Design and Analysis of Algorithms

05-04
Directed Acyclic Graph

Topological Sorting

Imran Ihsan
Assistant Professor, Department of Computer Science
Air University, Islamabad, Pakistan
www.imranihsan.com

Directed Graphs

Sometimes we want the edges of a graph to have a direction.

Definition

A directed graph is a graph where each edge has a start vertex and an end vertex.

Examples

Directed graphs might be used to represent:

- Streets with one-way roads.
- Links between webpages.
- Followers on social network.
- Dependencies between tasks.

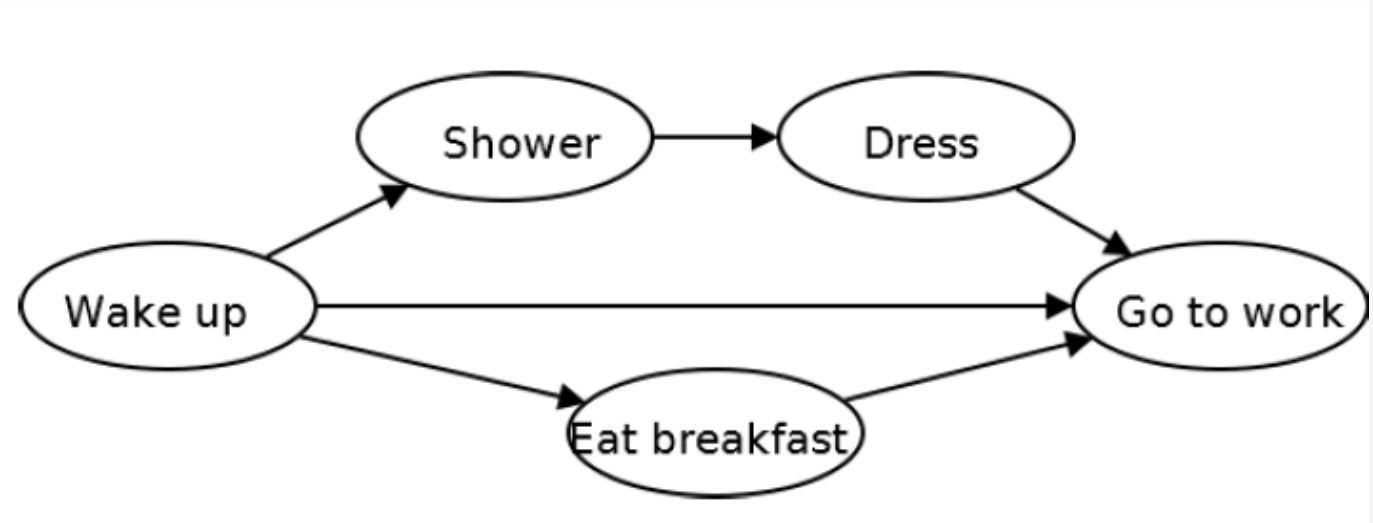
Directed DFS

Can still run DFS in directed graphs.

- Only follow directed edges.
- explore(v) finds all vertices reachable from v.
- Can still compute pre- and post-orderings.

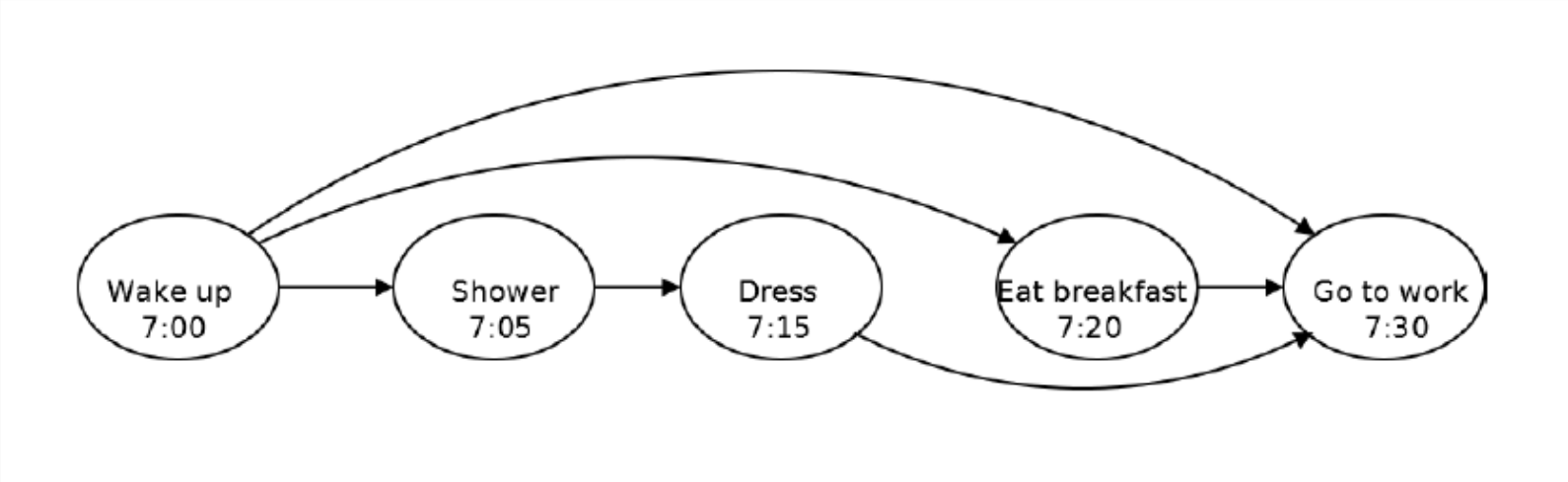
Morning Routine

The following morning tasks must be performed some before others.



Linear Ordering

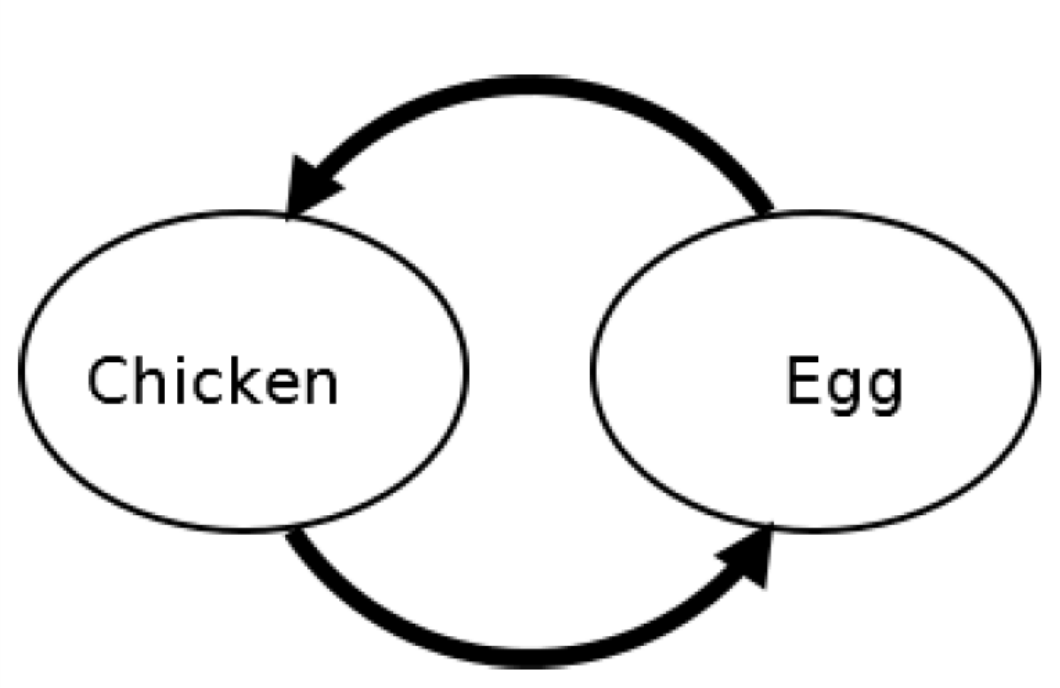
We would like to order tasks to respect dependencies as below.



Is it

Example

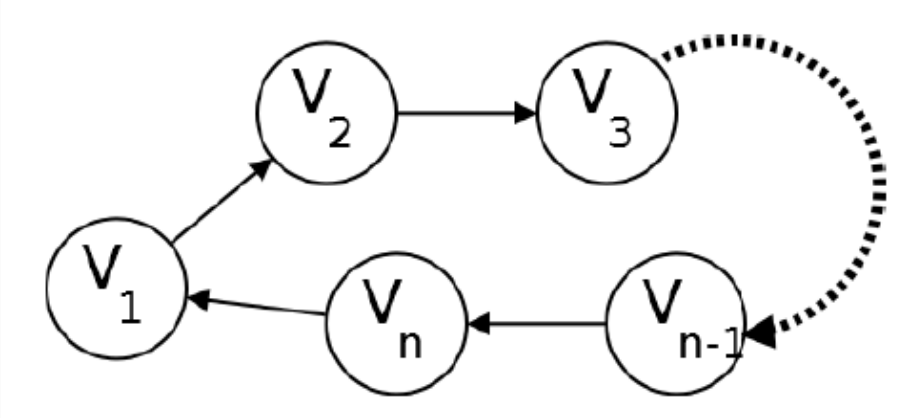
No!



