

Decomposition of Graphs: Topological Sort

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

Graph Algorithms
Data Structures and Algorithms

Learning Objectives

- Implement the topological sort algorithm.
- Prove that a DAG can be linearly ordered.

Outline

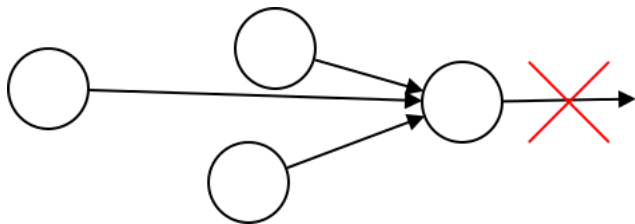
- 1 Idea
- 2 Algorithms
- 3 Correctness

Last Time

- Directed graphs.
- Linearly order vertices.
- Requires DAG.

Last Vertex

Consider the last vertex in the ordering. It cannot have any edges pointing out of it.



Sources and Sinks

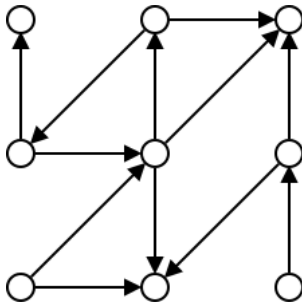
Definition

A **source** is a vertex with no incoming edges.

A **sink** is a vertex with no outgoing edges.

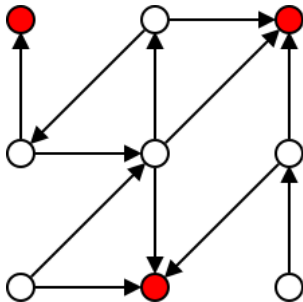
Problem

How many sinks does the graph below have?



Solution

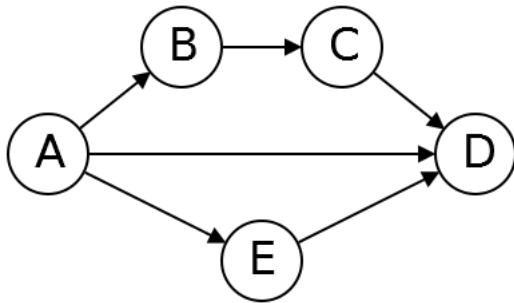
3.



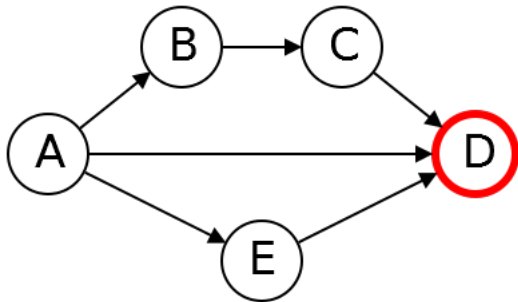
Idea

- Find sink.
- Put at end of order.
- Remove from graph.
- Repeat.

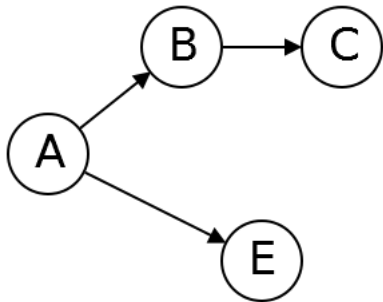
Example



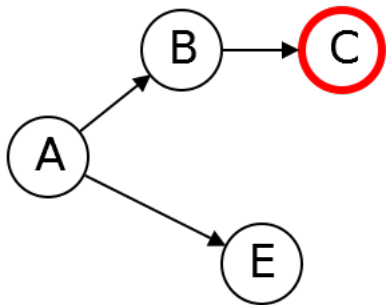
Example



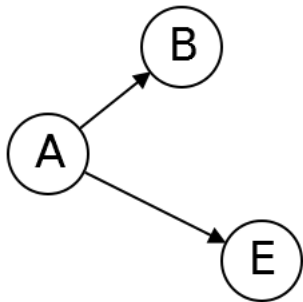
Example



Example



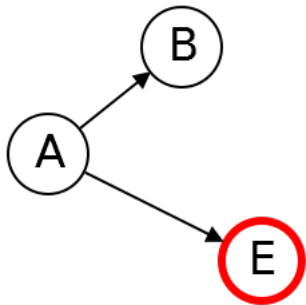
Example



C

D

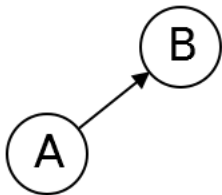
Example



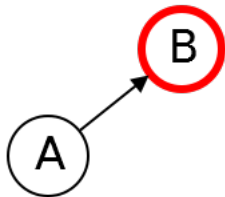
C

D

Example



Example



Example

A

B

E

C

D

Example

A

B

E

C

D

Example



Finding Sink

Question: How do we know that there is a sink?

