More Graph Algorithms

Note Title 12/3/2013

Announcements

- Final HW up - due Monday
- No class tomorrow
  (also no office hours)

  Office hours Friday 9-10am

- Set review session: Friday at 2pm

Algorithms on Graphs

Basic Question: Given 2 vertices, are they connected?

How to solve?

Search strategy

depth first search

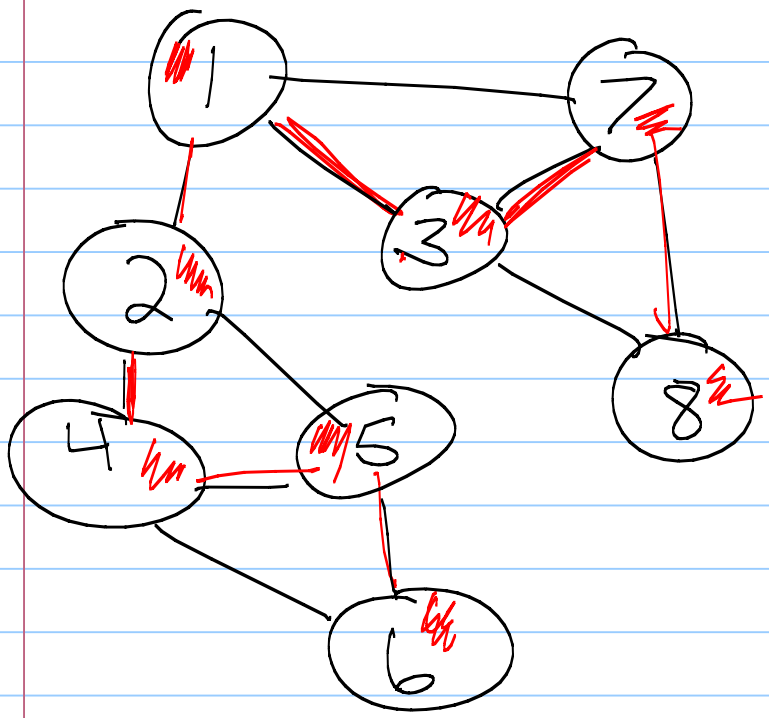
RecursiveDFS(u):

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

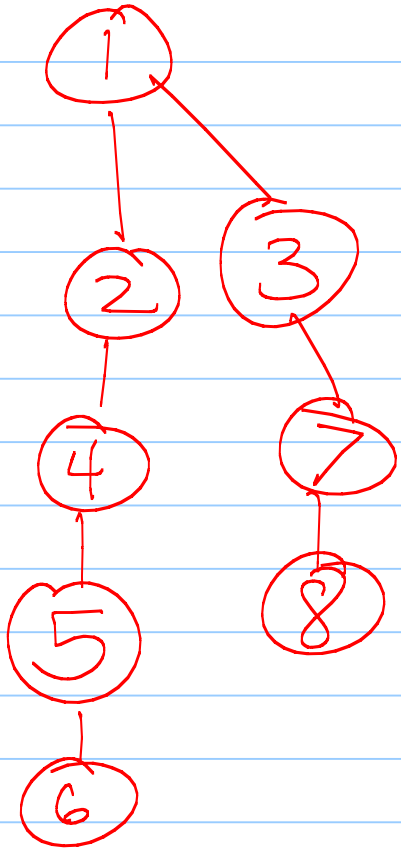
To check if s & t are connected, call DFS(s).

At end, if t is marked, return true

DFS (1)



DFS "tree"