Cycles

Definition

A cycle in a graph G is a sequence of vertices v_1, v_2, …… , v_n so that (v_1, v_2), (v_2, v_3),…, (v_{n-1}, v_n), (v_n, v_1) are all edges.



Cycles

Theorem

If G contains a cycle,

it cannot be linearly ordered.

Proof

Has cycle v_1, v_2, …… , v_n.

Suppose linearly ordered.

Suppose v_k comes first.

Then v_k comes before v_{k-1}, contradiction..

DAGs

Definition

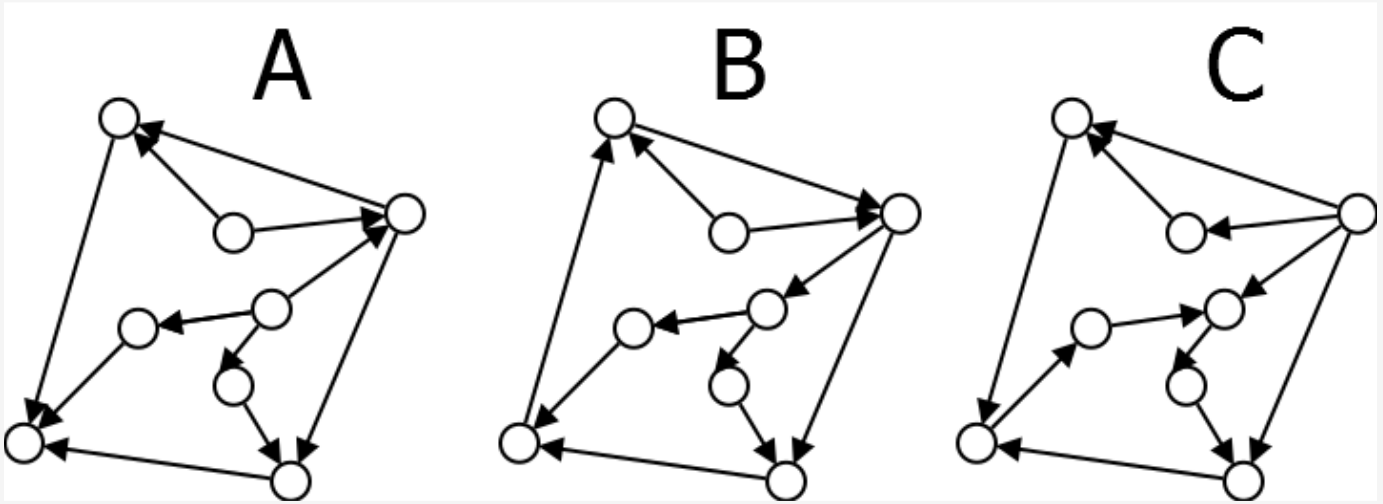
A directed graph G is a Directed Acyclic Graph (or DAG) if it has no cycles.

By the above being a DAG is necessary to linearly order.

Is it sufficient?

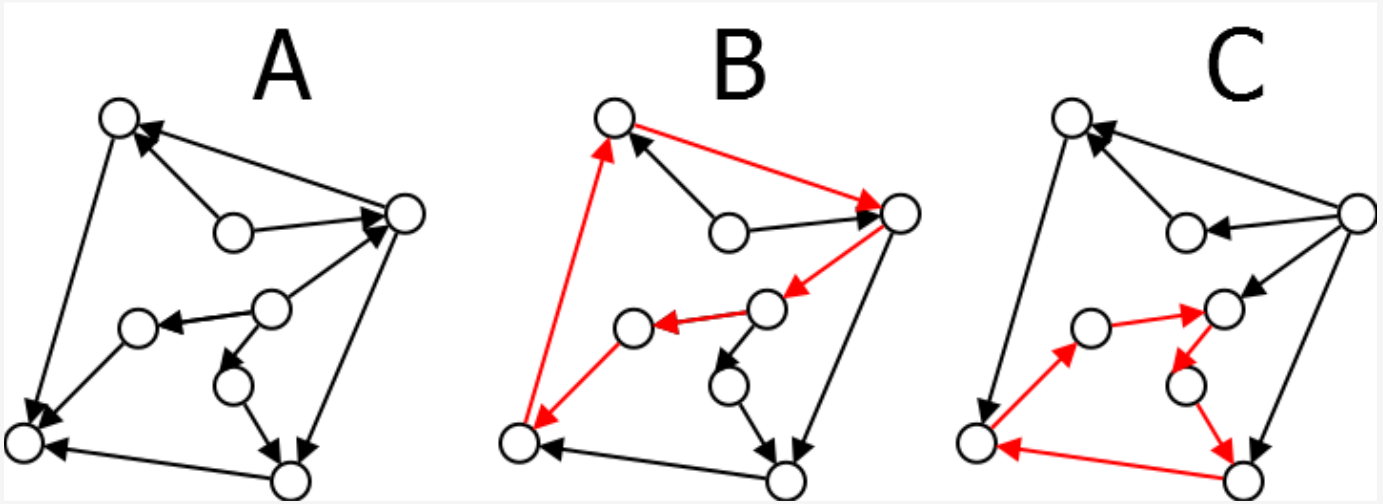
Problem

Which of the following graphs is a DAG?



Solution

A



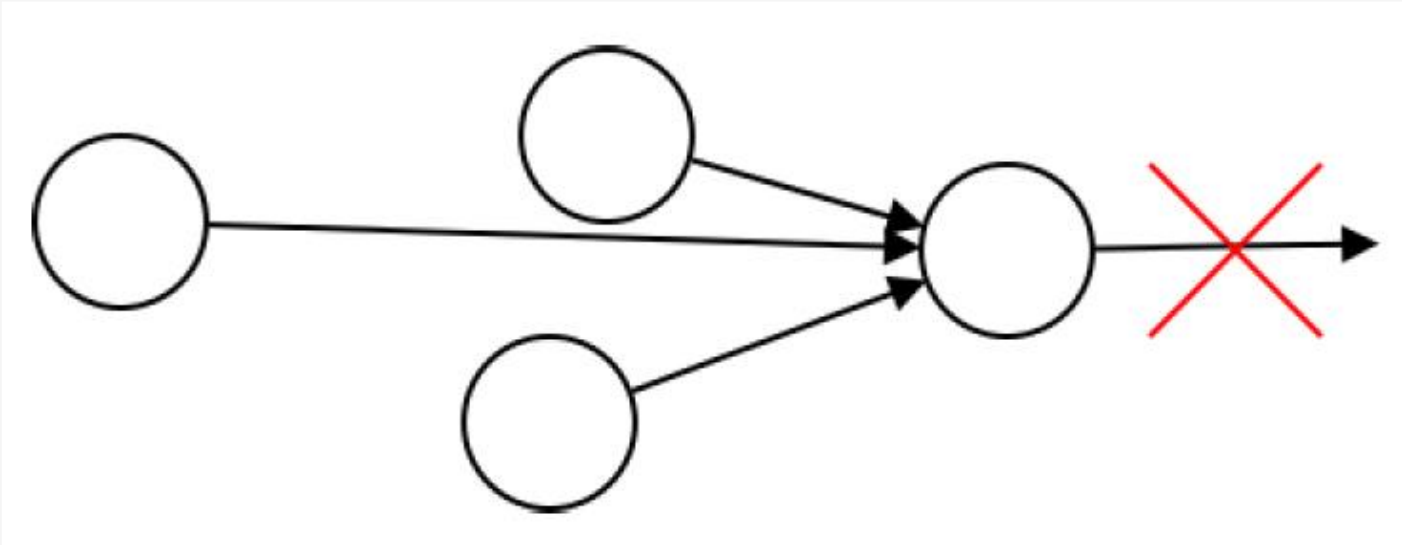
Topological sorting

Can a dag be linearly ordered?

Last Vertex

Consider the last vertex in the ordering.

It cannot have any edges pointing out of it.



