Design and Analysis
of Algorithms I

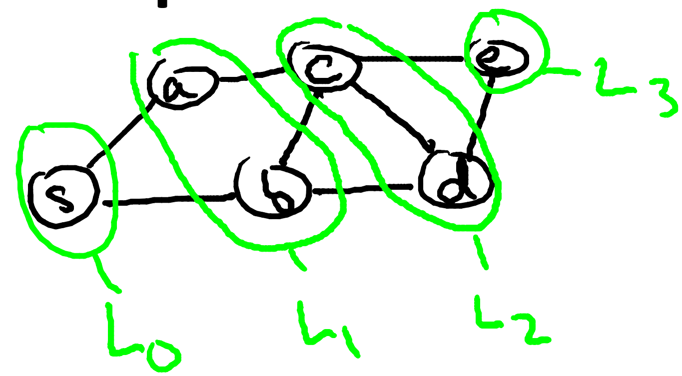
Graph Primitives

Breadth-First Search

Overview and Example

Breadth-First Search (BFS)

- explore nodes in "layers"
- can compute shortest paths
- connected components of undirected graph



Run time: $O(m+n)$ (linear time)

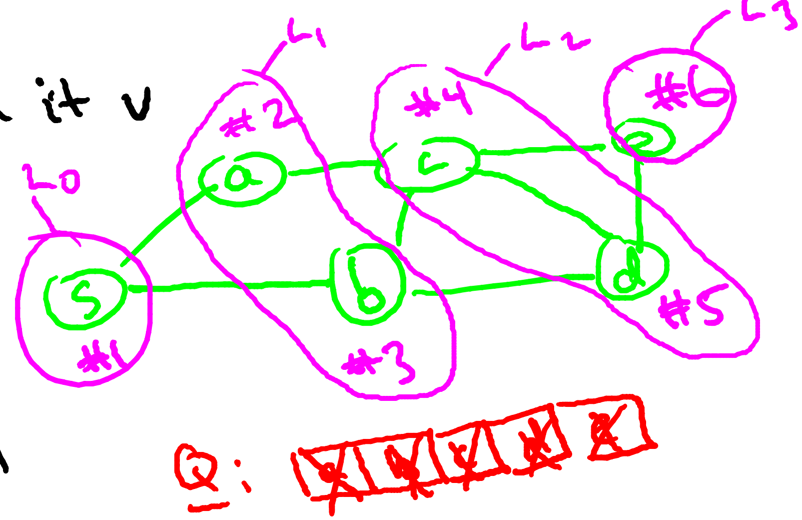
The Code

BFS (graph G, start vertex s)

[all nodes initially unexplored]

- mark s as explored
- let Q = queue data structure (FIFO), initialized with s
- while Q ≠ ϕ :
  - remove the first node of Q, call it v
  - for each edge (v, w) :
    - if w unexplored
      - mark w as explored
      - add w to Q (at the end)

O(1) time



Basic BFS Properties

Claim #1: at the end of BFS, v explored $\Longleftrightarrow$ G has a path from s to v.

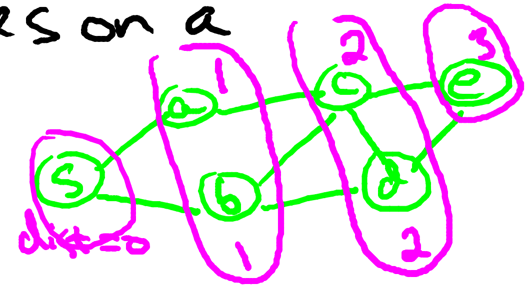
Reason: special case of the generic algorithm.

Claim #2: running time of main while loop $= O(n_s + m_s)$, where n_s = # of nodes reachable from s
m_s = # of edges "

Reason: by inspection of code.

Application: Shortest Paths

Goal: compute dist(v), the fewest # of edges on a path from s to v.



Extra code: - initialize $\text{dist}(v) = \begin{cases} 0 & \text{if } v = s \\ +\infty & \text{if } v \neq s \end{cases}$

- when considering edge (v,w):
  - if w unexplored, then set dist(w) = dist(v)+1.

Claim: at termination, dist(v) = i $\Leftrightarrow$ v in ith layer

(i.e., $\Leftrightarrow$ shortest s-v path has i edges).

Proof idea: every layer-i node w is added to Q by a layer-(i-1) node v via the edge (v,w).

Tim Roughgarden

Application: Undirected Connectivity

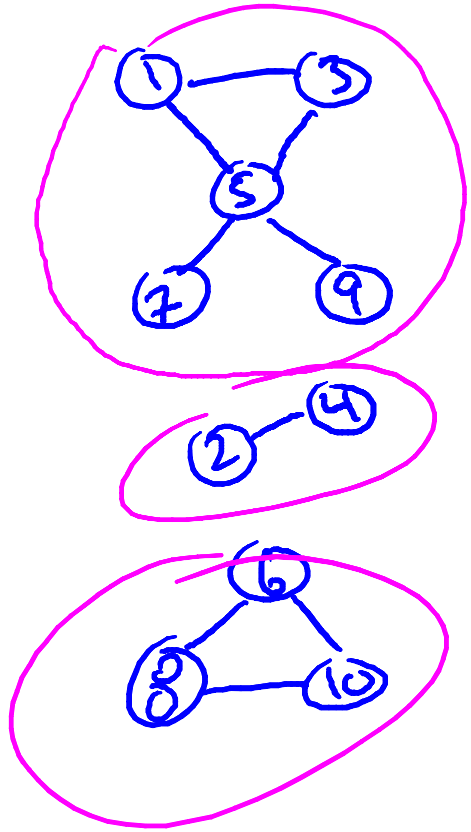
Let $G=(V,E)$ be an undirected graph.

Connected components = the "pieces" of G.

Formal definition: equivalence classes of the relation $u \sim v \Leftrightarrow \exists$ u-v path in G. [check: $\sim$ is an equivalence relation]

Goal: Compute all connected components.

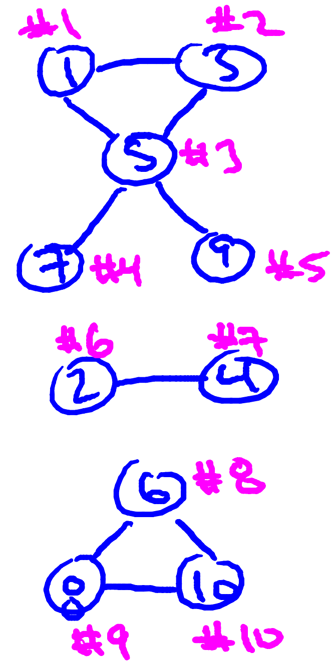
Why? - check if network is disconnected
- graph visualization - clustering



Connected Components via BFS

To compute all components: (undirected case)

- all nodes unexplored $O(n)$
  [assume labelled 1 to n]
- for $i = 1$ to n $O(n)$
  - if i not yet explored $O(n)$ [in some previous BFS]
    - BFS(G, i) → discovers precisely i's connected component



Note: finds every connected component.

Running time: $O(m + n)$

$O(1)$ per node

$O(1)$ per edge in each BFS