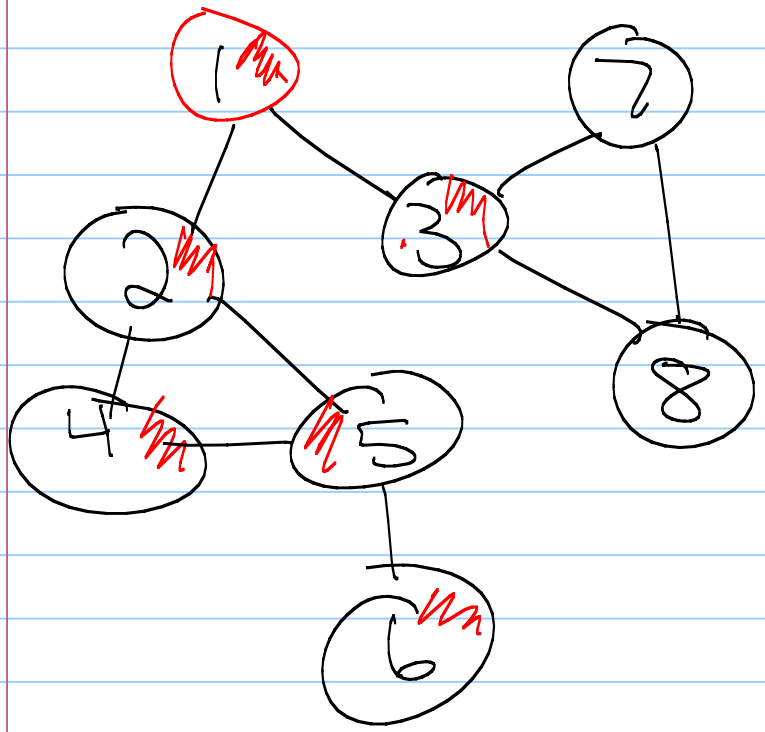
Another version of DFS

```
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)
```

stack!

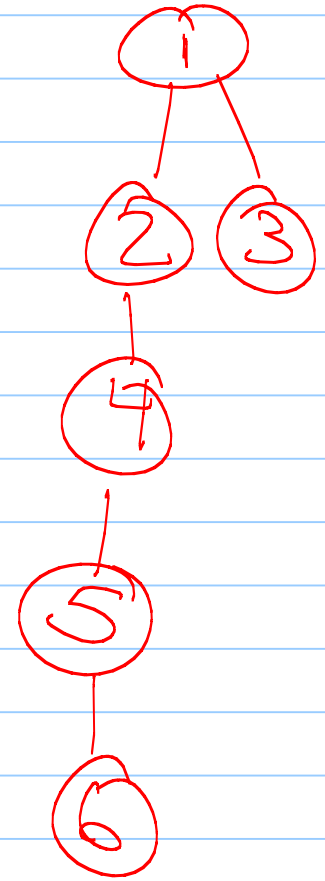
Iterative DFS (1):



Stack:

1
7
8

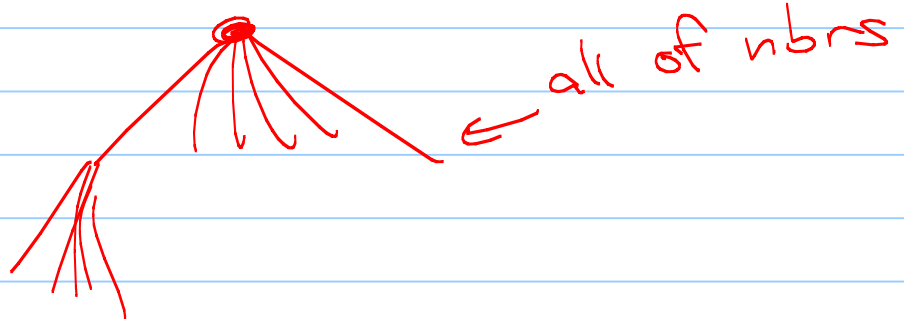
Stack



BFS: Breadth first search

Instead of a stack, could push all the neighbors on a queue!

So from S all of S's neighbors will connect to it.



