Lecture 14: Topological Sort and Depth-first Search

CSC2100 Data Structure

Yufei Tao

CSE department, CUHK

April 8, 2011

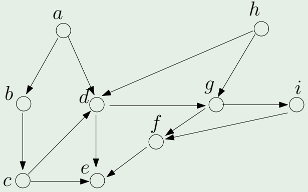
In this lecture, we will discuss a problem called *topological sort*. Our solution is based on an algorithm called *depth-first search*, which is another method for traversing a graph.

1. Problem
   - Dag
   - Topological sort
2. DFS
   - Rationale
   - Formal description and analysis
3. Topological sort
   - Algorithm
   - Correctness

Directed acyclic graph

A directed graph G is a *dag* (*directed acyclic graph*) if it does not have any cycle.

Example



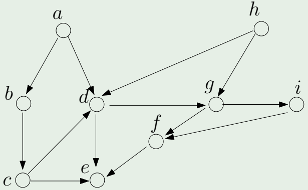
The left graph is a dag, but the right is not.

Topological sort problem

Problem (Topological sort)

Given a dag $G = (V, E)$, find an ordering of V, such that, for each edge $(u, v) \in E$, u precedes v in the ordering. The ordering is called a *topological order*.

Example



- A topological order: $a, b, c, h, d, g, i, f, e$.
- Another: $h, a, b, c, d, g, i, f, e$.

The result is not unique.

Depth-first search

- Before dealing with the topological sort problem, let us first clarify how to perform a *depth-first search* (DFS) on a graph.
- At each vertex v, DFS "eagerly" switches to a neighbor of v (unlike BFS that switches only after having inspected all the neighbors of v), as we will see.

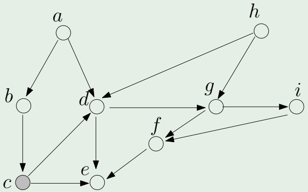
Note

In the sequel, we will say that vertex v is a *neighbor* of vertex u, if $(u, v) \in E$, namely, there is an edge from u to v.

DFS example

Example

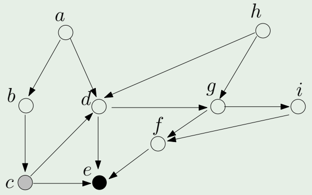
At the beginning, color all vertices white (which means "not visited yet"). Consider a DFS starting from vertex c.



- Color c as grey (which means "visited, but neighbors not finished yet").
- Switch to a white neighbor of c, say e.

DFS example (cont.)

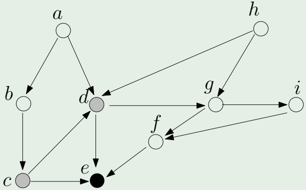
Example



- Color e grey.
- Attempt to switch to a white neighbor of e. As such a neighbor does not exist, color e black (which means "visited, and all neighbors done").
- The algorithm then backtracks to c.
- At c, switch to the next white neighbor of c, i.e., d.

DFS example (cont.)

Example

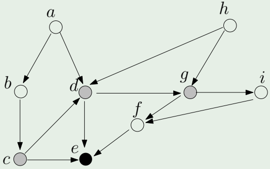


- Color d grey.
- Switch to a white neighbor of d, i.e., g.

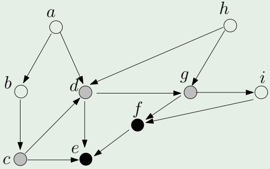
DFS example (cont.)

Example

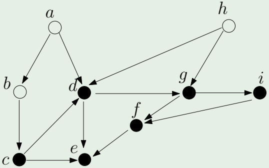
Access g:



Access f:



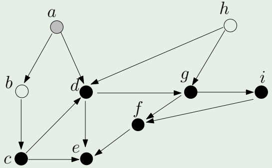
Backtrack all the way:

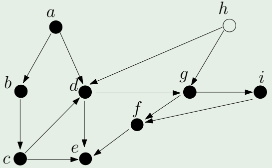


DFS enters a dead end here. To break the dead end, simply re-start DFS from another white vertex, say a.

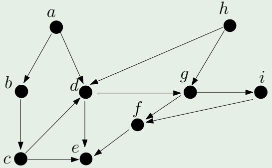
DFS example (cont.)

Example



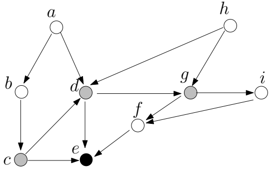


Another dead end. Re-start from a white vertex. There is only one left: h.



Backtracking

We must be able to backtrack efficiently. For instance, in the following situation, we need to return to d after finishing with g, and likewise, return to c after finishing with d.



This can be easily achieved by managing all the grey vertices using a stack (see the tutorial of Week 6 and its first-in-last-out property).

Pseudocode

Algorithm *DFS*

color all vertices white
initialize an empty stack S
while there is still a white vertex u
    $color[u] = \text{grey}$
    $v_{active} = u$
    do
        if v_{active} has a white neighbor v
            $color[v] = \text{grey}$
            insert v_{active} into S
            $v_{active} = v$
        else
            $color[v_{active}] = \text{black}$
            pop the top vertex of S, and set it to v_{active}
    while $v_{active} \neq \emptyset$

Running time

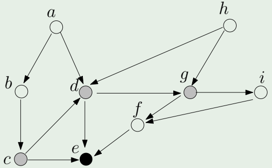
- Each edge is explored exactly once.
- Each vertex enters and leaves the stack exactly once.

Total time $= O(|V| + |E|)$.

Algorithm for topological sort

Simply perform a DFS, and output the vertices in the reverse order of turning black.

Example



- Order of turning black: $e, f, i, g, d, c, b, a, h$.
- A topological order: $h, a, b, c, d, g, i, f, e$.

Running time $= O(|V| + |E|)$.

Correctness proof

Prove: The algorithm in the previous slide correctly finds a topological order.

Proof.

Take any edge (u, v). We will show that u turns black after v. Consider the moment when DFS explores (u, v); namely, at this point, u is grey, and the algorithm is checking whether it should switch to v.

- If v is black, then obviously u turns black after v.
- If v is grey, then there is a cycle, i.e., from v via another path to u, plus edge (u, v). This contradicts the fact that the graph is a dag.
- If v is white, then it will be inserted into the stack after u, and popped out (and hence, turn black) before u.

$\square$

Playback of this lecture:

- Depth-first search takes $O(|V| + |E|)$ time.
- Same for topological sort.