COL 106 Lecture 33

Topic : Breadth-First Traversal

Operation	Vertex and Edge lists	Adjacency Matrix	Adjacency Lists
addVertex(v)	1	n (amortised)	1
addEdge(u,v)	1	1	1. (if u,v edge is guaranteed to not exist already)
delEdge(u,v)	m	1	$d_u + d_v$
delVertex(v)	m	depends!	$\sum_{u \text{ nbr of } v} d_u$
			Can be improved to d_v with a little trick. How?

Some interesting computational problems

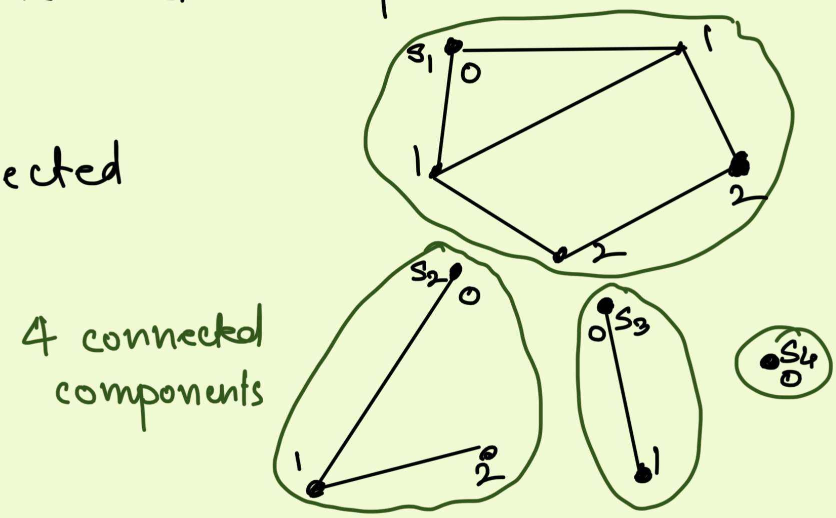
Given a graph G:

1. Reachability: Does there exist a path between u and v?

2. Find a shortest path between u and v (if a path exists).

3. Connectivity: Does there exist a path between every pair of vertices?

4. Identify the connected components of G.



Algorithm for Connectivity (?)

Input: G, u, v

$q \leftarrow$ Empty queue
q.Enqueue(u)
while (q is not empty):
    $x \leftarrow$ q.Dequeue()
    If $x = v$ then return True
    Enqueue all neighbors of x into q.
return False