Iterative BFS(u)

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

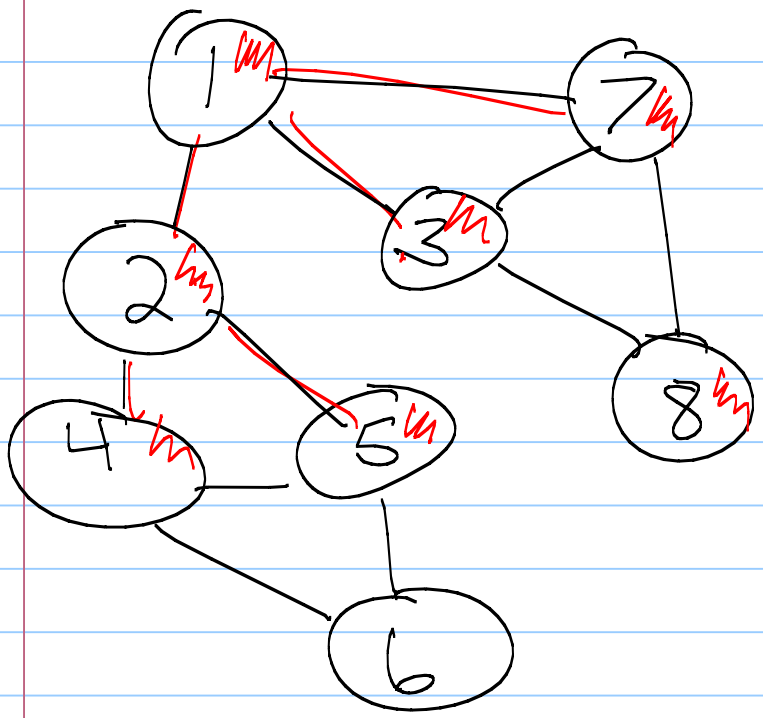
front

Queue: ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 1 3 8

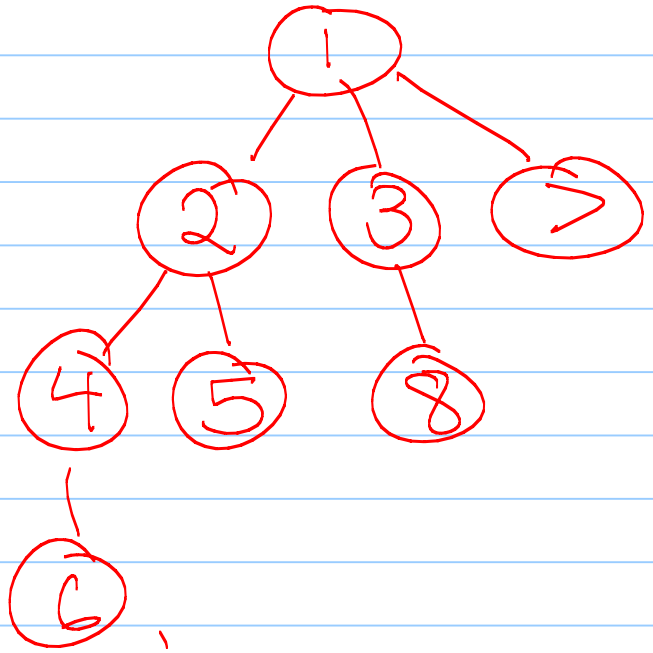
back

2 5 6 2 4 6

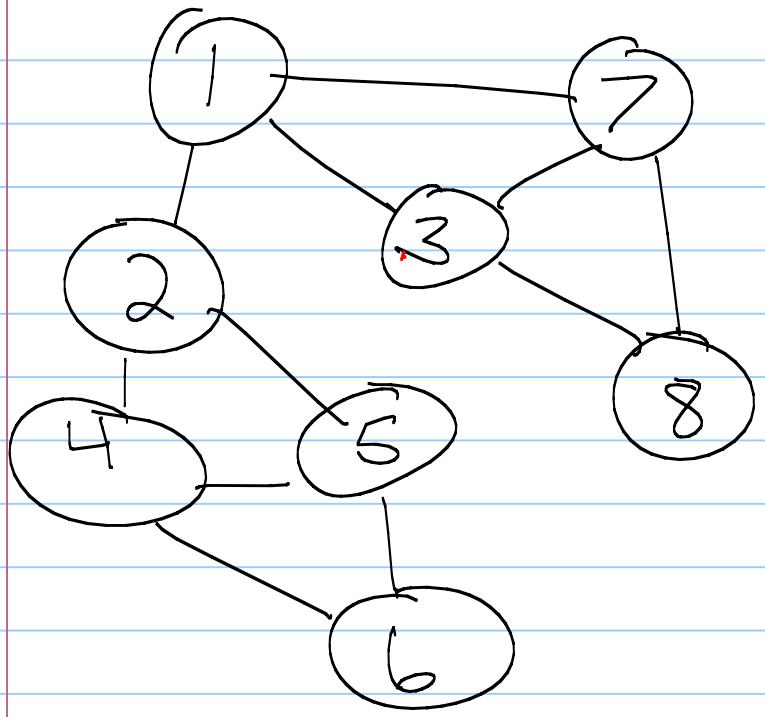
BFS (1):



