16.4.2

DFS and cycle detection: Topological sorting using DFS

DFS

DFS

Cycles in graphs

Question: Given an undirected graph how do we check whether it has a cycle and output one if it has one?

Question: Given an directed graph how do we check whether it has a cycle and output one if it has one?

Cycles in graphs

Question: Given an undirected graph how do we check whether it has a cycle and output one if it has one?

Question: Given an directed graph how do we check whether it has a cycle and output one if it has one?

Cycle detection in directed graph using topological sorting

Question

Given G, is it a DAG?

If it is, compute a topological sort.
If it failes, then output the cycle C.

Topological sort a graph using DFS...

And detect a cycle in the propcesss

DFS based algorithm:

1. Compute **DFS**(G)
2. If there is a back edge $e = (v, u)$ then G is not a DAG. Output cycle C formed by path from u to v in T plus edge (v, u).
3. Otherwise output nodes in decreasing post-visit order. Note: no need to sort, $DFS(G)$ can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in $O(n + m)$ time.
Correctness is not so obvious. See next two propositions.

Topological sort a graph using DFS...

And detect a cycle in the propcesss

DFS based algorithm:

1. Compute **DFS**(G)
2. If there is a back edge $e = (v, u)$ then G is not a DAG. Output cycle C formed by path from u to v in T plus edge (v, u).
3. Otherwise output nodes in decreasing post-visit order. Note: no need to sort, $DFS(G)$ can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in $O(n + m)$ time.

Correctness is not so obvious. See next two propositions.

Topological sort a graph using DFS...

And detect a cycle in the propcesss

DFS based algorithm:

1. Compute **DFS**(G)
2. If there is a back edge $e = (v, u)$ then G is not a DAG. Output cycle C formed by path from u to v in T plus edge (v, u).
3. Otherwise output nodes in decreasing post-visit order. Note: no need to sort, $DFS(G)$ can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in $O(n + m)$ time.
Correctness is not so obvious. See next two propositions.

Back edge and Cycles

Proposition

G has a cycle $\iff$ *there is a back-edge in* **DFS**(G).

Proof.

If: (u, v) is a back edge implies there is a cycle C consisting of the path from v to u in **DFS** search tree and the edge (u, v).

Only if: Suppose there is a cycle $C = v_1 \to v_2 \to \ldots \to v_k \to v_1$.
Let v_i be first node in C visited in **DFS**.
All other nodes in C are descendants of v_i since they are reachable from v_i.
Therefore, (v_{i-1}, v_i) (or (v_k, v_1) if $i = 1$) is a back edge. □

Decreasing post numbering is valid

Proposition

If G is a DAG *and* $\mathbf{post}(v) > \mathbf{post}(u)$, *then* $(u \to v)$ *is not in* G.

Proof.

Assume $\mathbf{post}(u) < \mathbf{post}(v)$ and $(u \to v)$ is an edge in G. One of two holds:

- Case 1: $[pre(u), post(u)]$ is contained in $[pre(v), post(v)]$.
- Case 2: $[pre(u), post(u)]$ is disjoint from $[pre(v), post(v)]$.

Decreasing post numbering is valid

Proposition

If G is a DAG *and* $\mathbf{post}(v) > \mathbf{post}(u)$*, then* $(u \to v)$ *is not in* G*.*

Proof.

Assume $\mathbf{post}(u) < \mathbf{post}(v)$ and $(u \to v)$ is an edge in G. One of two holds:

- Case 1: $[pre(u), post(u)]$ is contained in $[pre(v), post(v)]$.
- Case 2: $[pre(u), post(u)]$ is disjoint from $[pre(v), post(v)]$.

Decreasing post numbering is valid

Proposition

If G is a DAG *and* $\mathbf{post}(v) > \mathbf{post}(u)$*, then* $(u \rightarrow v)$ *is not in G.*

Proof.

Assume $\mathbf{post}(u) < \mathbf{post}(v)$ and $(u \rightarrow v)$ is an edge in G. One of two holds:

- Case 1: $[pre(u), post(u)]$ is contained in $[pre(v), post(v)]$. Implies that u is explored during $\mathbf{DFS}(v)$ and hence is a descendent of v. Edge (u, v) implies a cycle in G but G is assumed to be DAG!
- Case 2: $[pre(u), post(u)]$ is disjoint from $[pre(v), post(v)]$. This cannot happen since v would be explored from u.

Translation

We just proved:

Proposition

If G is a DAG and $\mathbf{post}(v) > \mathbf{post}(u)$, then $(u \to v)$ is not in G.

$\Longrightarrow$ sort the vertices of a DAG by decreasing post nubmering in decreasing order, then this numbering is valid.

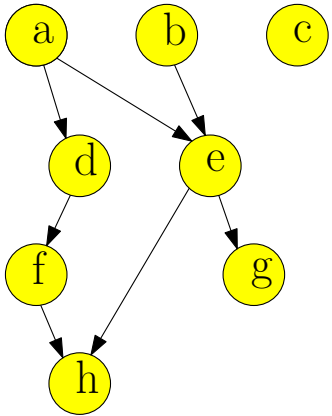
Topological sorting

Theorem

$G = (V, E)$: Graph with n vertices and m edges.
Comptue a topological sorting of G using **DFS** *in $O(n + m)$ time.*
That is, compute a numbering $\pi : V \to \{1, 2, \ldots, n\}$, such that

$$(u \to v) \in E(G) \implies \pi(u) < \pi(v).$$

Example



THE END

...

(for now)