Follow Path

Follow path as far as possible

$v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$. Eventually either:

- Cannot extend (found sink).
- Repeat a vertex (have a cycle).

Outline

- 1 Idea
- 2 Algorithms
- 3 Correctness

First Try

LinearOrder(G)

while G non-empty:

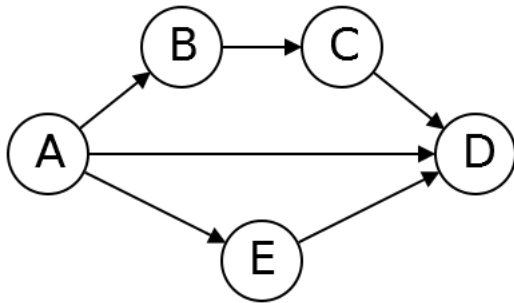
 Follow a path until cannot extend

 Find sink v

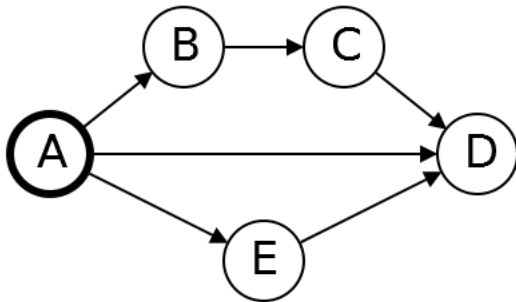
 Put v at end of order

 Remove v from G

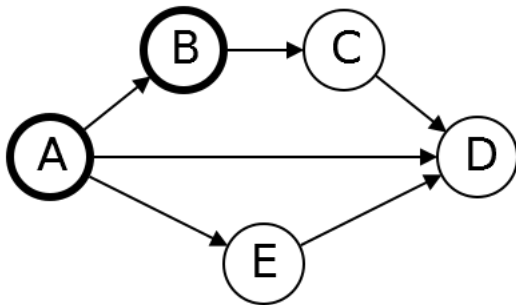
Example



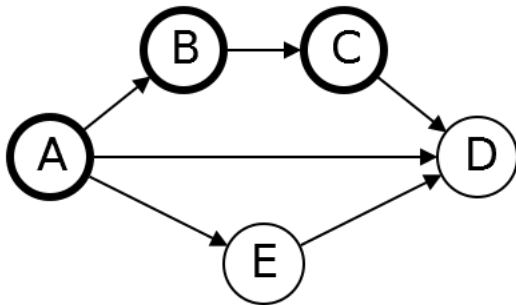
Example



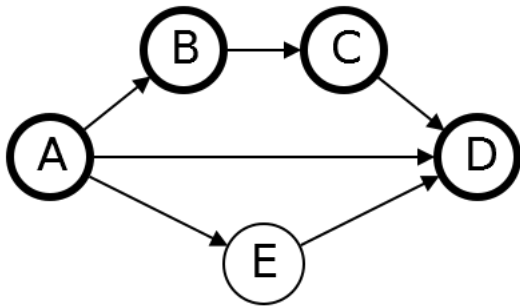
Example



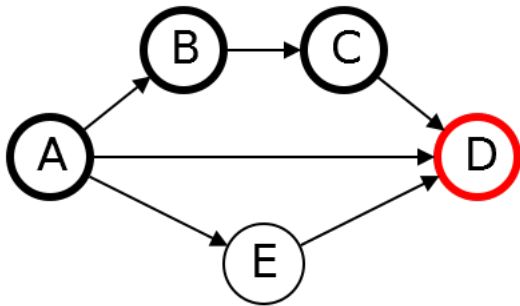
Example



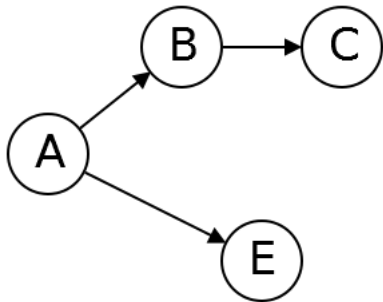
Example



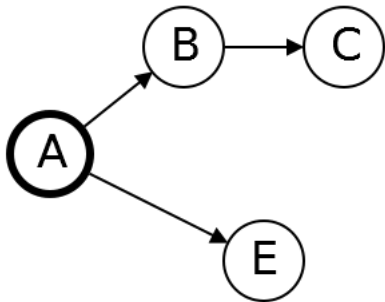
Example



