CS 314 - BFS

Note Title 1/22/2010

Announcements

- HW due Friday
  (you should have already started!)

HW0: average ~28

Regrades - due within 2 weeks

How big can m be (in terms of n)?

Representing Graphs

n = # vertices
m = # of edges

How can we record a graph?

① Adjacency matrix

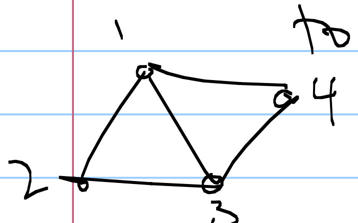
$$\begin{array}{c|ccc} & v_1 & \cdots & v_n \\ \hline v_1 & & & \\ \vdots & 1 & 0 & \\ v_n & & & \end{array}$$

1 means v_i & v_j connected

0 not connected

space: $O(n^2)$

② Adjacency Lists:
each vertex gets
a list of the other
vertices it connects to



1 : 2, 3, 4
2 : 1, 3

$$\sum_{i=1}^{n} d(v_i) = 2m$$

Space: $O(n+m)$

Algorithms on graphs

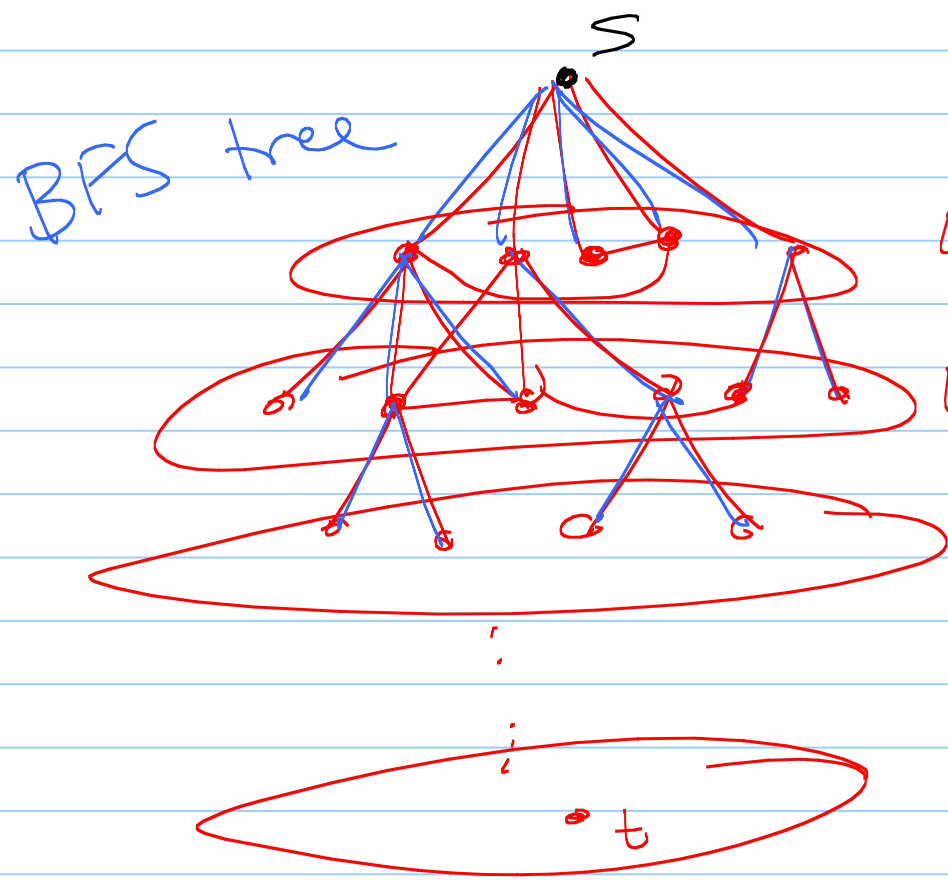
Basic question: given 2 nodes, are they connected?

s-t connectivity: is there a path from s to t in G?

How? Search out from s until we find t, or can't search anymore

Breadth First Search (BFS)

Who is connected to s?



