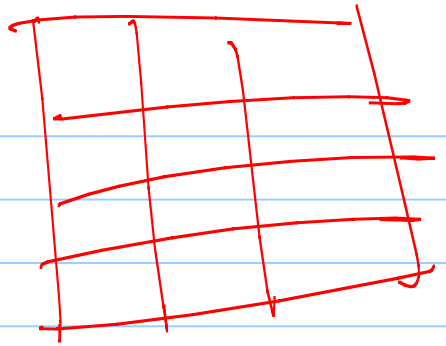
check neighbor: $O(1)$

Which is best?

Just depends.

Size versus lookup time.

Incidence Matrix

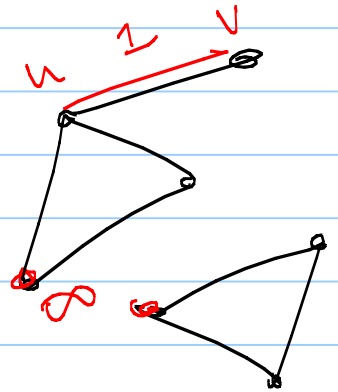


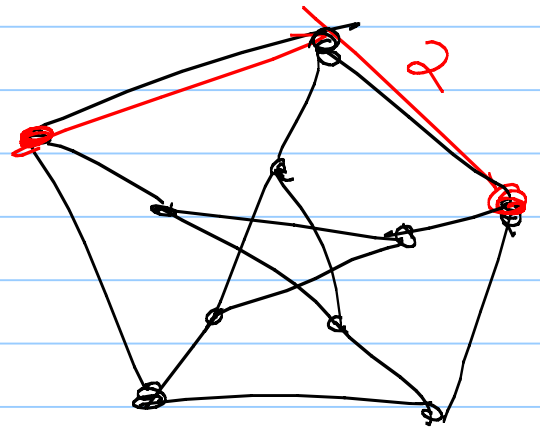
	e_1	...	e_m
v_1	0		
v_2	⋮		
⋮	0		
v_n	0		

$O(n \cdot m)$ space

Dfns

- G is connected if for all $u \neq v$, there is a path from u to v.
- The distance from u to v, $d(u,v)$, is equal to the length of the minimum u,v-path.





Algorithms on Graphs

Basic Question: Given 2 vertices, are they connected?

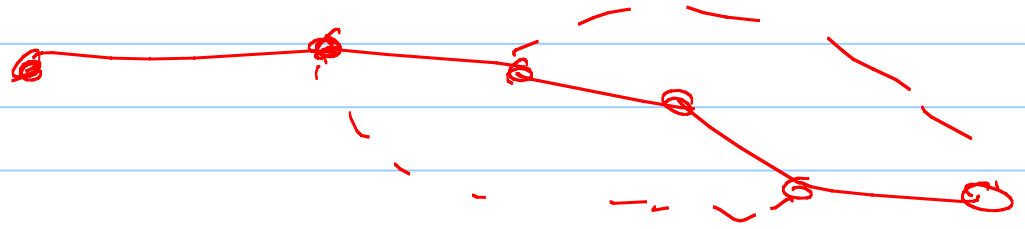
How to solve?

Search strategy

Suggestion:

- Suppose we're in a maze, searching for a treasure.

What do you do?



depth first search

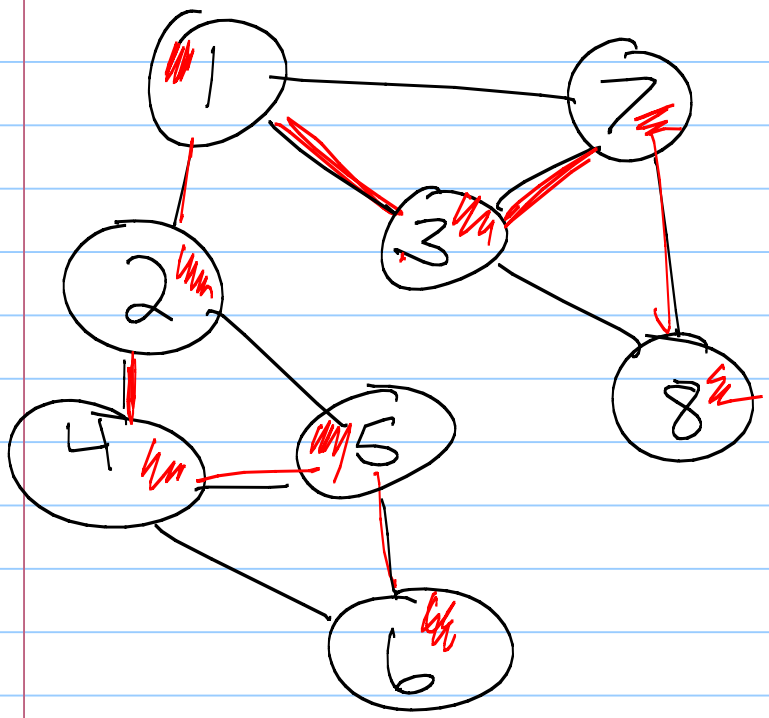
RecursiveDFS(u):

If u is unmarked:
- mark u
- for each edge $\{u, v\} \in E$
  - RecursiveDFS(v)

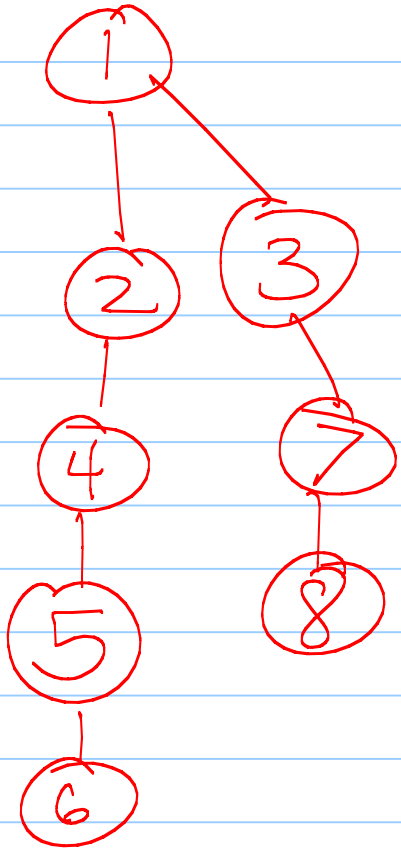
To check if s & t are connected, call DFS(s).

At end, if t is marked, return true

DFS (1)



DFS "tree"



Another version of DFS

```
IterativeDFS(u):
    create empty stack S
    S.push(u)

    while S is not empty:
        v ← S.pop
        if v is not marked
            mark(v)
            for each edge vw
                S.push(w)
```