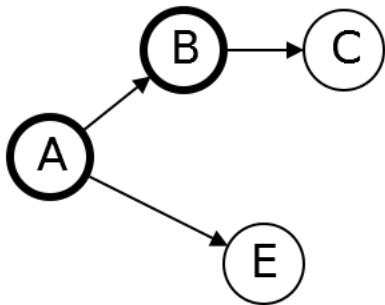
Example



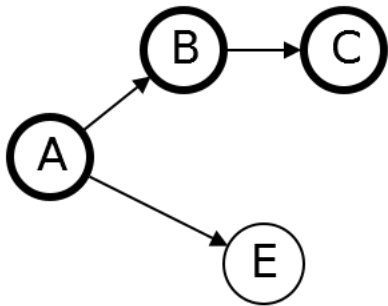
Example



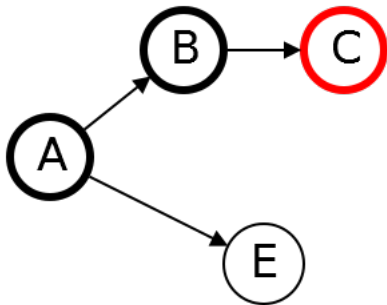
Example



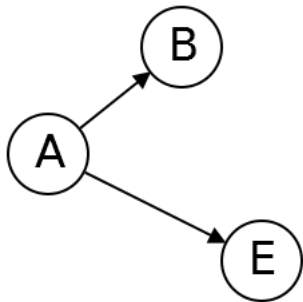
Example



Example



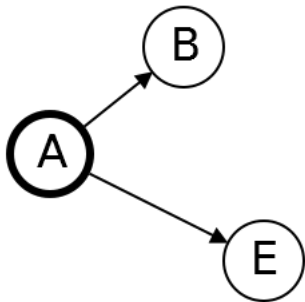
Example



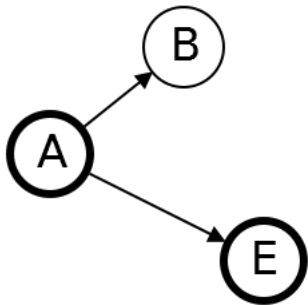
C

D

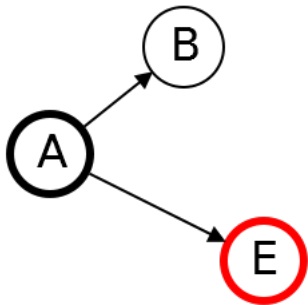
Example



Example



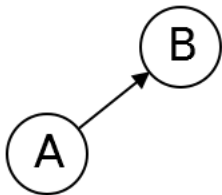
Example



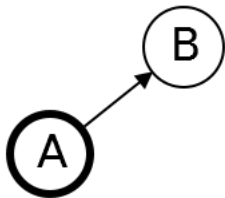
C

D

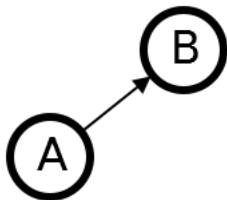
Example



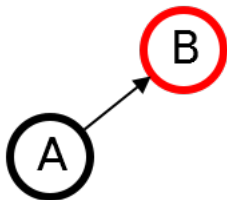
Example



Example



Example



Example

A

B

E

C

D

Example

A

B

E

C

D

Example

A

B

E

C

D

Example



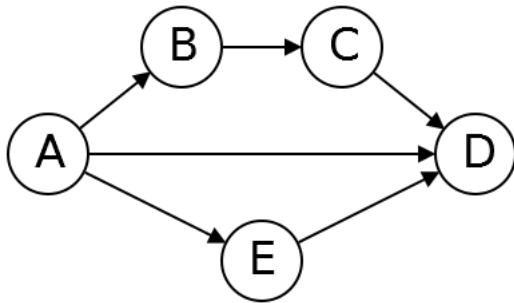
Runtime

- $O(|V|)$ paths.
- Each takes $O(|V|)$ time.
- Runtime $O(|V|^2)$.