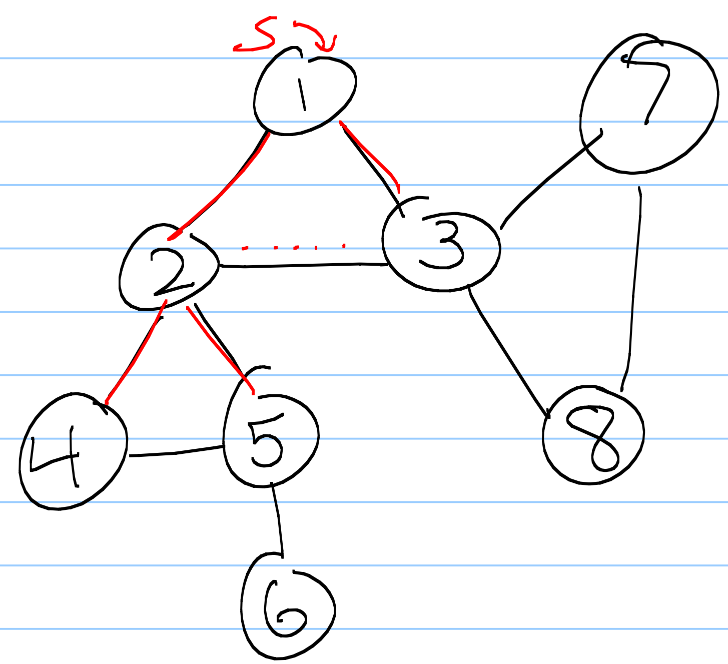
L_1 - at distance 1 from s

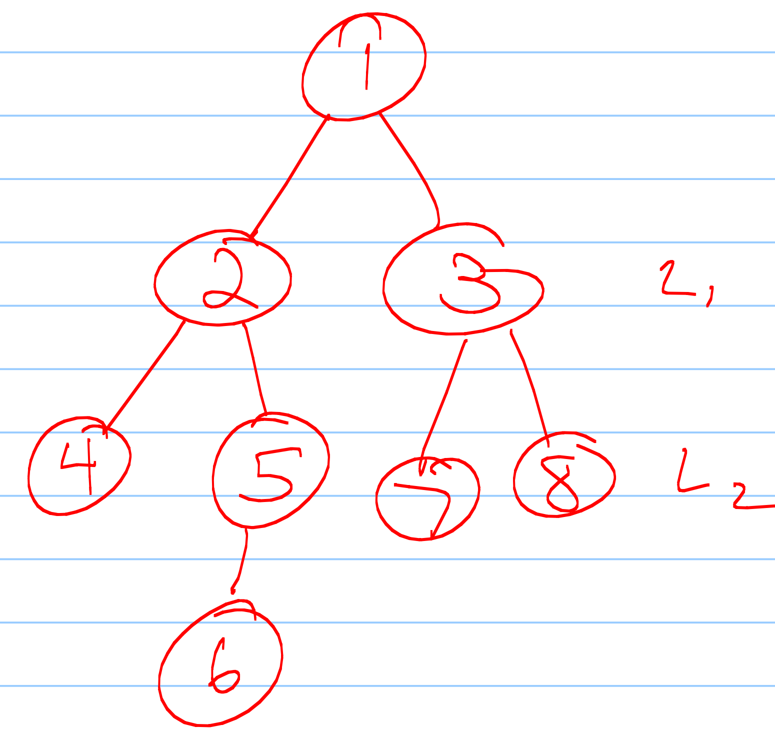
L_2 - at distance 2 from s

L_3 - at distance 3

L_i

Exercise: What does the BFS tree look like for this graph?





$p \longleftrightarrow q$

<u>Property</u>: For each $j \geq 1$, layer L_j in the BFS tree consists of all nodes at distance j from s.
There is a path from s to t if & only if t is in the BFS tree.

"<u>pf</u>": Path from a vertex in L_j goes through a vertex in L_{j-1}.
→ use induction on j

if path from s to t, then t in BFS tree.
why? t will be in level L_j, where j = distance from s to t
if t in BFS tree, can follow tree "up" from t & will end at s

Pseudo code: (from text)

```
BFS(G = (V,E), s):
    Create array Discovered[1..n], initialized to false
    Discovered[s] ← true
    Initialize list L[0] to be single element s
    Counter ← 0
    BFStree ← ∅
    While L[i] ≠ ∅
        Initialize empty list L[i+1]
        For each u ∈ L[i]                      every vertex
            For each edge e = {u,v}            every adjacent vertex / adjacent to u, let v be other endpt
                If Discovered[v] = false
                Discovered[v] ← true
                Add {u,v} to BFStree
                Add v to L[i+1]
        i ← i+1
```

Runtime:

Nested for loops $- O(n^2)$

Careful:

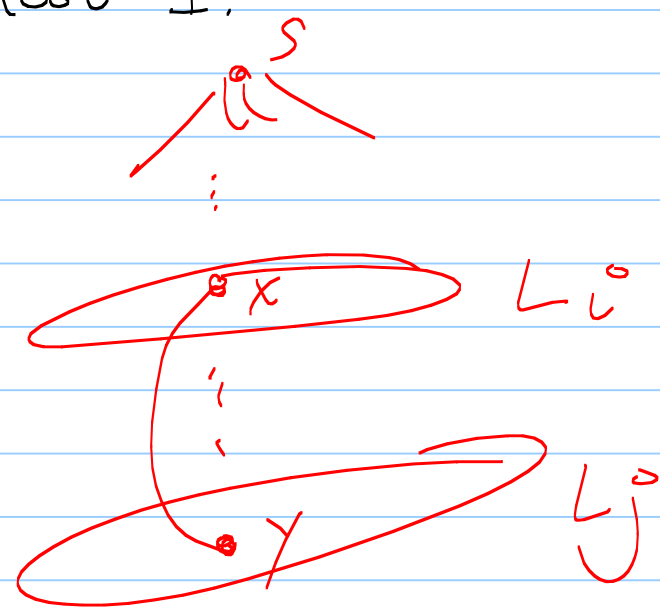
$$\sum_{v \in V} O(1) \cdot d(v) = O(\overset{\#\text{ edges}}{m}) = 2|E| \cdot O(1)$$

For each vertex, look at all adjacent edges & do $O(1)$ per edge.

Handy property:

Let T be a BFS tree of G, + let $x, y \in V(G)$ belong to layers L_i + L_j, respectively. If $\exists \{x, y\} \in E(G)$, then i + j differ by at most 1.

Proof:



Say w.l.o.g. $i \leq j$.

If i is more than 1 less than j then y would have been put in level L_{i+1}