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 direction

(3) formulate a plan [e.g., how to fill in a Sudoku puzzle]

- nodes = a partially completed puzzle - arcs = filling in one new square

(4) compute the "pieces" (or "components") of a graph

- clustering, structure of the web graph, etc.

Generic Graph Search

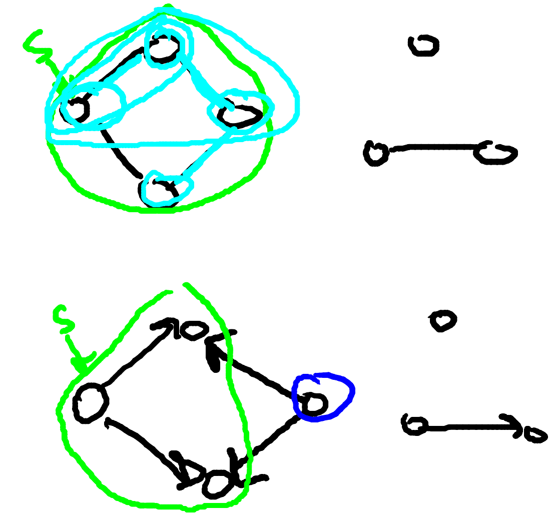
Goals: (1) find everything findable from a given start vertex

(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:
  - choose an edge (u,v) with u explored and v unexplored (if none, halt)
  - mark v explored



Tim Roughgarden

Generic Graph Search (con'd)

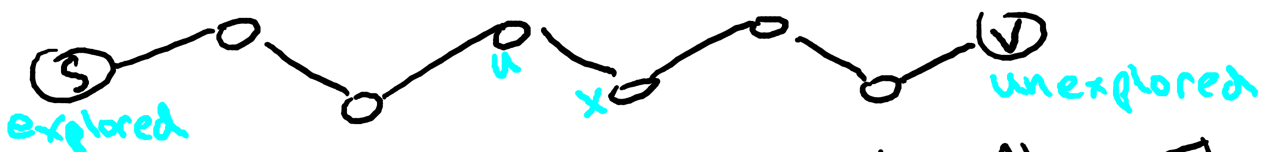
Claim: at end of the algorithm, v explored $\Longleftrightarrow$ G has a path from s to v.

(G undirected or directed)

Proof: ($\Rightarrow$) easy induction on number of iterations (you check).

($\Leftarrow$) By contradiction. Suppose G has a path P from s to v:

P:

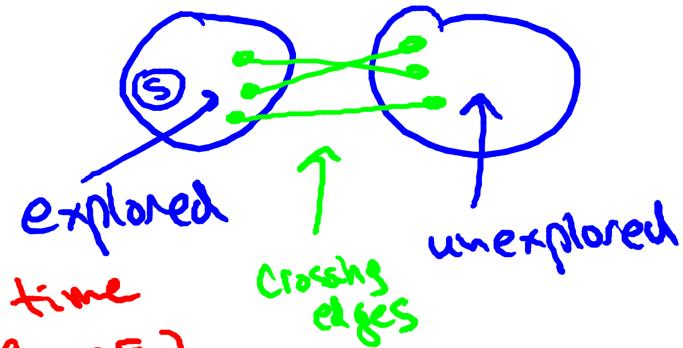


but v unexplored at end of the algorithm. Then $\exists$ an edge $(u,w) \in P$ with u explored and w unexplored.

But then algorithm would not have terminated, contradiction. QED!

BFS vs. DFS

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



Breadth-First Search (BFS)

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

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

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 directed acyclic graph
- compute connected components in directed graphs

Tim Roughgarden