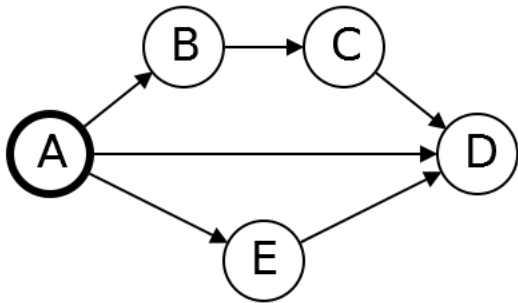
Speed Up

- Retrace same path every time.
- Instead only back up as far as necessary.

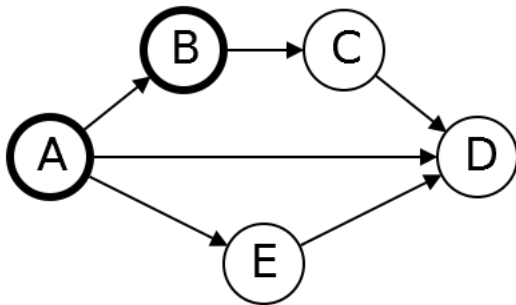
Example



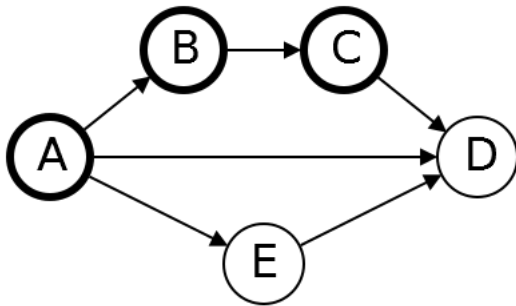
Example



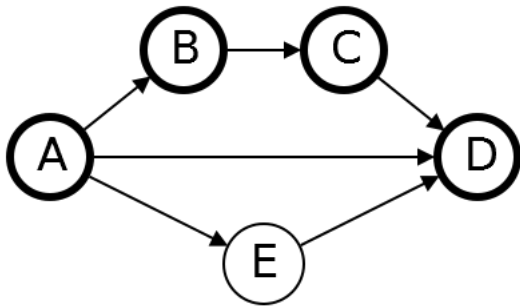
Example



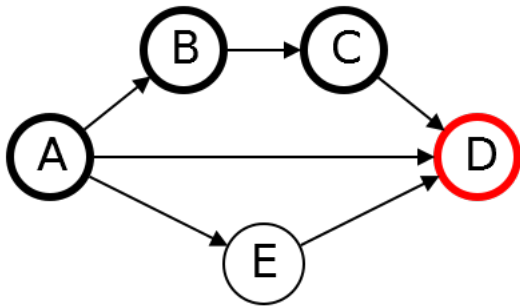
Example



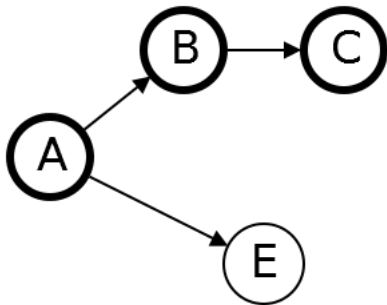
Example



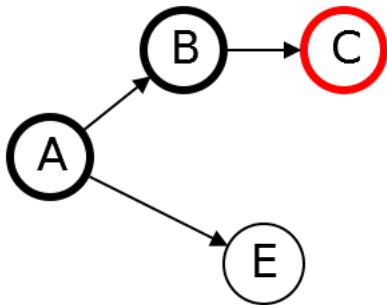
Example



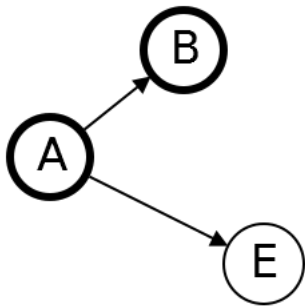
Example



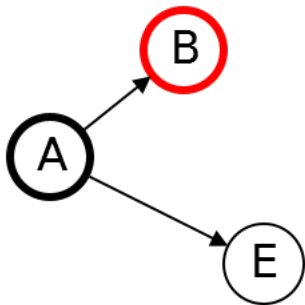
Example



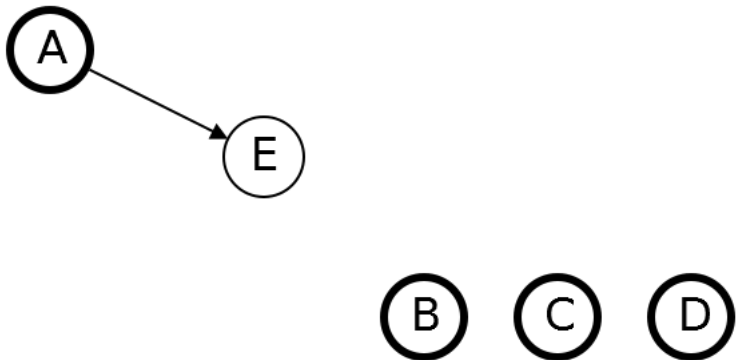
Example



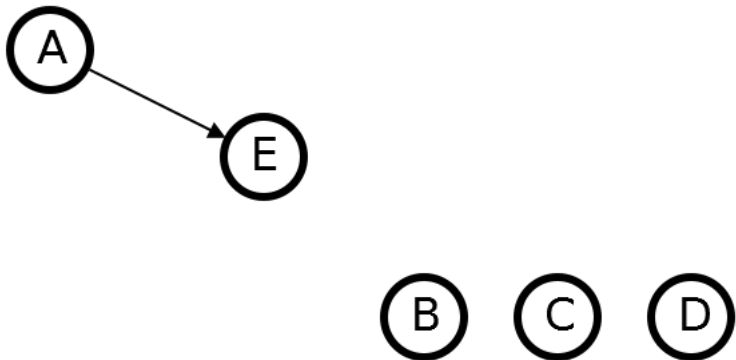
Example



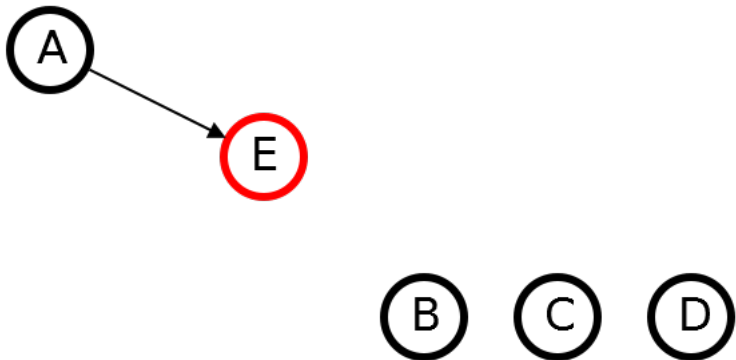
Example



Example



Example



Example

A

E

B

C

D

Example

A

E

B

C

D

Example



Observation

This is just DFS!

Observation

This is just DFS!

We are sorting vertices based in postorder!

Better Algorithm

`TopologicalSort(G)`

`DFS(G)`

sort vertices by reverse post-order

Outline

- 1 Idea
- 2 Algorithms
- 3 Correctness

Theorem

Theorem

If G is a DAG, with an edge u to v ,
 $\text{post}(u) > \text{post}(v)$.

Proof

Consider cases

- Explore v before exploring u .
- Explore v while exploring u .
- Explore v after exploring u (cannot happen since there is an edge).

Case I

Explore v before exploring u .

- Cannot reach u from v (DAG)
- Must finish exploring v before find u
- $\text{post}(u) > \text{post}(v)$.

Case II

Explore v while exploring u .

Must finish exploring v before can finish exploring u . Therefore $\text{post}(u) > \text{post}(v)$.

Next Time

Connectivity in directed graphs.