OLD CS 473: Fundamental Algorithms, Spring 2015

DFS in Directed Graphs, Strong Connected Components, and DAGs

Lecture 2

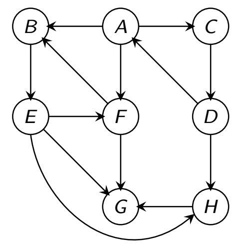
January 22, 2015

Strong Connected Components (SCCs)

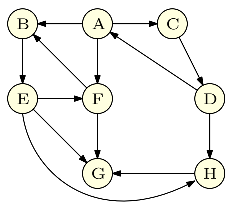
Algorithmic Problem

Find all SCCs of a given directed graph.

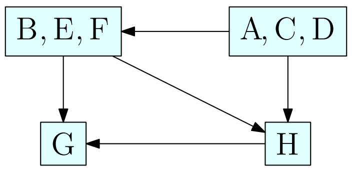
Previous lecture:
Saw an $O(n \cdot (n+m))$ time algorithm.
This lecture: $O(n+m)$ time algorithm.



Graph of SCCs



Graph G



G^{SCC}: Graph of SCCs

Meta-graph of SCCs

Let $S_1, S_2, \ldots S_k$ be the strong connected components (i.e., SCCs) of G. The graph of SCCs is G^{SCC}

1. Vertices are $S_1, S_2, \ldots S_k$
2. There is an edge (S_i, S_j) if there is some $u \in S_i$ and $v \in S_j$ such that (u, v) is an edge in G.

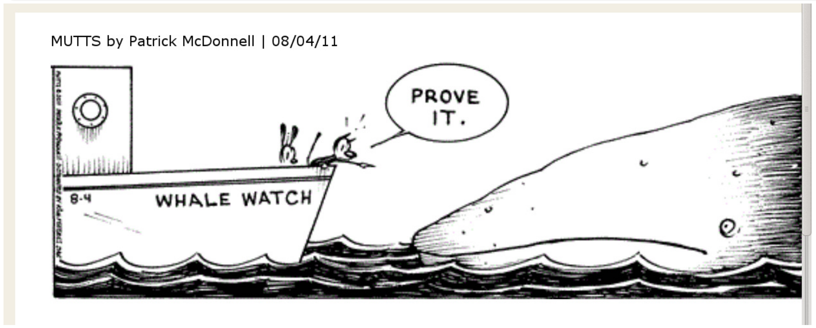
Reversal and SCCs

Proposition

For any graph G, the graph of SCCs of G^{rev} is the same as the reversal of G^{SCC}.

Proof.

Exercise. □



SCCs and DAGs

Proposition

For any graph G, the graph G^{SCC} has no directed cycle.

Proof.

If G^{SCC} has a cycle $S_1, S_2, \ldots, S_k$ then $S_1 \cup S_2 \cup \cdots \cup S_k$ should be in the same SCC in G. Formal details: exercise. □

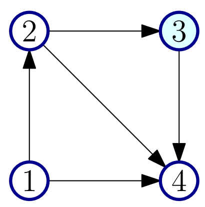
Part I

Directed Acyclic Graphs

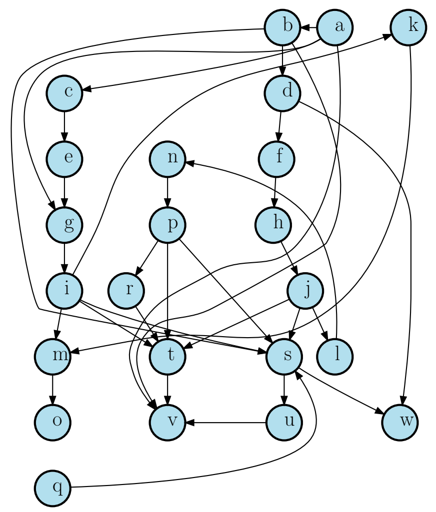
Directed Acyclic Graphs

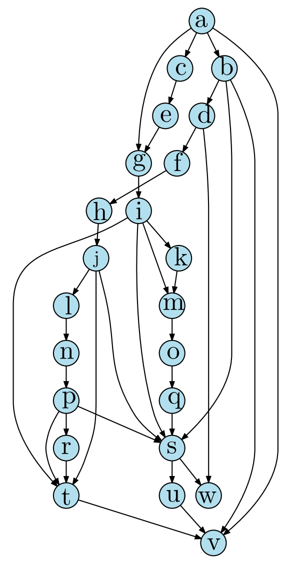
Definition

A directed graph G is a **directed acyclic graph** (DAG) if there is no directed cycle in G.