Sources and sinks

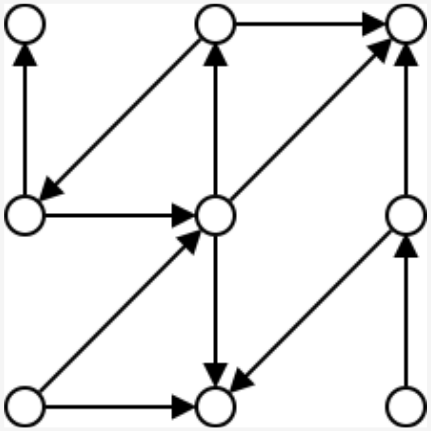
Definition

A source is a vertex with no incoming edges.

A sink is a vertex with no outgoing edges.

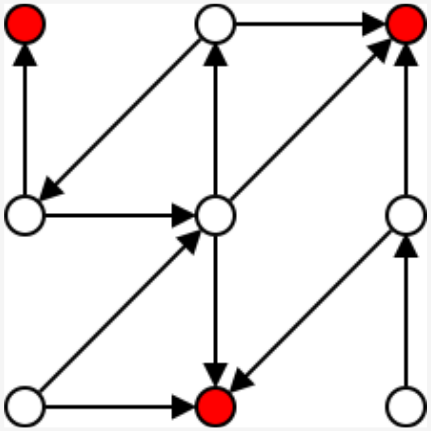
Problem

How many sinks does the graph below have?



Problem

3



Idea

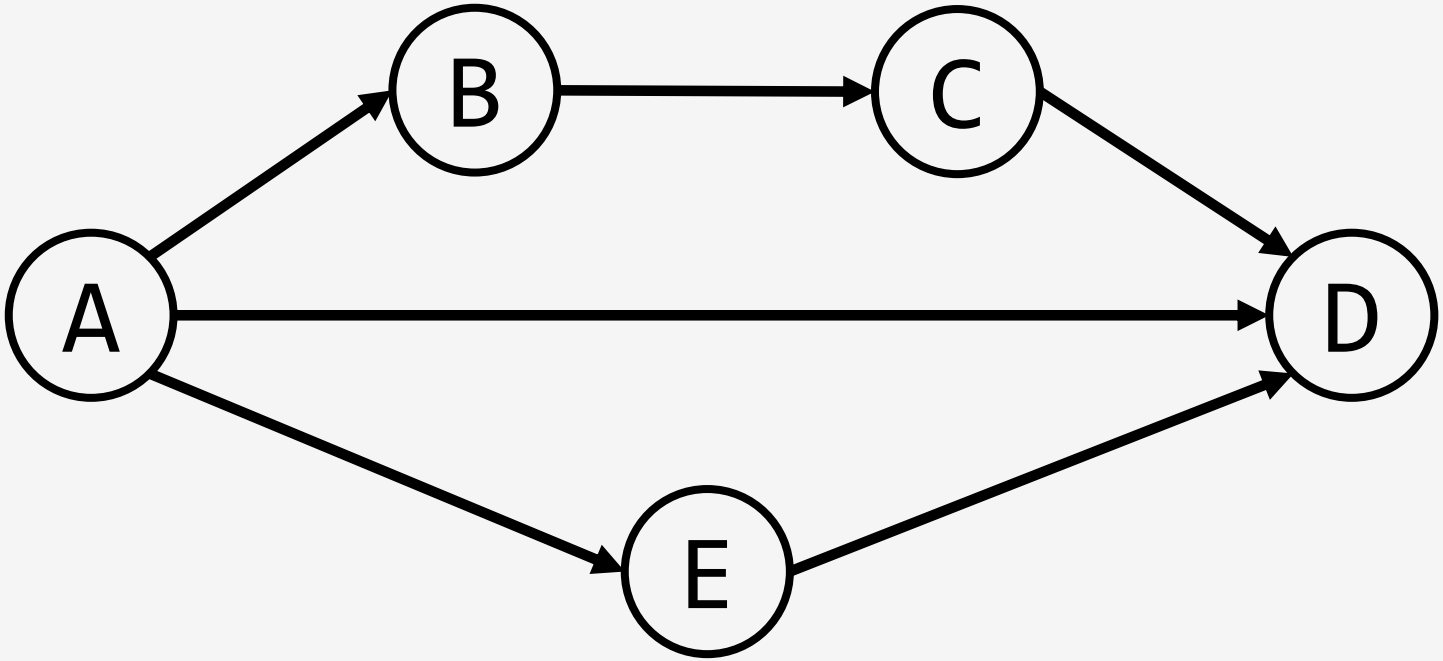
Find sink.

Put at end of order.

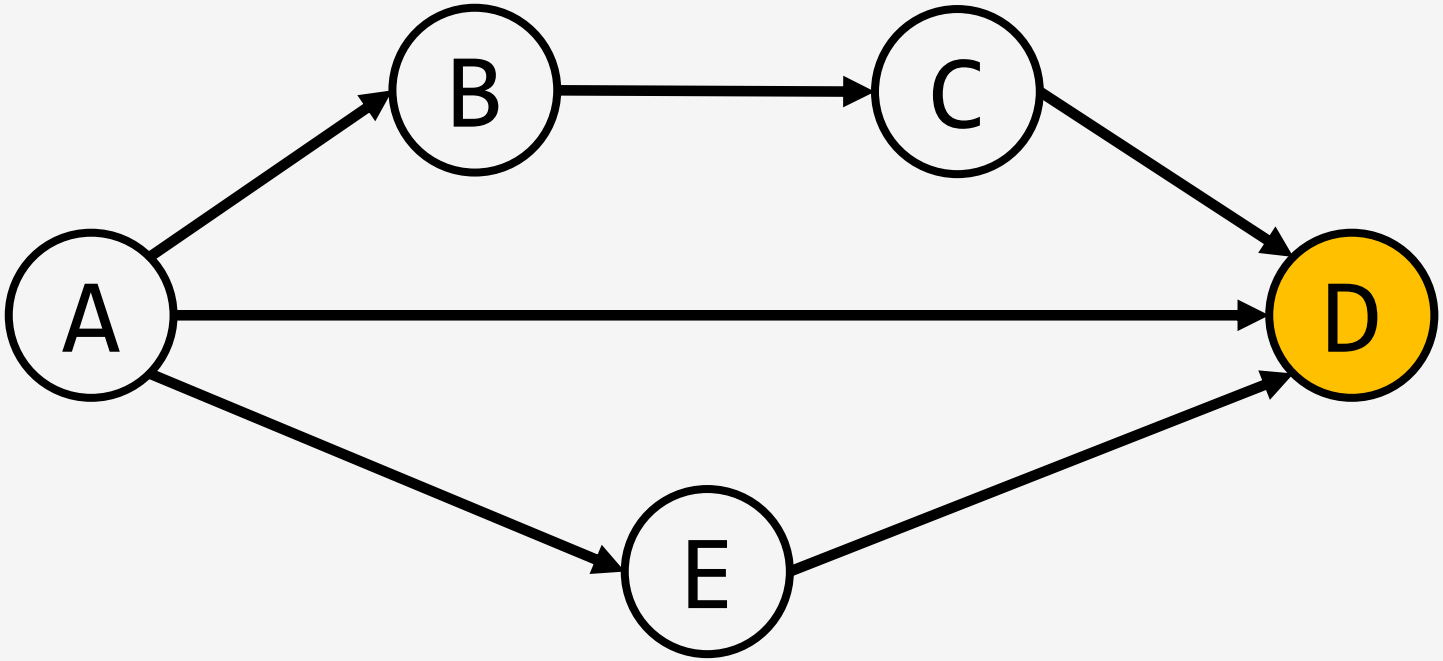
Remove from graph.

Repeat.

