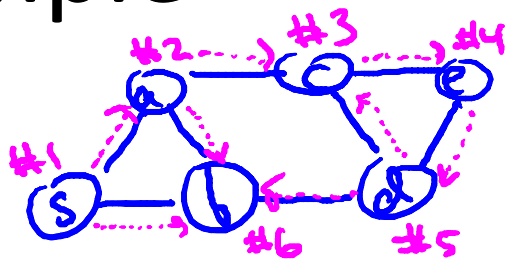
Design and Analysis of Algorithms I

Graph Primitives

Depth-First Search

Overview and Example



Depth-First Search (DFS): explore aggressively only backtrack when necessary.

- also computes a topological ordering of a directed acyclic graph
- and strongly connected components of directed graphs

Run Time: $O(m+n)$

The Code

Exercise: mimic BFS code, use a stack instead of a queue [+ minor other modifications].

Recursive version: DFS (graph G, start vertex s)

- mark s as explored
- for every edge (s,v):
  - if v unexplored
    - DFS(G,v)

Basic DFS Properties

Claim #1: at end of the algorithm, v marked as explored $\iff$ $\exists$ path from s to v in G.

Reason: particular instantiation of generic search procedure.

Claim #2: running time is $O(n_s + m_s)$

n_s: # of nodes reachable from s

m_s: # of edges reachable from s

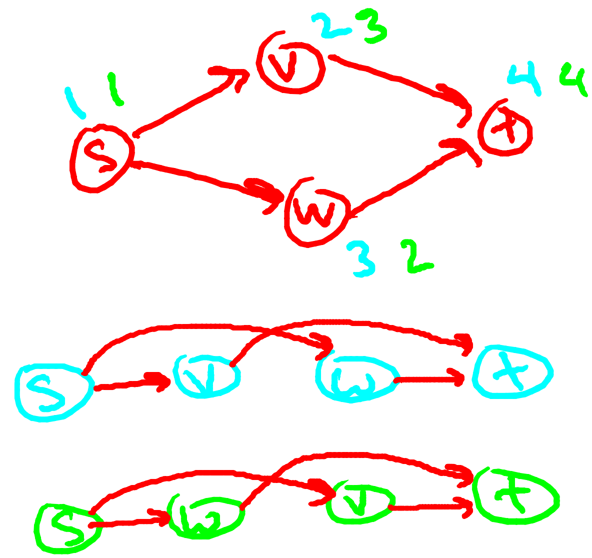
Reason: look at each node in connected component of s at most once, each edge at most twice.

Application: Topological Sort

Definition: A topological ordering of a directed graph G is a labelling f of G's nodes such that:

(1) the $f(v)$'s are the set $\{1, 2, \ldots, n\}$

(2) $(u,v) \in G \Rightarrow f(u) < f(v)$



Motivation: sequence tasks while respecting all precedence constraints.

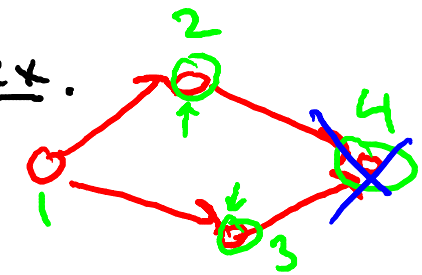
Note: G has directed cycle $\Rightarrow$ no topological ordering.

Theorem: no directed cycle $\Rightarrow$ can compute topological ordering in $O(m+n)$ time.

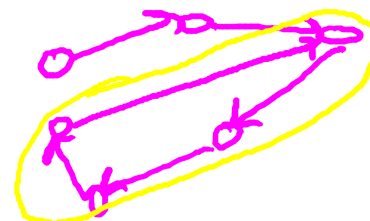
Tim Roughgarden

Straightforward Solution

Note: every directed acyclic graph has a sink vertex.



Reason: if not, can keep following outgoing arcs to produce a directed cycle.



To compute topological ordering:

- let v be a sink vertex of G
- set f(v) = n
- recurse on G − {v}

Why does it work?: when v is assigned to position i, all outgoing arcs already deleted ⟹ all lead to later vertices in ordering.

Tim Roughgarden

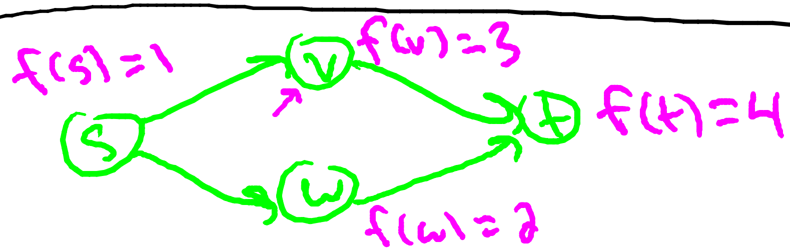
Topological Sort via DFS (Slick)

DFS-Loop (graph G)

- mark all nodes unexplored
- current_label = n [to keep track of ordering]
- for each vertex $v \in G$
  - if v not yet explored [in previous DFS call]
    - DFS(G,v)

DFS (graph G, start vertex s)

- for every edge (s,v)
  - if v not yet explored
    - mark v explored
    - DFS(G,v)
- set f(s) = current_label
- current_label--

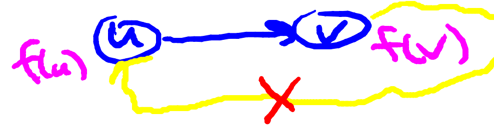


Topological Sort via DFS (con'd)

Running Time: $O(m+n)$.

Reason: $O(1)$ time per node, $O(1)$ time per edge.

Correctness: need to show that if (u,v) is an edge, then $f(u) < f(v)$.



(since no directed cycles)

Case 1: u visited by DFS before v. $\Rightarrow$ recursive call corresponding to v finishes before that of u (since DFS).
$\Rightarrow f(v) > f(u)$

Case 2: v visited before u. $\Rightarrow$ v's recursive call finishes before u's even starts. $\Rightarrow f(v) > f(u)$.   QED!