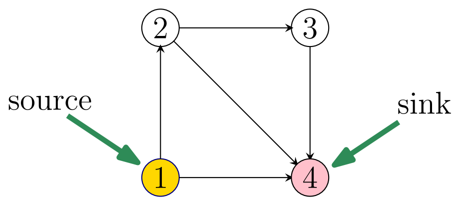


Is this a DAG?





Sources and Sinks



Definition

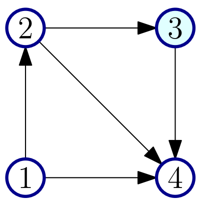
1. A vertex u is a **source** if it has no in-coming edges.
2. A vertex u is a **sink** if it has no out-going edges.

Simple DAG Properties

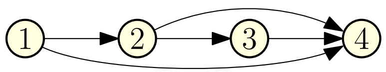
1. Every DAG G has at least one source and at least one sink.
2. If G is a DAG if and only if G^{rev} is a DAG.
3. G is a DAG if and only each node is in its own strong connected component.

Formal proofs: exercise.

Topological Ordering/Sorting



Graph G



Topological Ordering of G

Definition

A **topological ordering**/**topological sorting** of $G = (V, E)$ is an ordering $\prec$ on V such that if $(u, v) \in E$ then $u \prec v$.

Informal equivalent definition:

One can order the vertices of the graph along a line (say the x-axis) such that all edges are from left to right.

DAGs and Topological Sort

Lemma

A directed graph G can be topologically ordered iff it is a DAG.

Proof.

$\Longrightarrow$: Suppose G is not a DAG and has a topological ordering $\prec$. G has a cycle $C = u_1, u_2, \ldots, u_k, u_1$.
Then $u_1 \prec u_2 \prec \ldots \prec u_k \prec u_1$!
That is... $u_1 \prec u_1$.
A contradiction (to $\prec$ being an order).
Not possible to topologically order the vertices. $\square$

DAGs and Topological Sort

Lemma

A directed graph G can be topologically ordered iff it is a DAG.

Continued.

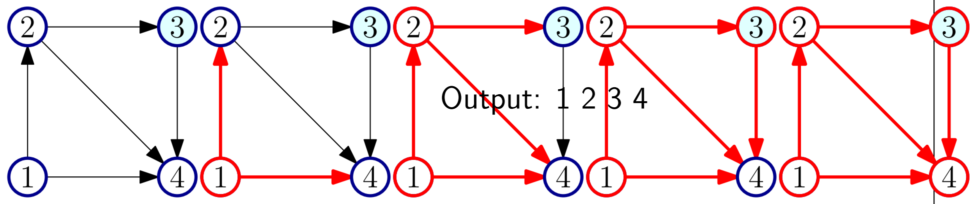
$\Leftarrow$: Consider the following algorithm:

1. Pick a source u, output it.
2. Remove u and all edges out of u.
3. Repeat until graph is empty.
4. Exercise: prove this gives an ordering.

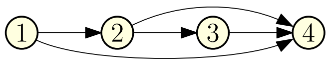
□

Exercise: show above algorithm can be implemented in $O(m + n)$ time.

Topological Sort: An Example



Topological Sort: Another Example



DAGs and Topological Sort

Note: A DAG G may have many different topological sorts.

Question: What is a DAG with the most number of distinct topological sorts for a given number n of vertices?

Question: What is a DAG with the least number of distinct topological sorts for a given number n of vertices?