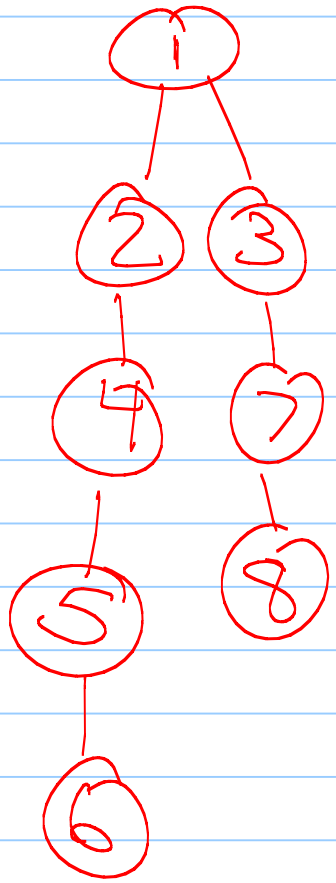
BFS "tree":



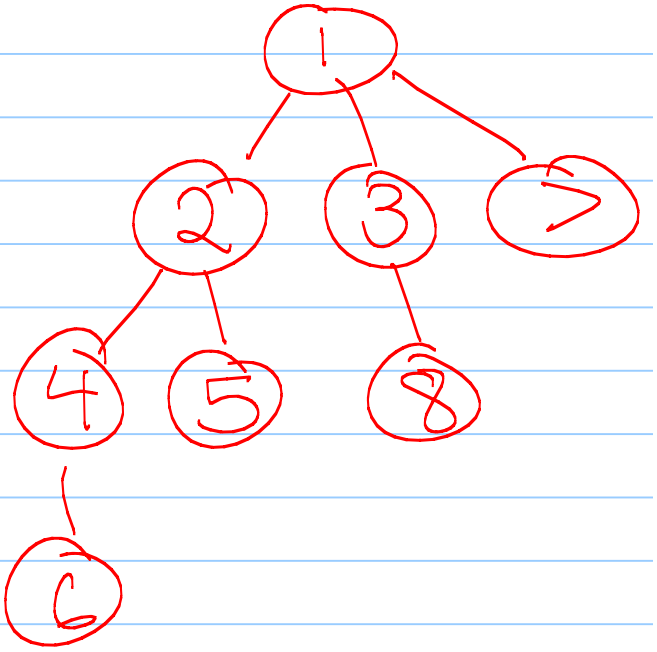
Spanning trees:



DFS:



BFS:



BFS versus DFS

- Both can tell if 2 vertices are connected
- Both can be used to detect cycles.
  How?

  Any "extra" edge forms a cycle.
- Difference is structure of trees

Runtimes:

$m = |E|$
$n = |V|$

- each push/pop is $O(1)$
- must visit each node (mark) $\Big]\ O(n)$

→ Each node v is pushed (& popped) $d(v)$ times

$$\sum_v d(v) = 2m \quad \text{(degree sum formula)}$$
$$= O(m)$$

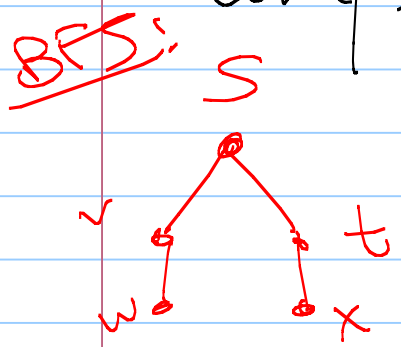
$$\Rightarrow O(m+n) \quad [\text{in general, } O(m)]$$