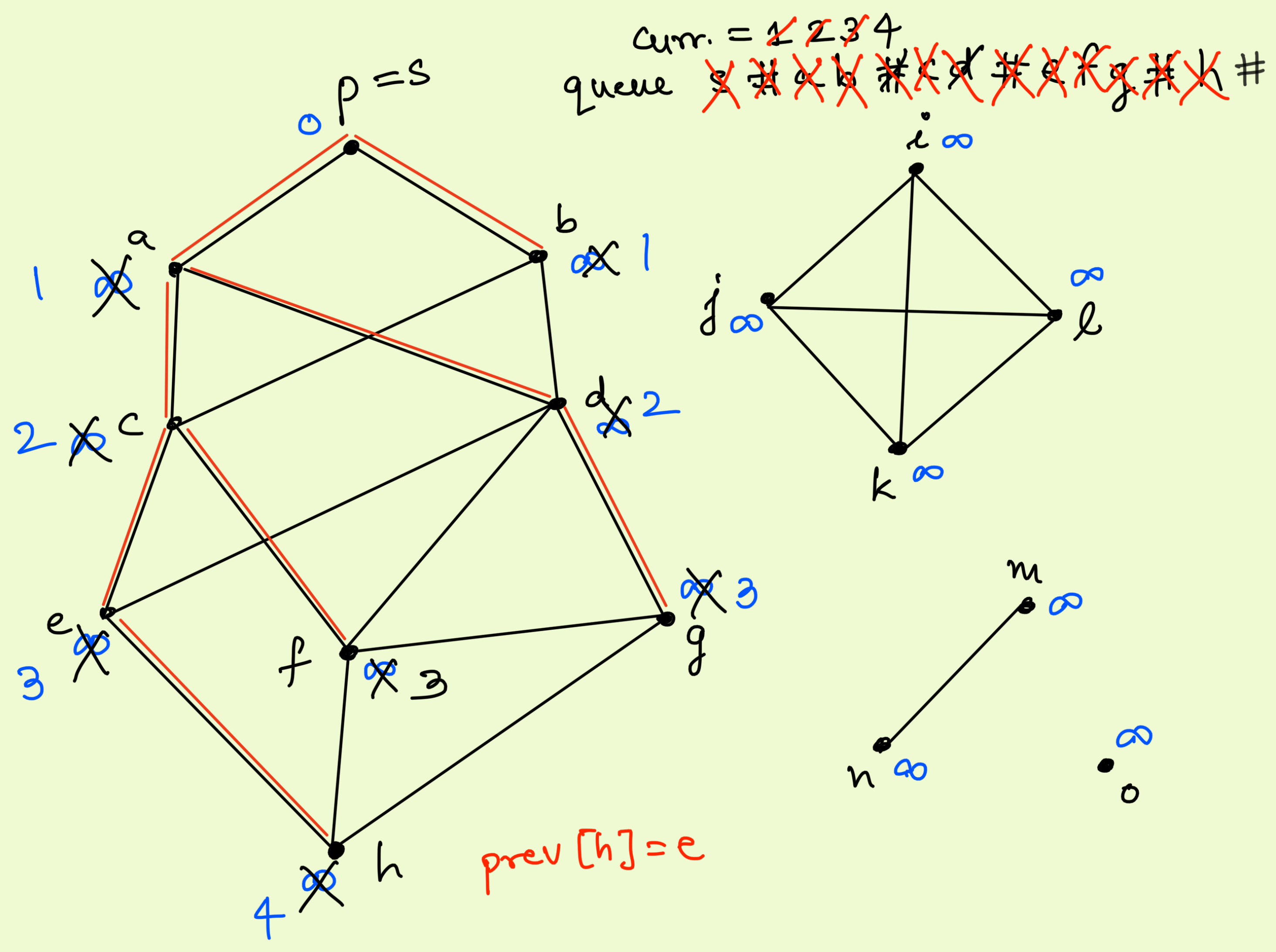
Doesn't work! Runs forever.

Idea: Enqueue every vertex at most once.

A Unifying Problem
(a.k.a. Single Sourch Shortest Path Problem)

Given a graph G and a vertex s of G

Label each t with the distance of a shortest s-t path (∞ if no such path exists.)



Breadth – First Traversal

BFT (G, s):

$q \leftarrow$ Empty queue,

$d[v] \leftarrow \begin{cases} 0 & \text{for } v = s \\ \infty & \text{otherwise} \end{cases} \longrightarrow O(n)$

curr $\leftarrow 1$

Enqueue s followed by # into q.

while (TRUE):

$x \leftarrow q$.Dequeue()

If x is #:

If q is empty
break

$O(1)$

curr $\leftarrow$ curr+1; Enqueue # into q

$\leq$ # vertices in the connected component of S (overall iterations)

Else

For each nbr y of x such that $d[y] = \infty$

d_x iterations

$d[y] \leftarrow$ curr

prev $[y] \leftarrow x$

Enqueue y into q.

$O(1)$

Let n' = # vertices and m' = # edges in the connected component of s.

Time Complexity: $O\left(n' + \sum_{x \in \text{conn. comp. of } s} d_x\right) = O(n' + m')$, because

$\sum_{x \in \text{conn. comp. of } s} d_x = 2m'$, by Handshake Lemma.