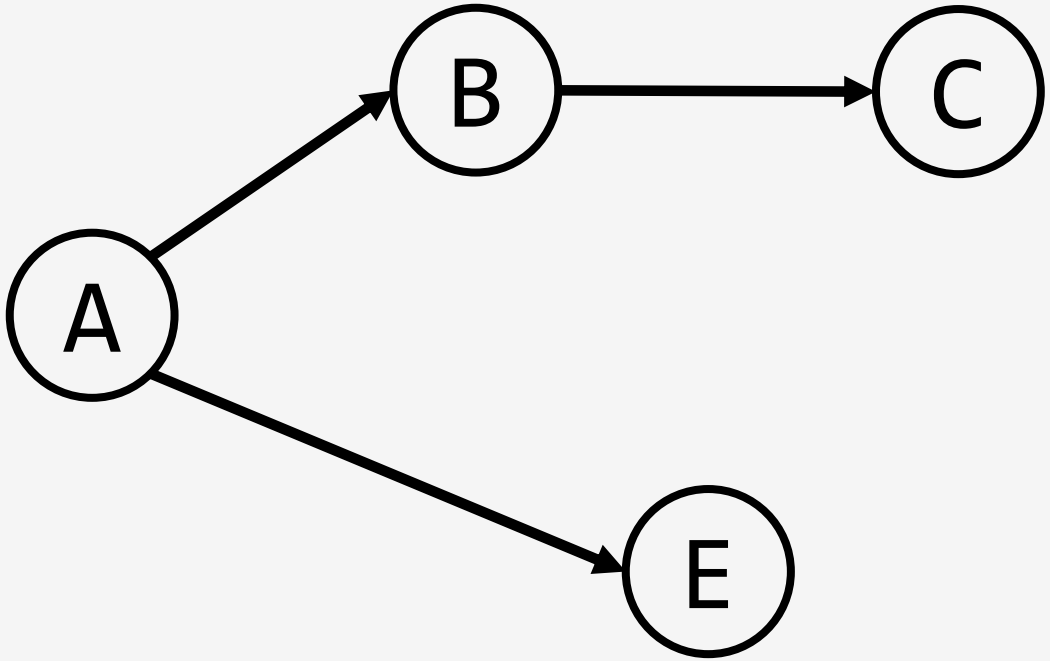
Example



Example

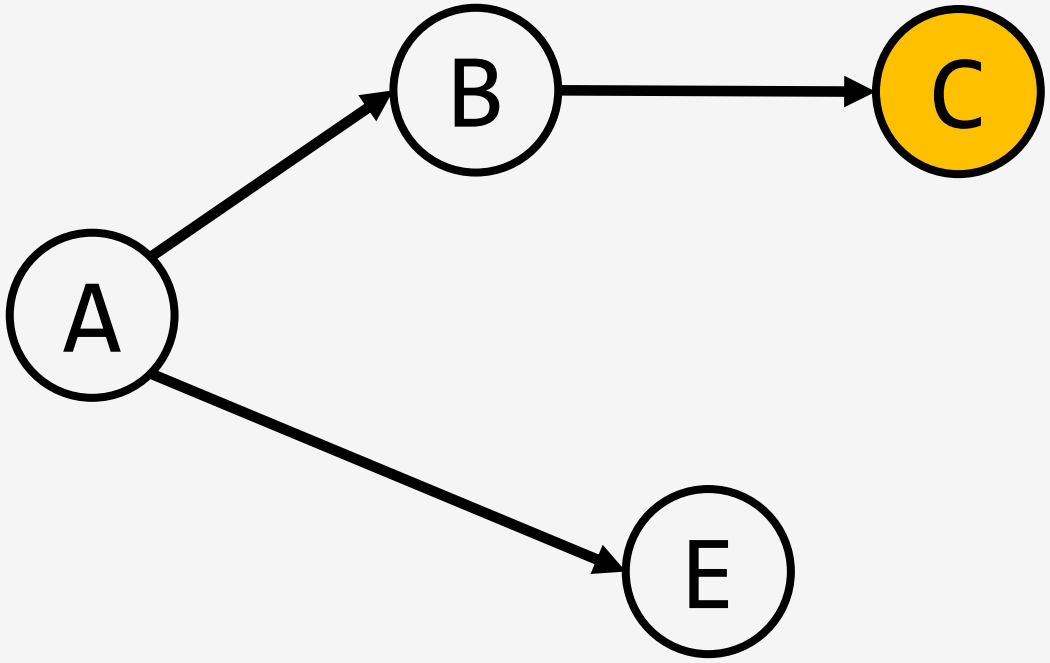


Example



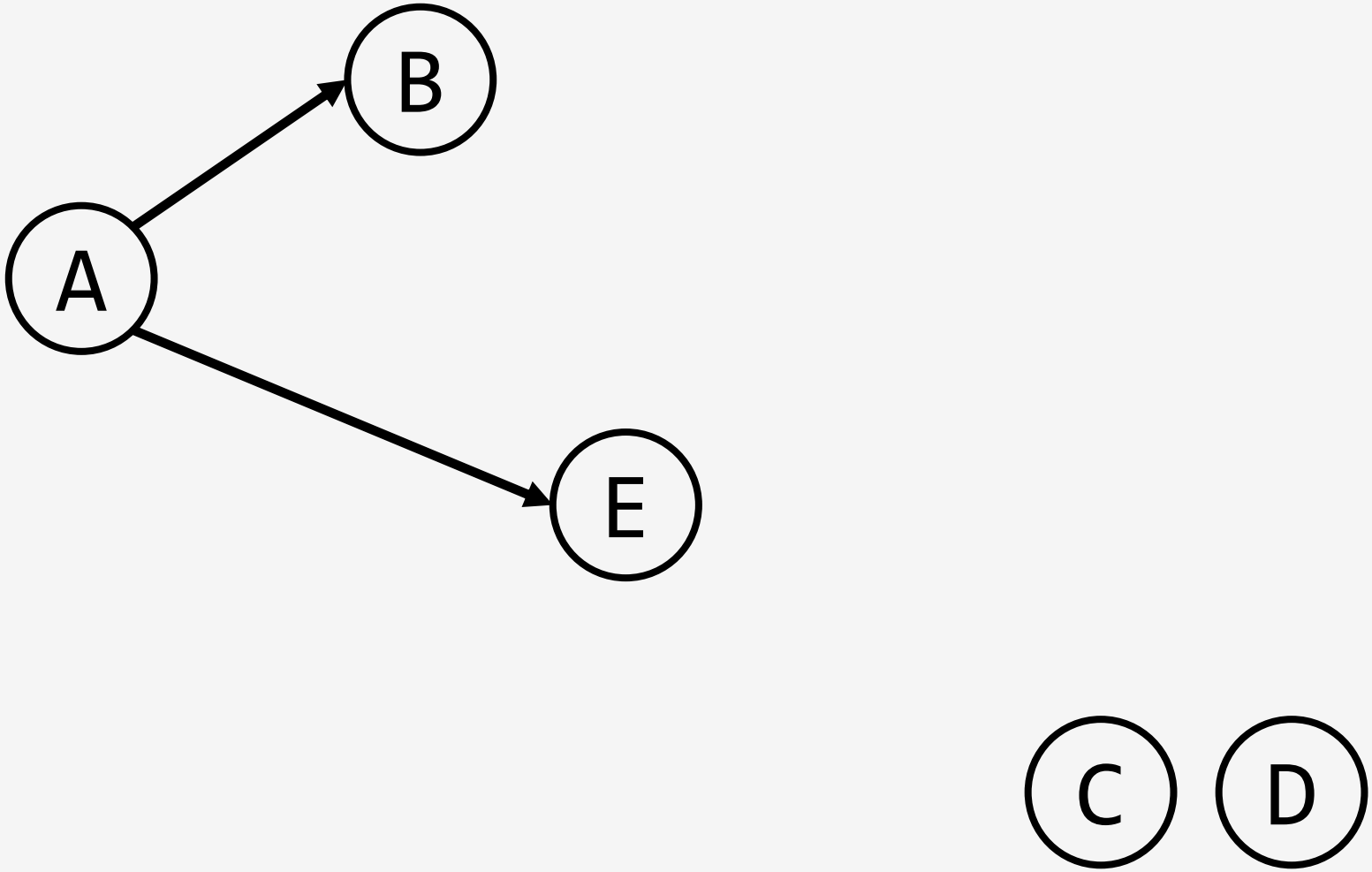
D

Example

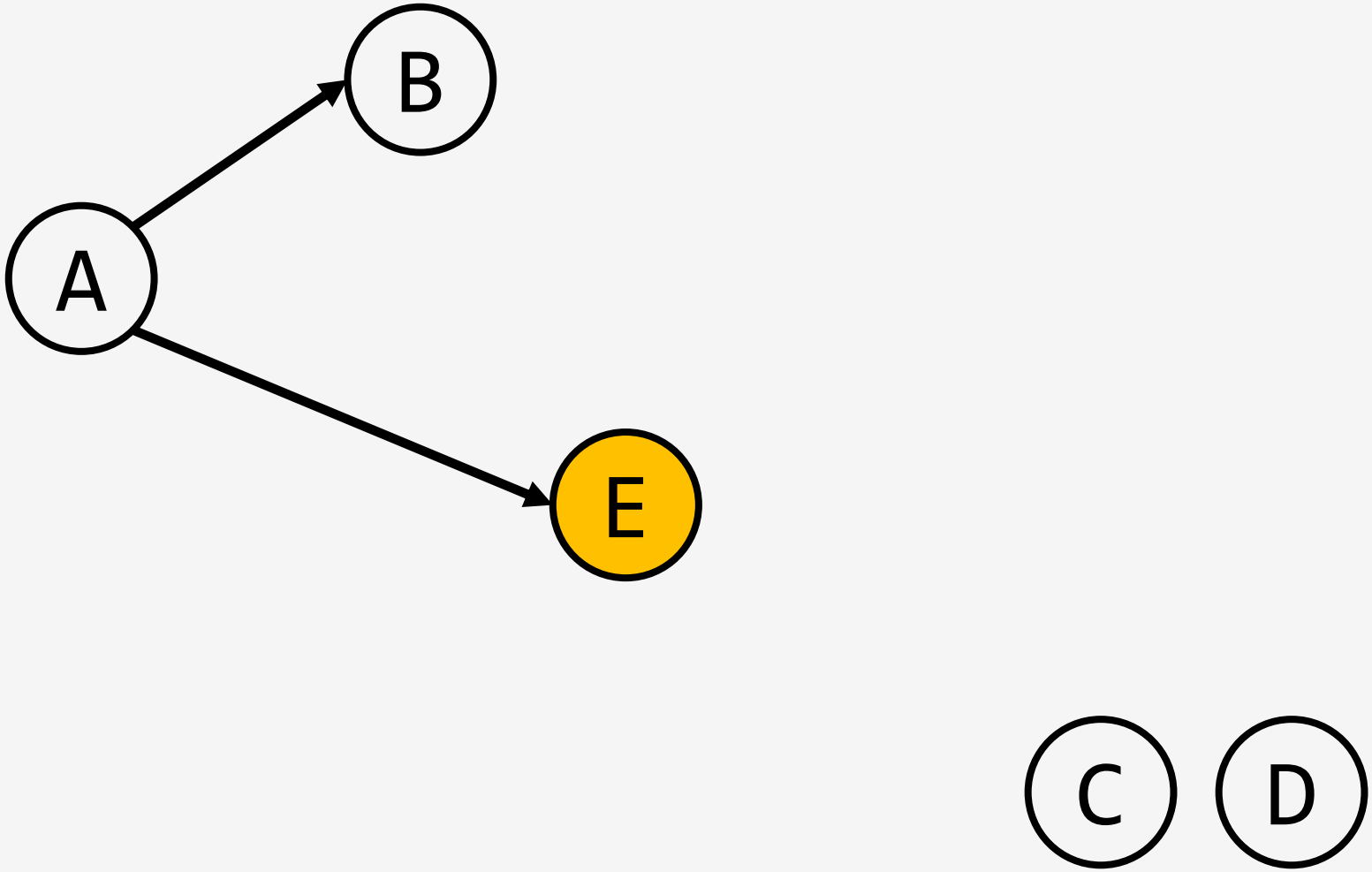


D