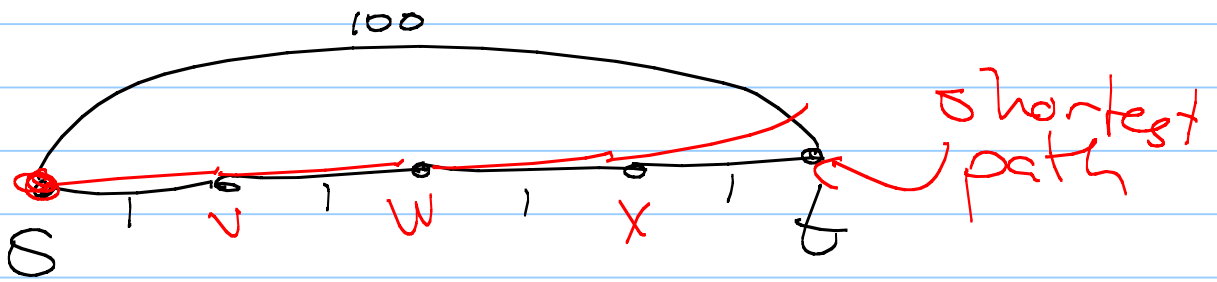
Other graph algorithms

- BFS returns a "short" s-t path, in some sense.

But won't work if graph has weights on the edges.

Why?





Which s-t path will be in BFS tree?

Shortest path trees

Given a weighted graph, find shortest path from s to t.

↳ Find all shortest paths from s to all other vertices

Uses?

- Road networks
- ⋮

Algorithms to solve this actually solve a more general problem:
Find shortest path from s to every other vertex.

Called the shortest path tree rooted at s.

Can be computed in polynomial time.

Key: use a heap

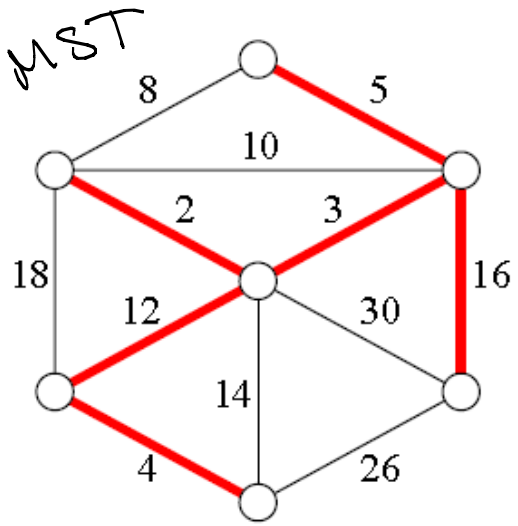
Another question:

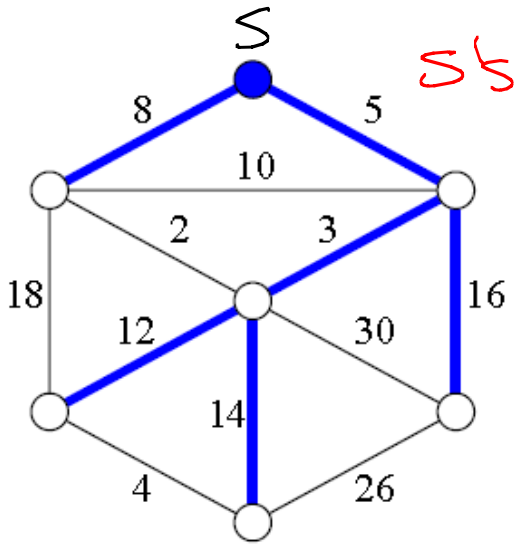
Given G, find a tree containing every vertex with minimum total weight.

Uses? - Network planning

This is called the minimum spanning tree of G.

Note: Not the same as shortest path tree!





s's shortest path tree

Review Session:

Thursday

9 am

~~10 am~~

~~11 am~~

Friday

~~9 am~~

~~10 am~~

~~11 am~~

2 pm

3 pm

4 pm