# Topological Sort on a DAG

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

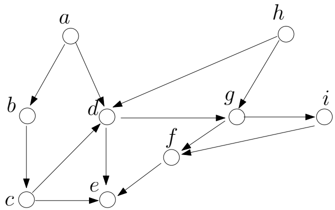
In this lecture, we will deploy depth first search (DFS) algorithm to settle another classic problem — called **topological sort** — in linear time.

## Topological Order

Let $G = (V, E)$ be a directed acyclic graph (DAG).

A **topological order** of $G$ is an ordering of the vertices in $V$ such that, for every edge $(u, v)$ in $E$, it must hold that $u$ precedes $v$ in the ordering.

## Example




The following are two possible topological orders:

- $h, a, b, c, d, g, i, f, e$.
- $a, h, b, c, d, g, i, f, e$.

An ordering that is not a topological order:

- $a, h, d, b, c, g, i, f, e$ (because of edge $(c, d)$).

Remarks:

- A directed cyclic graph has no topological orders (think: why?).
- Every DAG has a topological order.
    - This will be a corollary from our subsequent discussion.

**The Topological Sort Problem**

Let $G = (V, E)$ be a directed acyclic graph (DAG). The goal of **topological sort** is to produce a topological order of $G$.

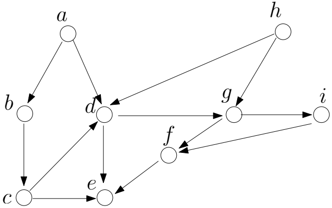
## Algorithm

Very simple:

1. Create an empty list $L$.
2. Run DFS on $G$. Whenever a vertex $v$ turns black (i.e., it is popped from the stack), append it to $L$.
3. Output the reverse order of $L$.

The total running time is clearly $O(|V| + |E|)$.

## Example 1

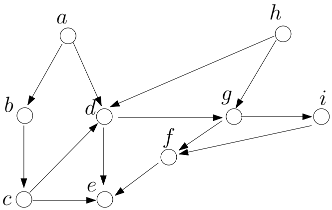



Suppose that we run DFS starting from $a$. The following is one possible order by which the vertices turn black:

- $e, f, i, g, d, c, b, a, h$.

Therefore, we output $h, a, b, c, d, g, i, f, e$ as a topological order.

## Example 2




Suppose that we run DFS starting from $d$, then restarting from $h$, and then from $a$. The following is one possible order by which the vertices turn black:

- $e, f, i, g, d, h, c, b, a$.

Therefore, we output $a, b, c, h, d, g, i, f, e$ as a topological order.

We will now prove that the algorithm is correct.

**Proof:** Take any edge $(u, v)$. We will show that $u$ turns black after $v$, which will complete the proof.

Consider the moment when $u$ enters the stack. We argue that currently $v$ cannot be in the stack. Suppose that $v$ was in the stack. As there must be a path chaining up all the vertices in the stack bottom up, we know that there is a path from $v$ to $u$. Then, adding the edge $(u, v)$ forms a cycle, contradicting the fact that $G$ is a DAG.

Now it remains to consider:

- $v$ is black at this moment: Then, trivially, $u$ will turn black after $v$.
- $v$ is white: Then by the white path theorem of DFS, we know that $v$ will become a proper descendant of $u$ in the DFS-forest. Therefore, $u$ will turn back after $v$.

□

The correctness of our algorithm also proves:

Every DAG has a topological order.