Design and Analysis of Algorithms I

Graph Primitives

Introduction to Graph Search

A Few Motivations

1. Check if a network is connected (can get to anywhere from anywhere else)

Bacon number = 2

2. Driving directions
3. Formulate a plan [e.g., how to fill in a Sudoku puzzle]
   -- nodes = a partially completed puzzle -- arcs = filling in one new sequence
4. Compute the “pieces” (or “components”) of a graph
   -- clustering, structure of the Web graph, etc.

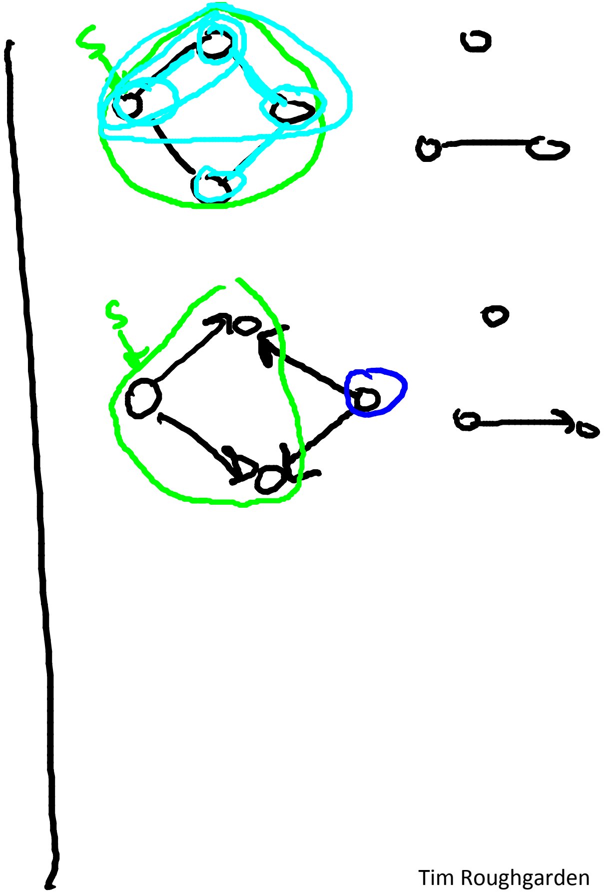
Tim Roughgarden

Generic Graph Search

Goals : 1) find everything findable from a given start vertex
2) don't explore anything twice

Goal:
O(m+n) time

Generic Algorithm (given graph G, vertex s)
-- initially s explored, all other vertices unexplored
-- while possible : (if none, halt)
    -- choose an edge (u,v) with u explored and v unexplored
    -- mark v explored



Generic Graph Search (con'd)

<u>Claim</u> : at end of the algorithm, v explored <==> G has a path from s to v

(G undirected or directed)

<u>Proof</u> : (=>) easy induction on number of iterations (you check)

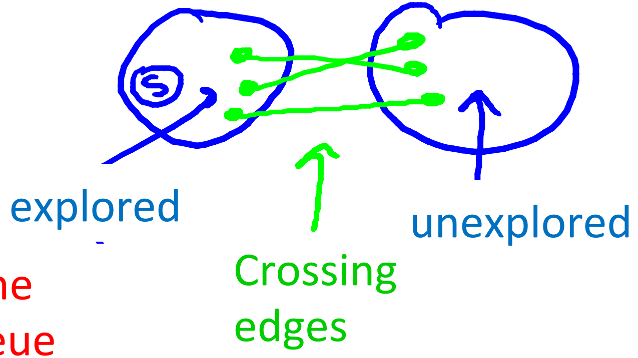
(<=) By contradiction. Suppose G has a path P from s to v:

But v unexplored at end of the algorithm. Then there exists an edge (u,x) in P with u explored and x unexplored.

But then algorithm would not have terminated, contradiction. Q.E.D.

BFS vs. DFS

Note : how to choose among the possibly many "frontier" edges ?



Breadth-First Search (BFS)

-- explored nodes in "layers"

-- can compute shortest paths

-- can compute connected components of an undirected graph

O(m+n) time using a queue (FIFO)

Depth-First Search (DFS)

O(m+n) time using a stack (LIFO) (or via recursion)

-- explore aggressively like a maze, backtrack only when necessary

-- compute topological ordering of a directed acyclic graph

-- compute connected components in directed graphs