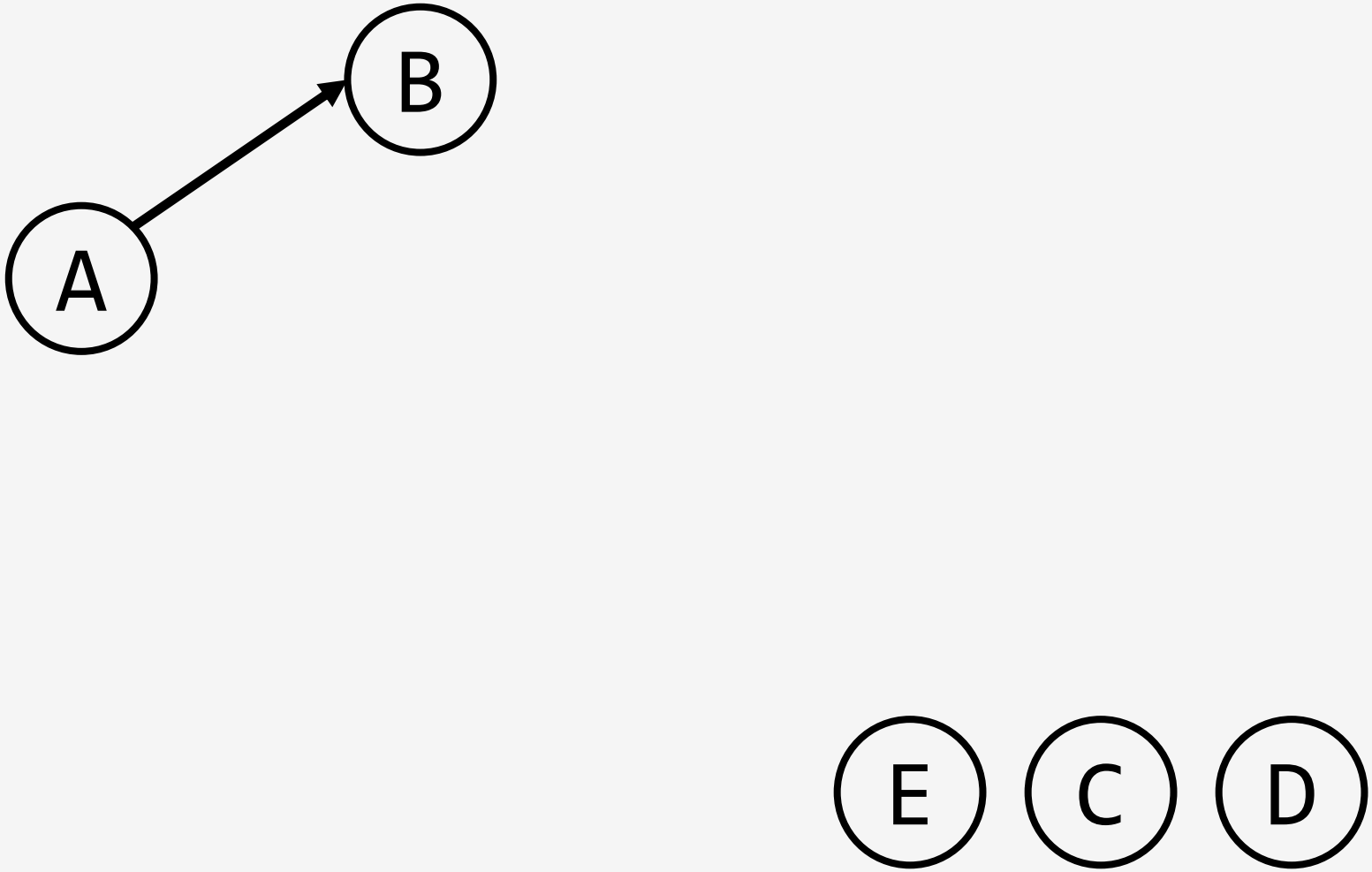
Example



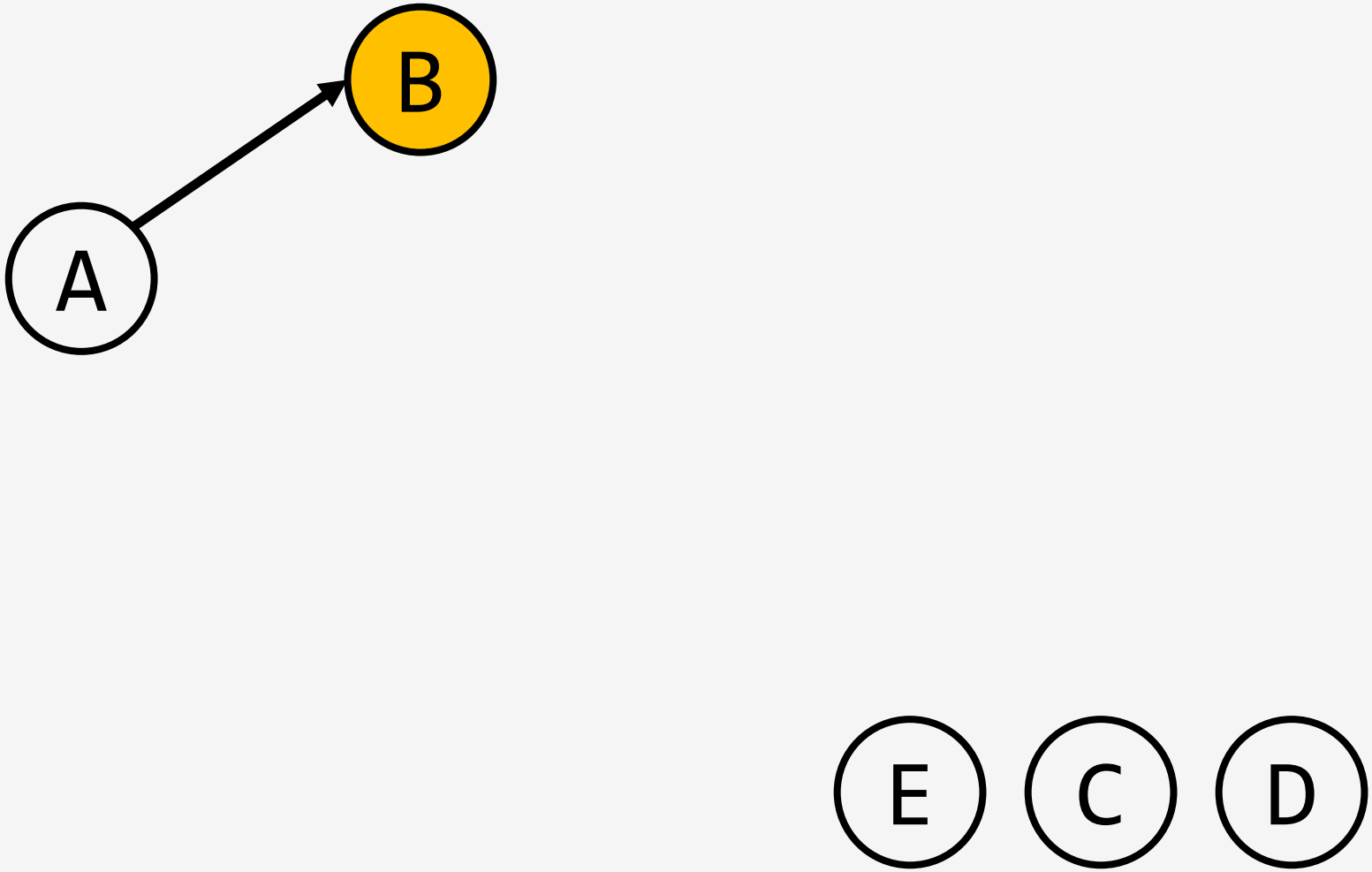
Example



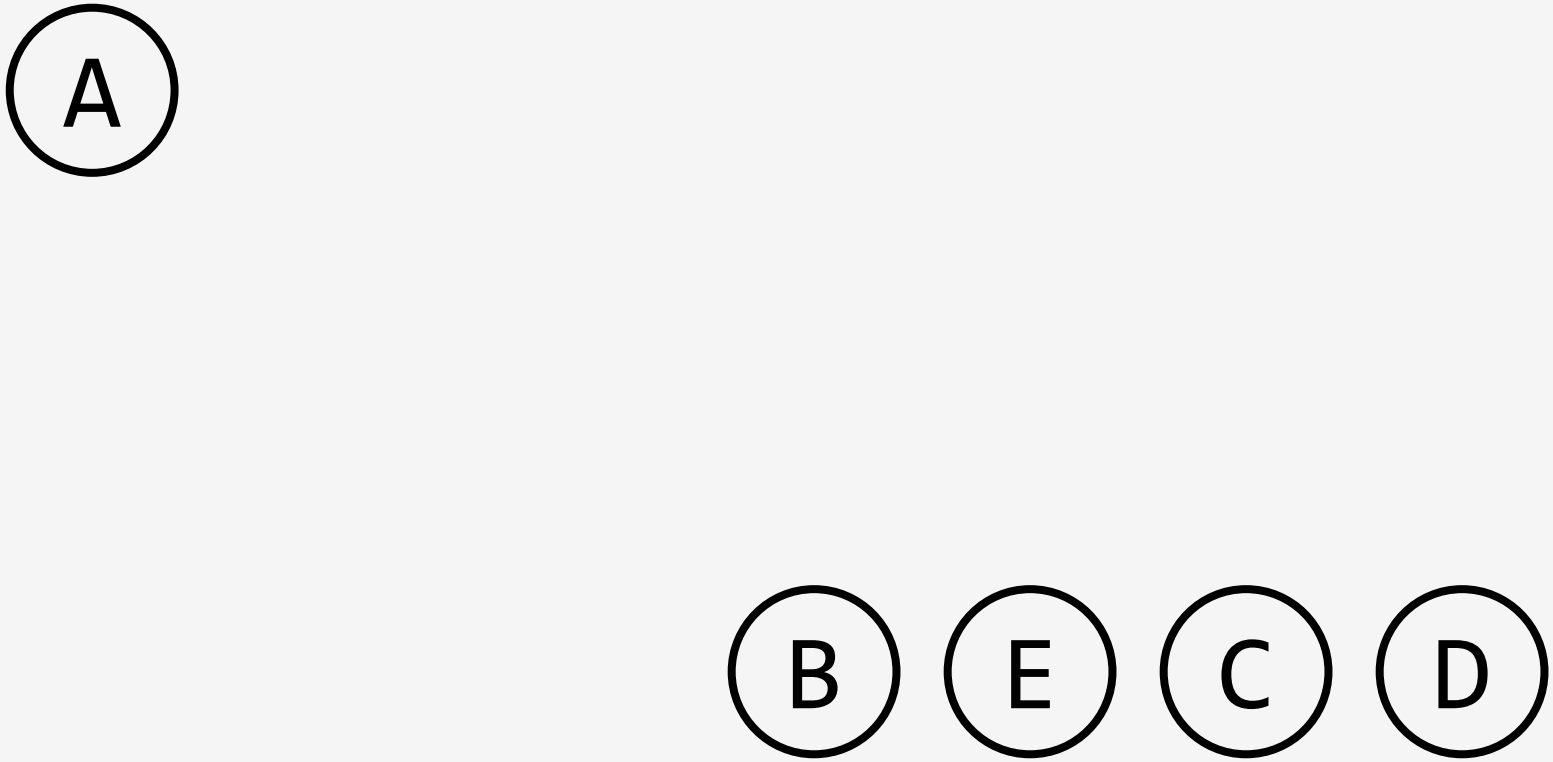
Example



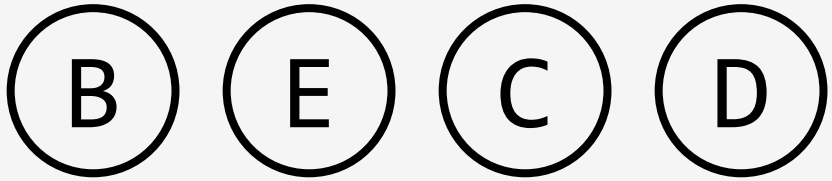
Example



Example



Example



Example

Finding Sink

Question:

How do we know that there is a sink?

Answer: Follow Path

Follow path as far as possible $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n$.

Eventually either:

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

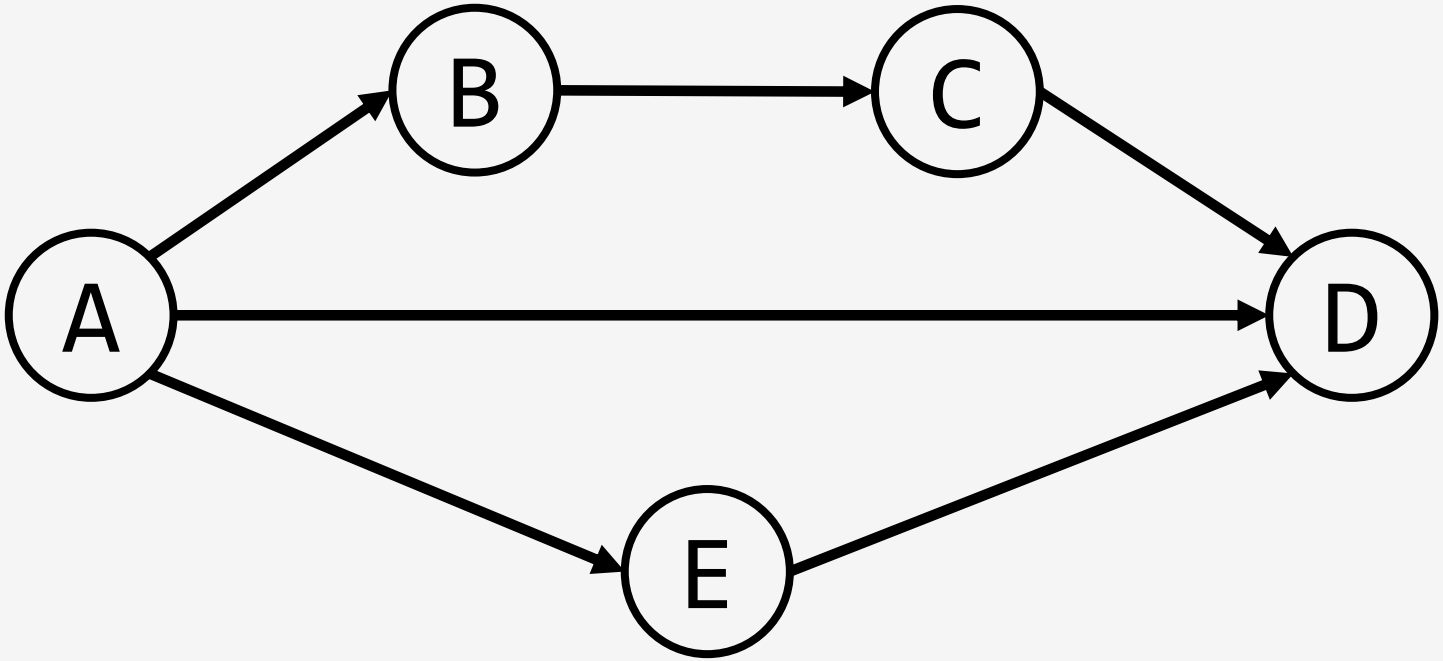
Topological sorting

Algorithm

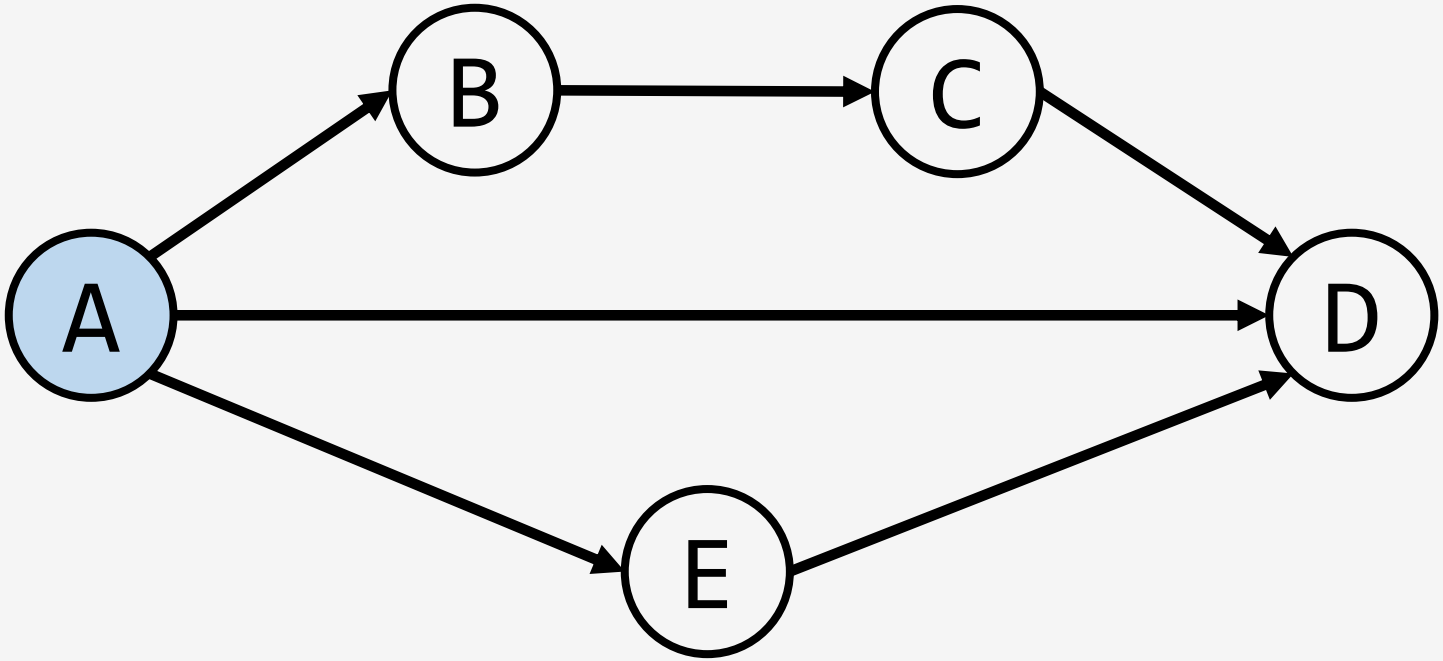
First Try

```
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
```

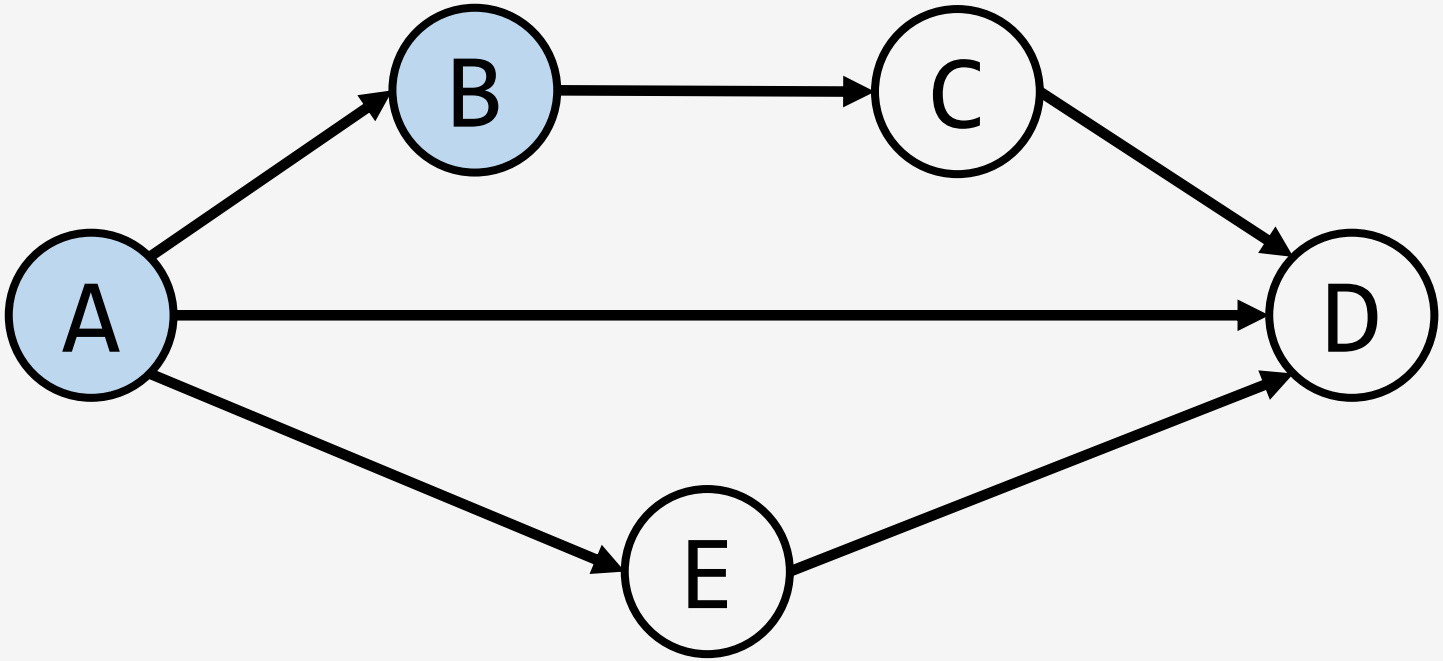
Example



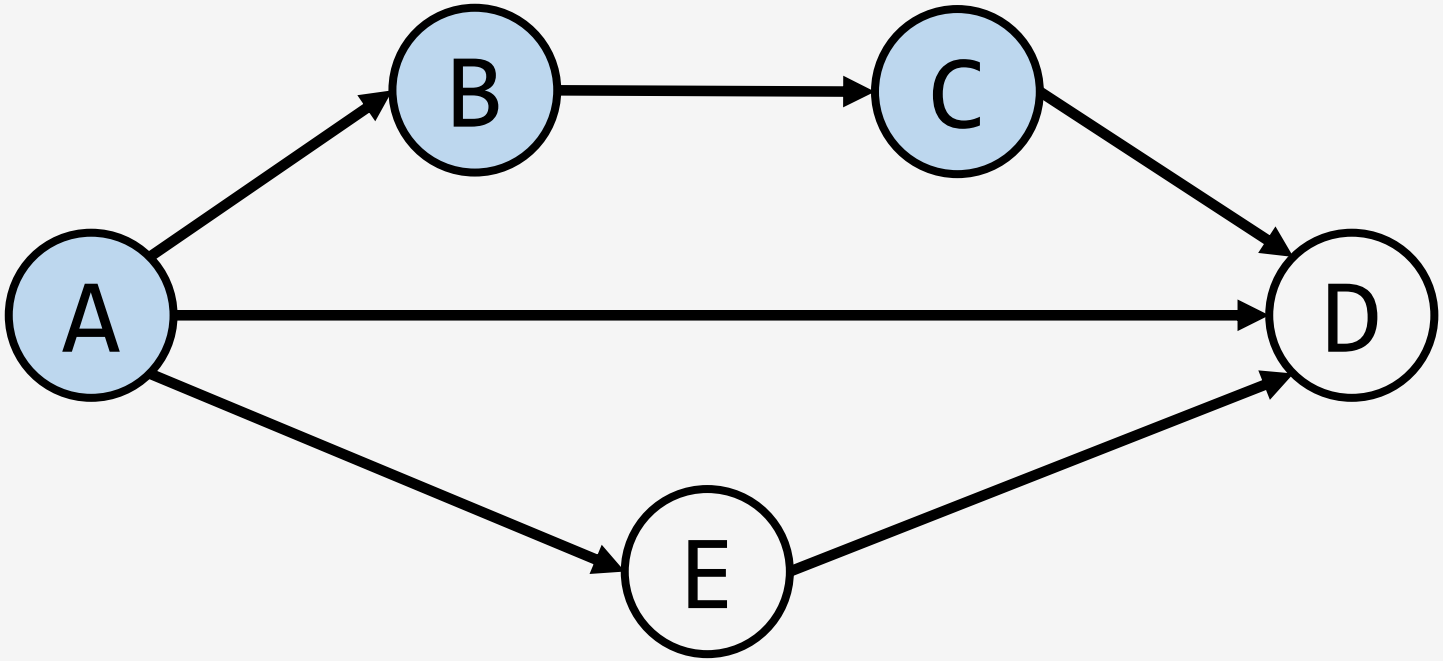
Example



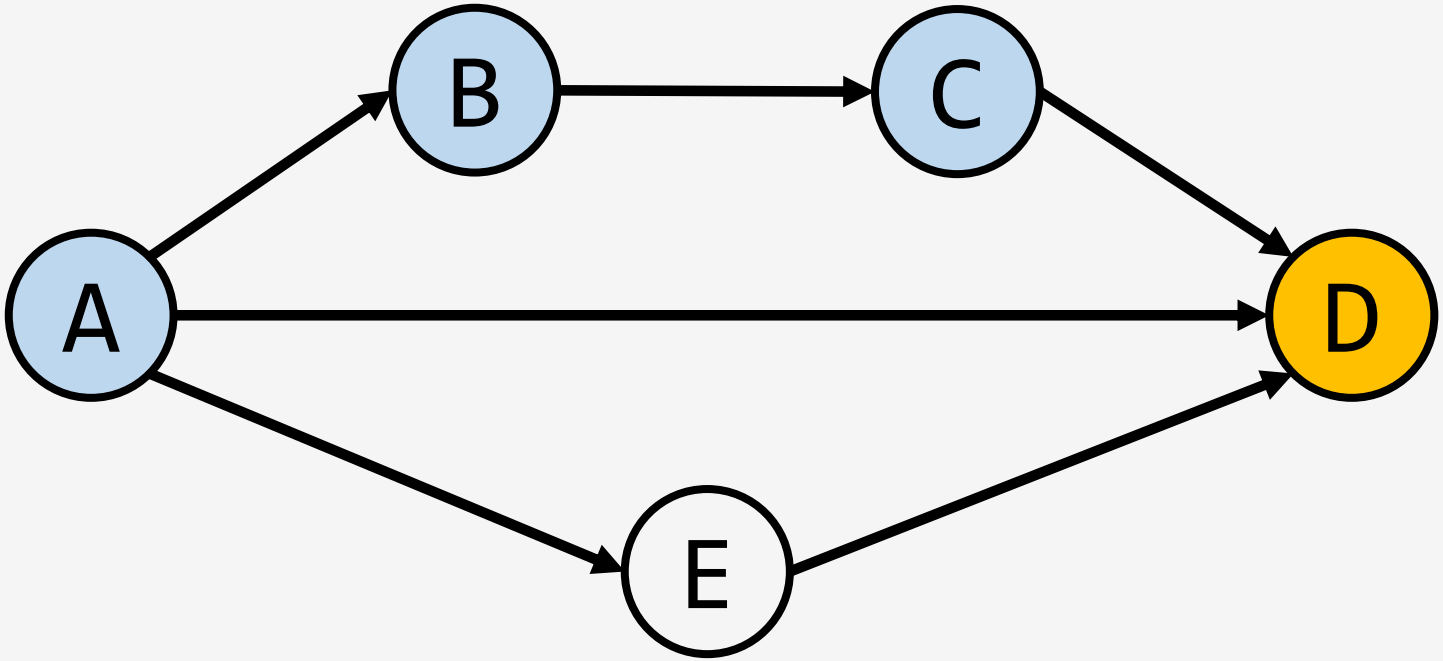
Example



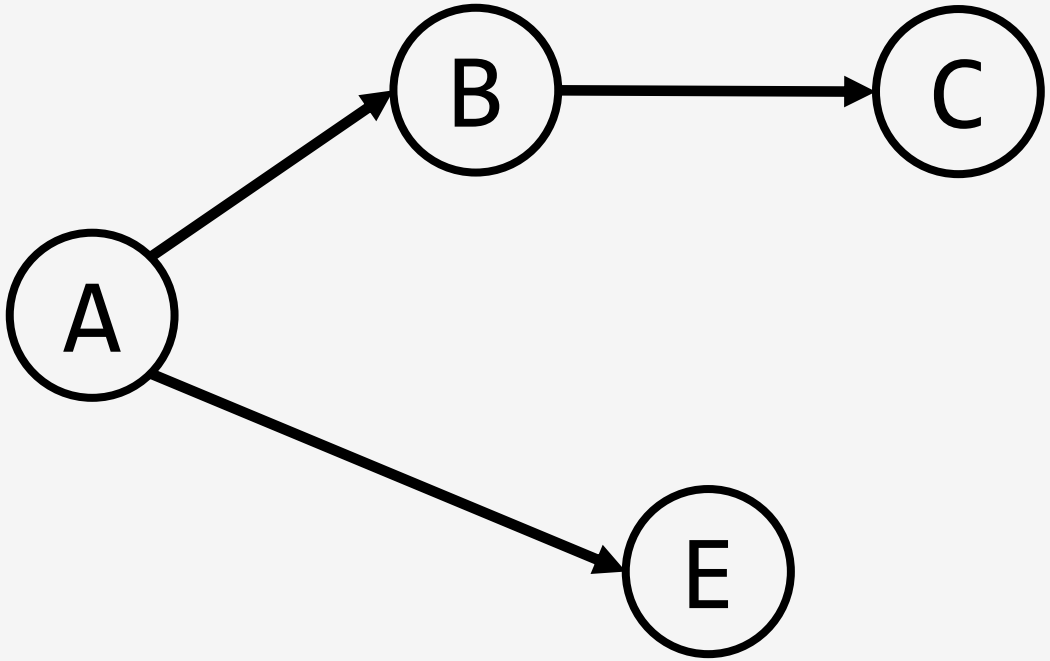
Example



Example

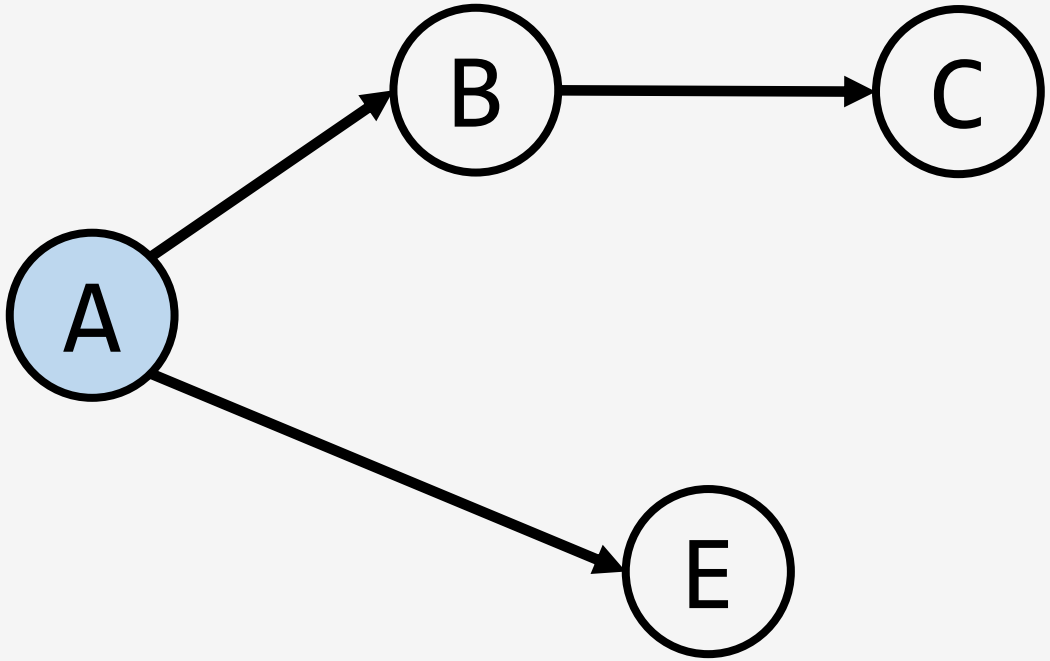


Example



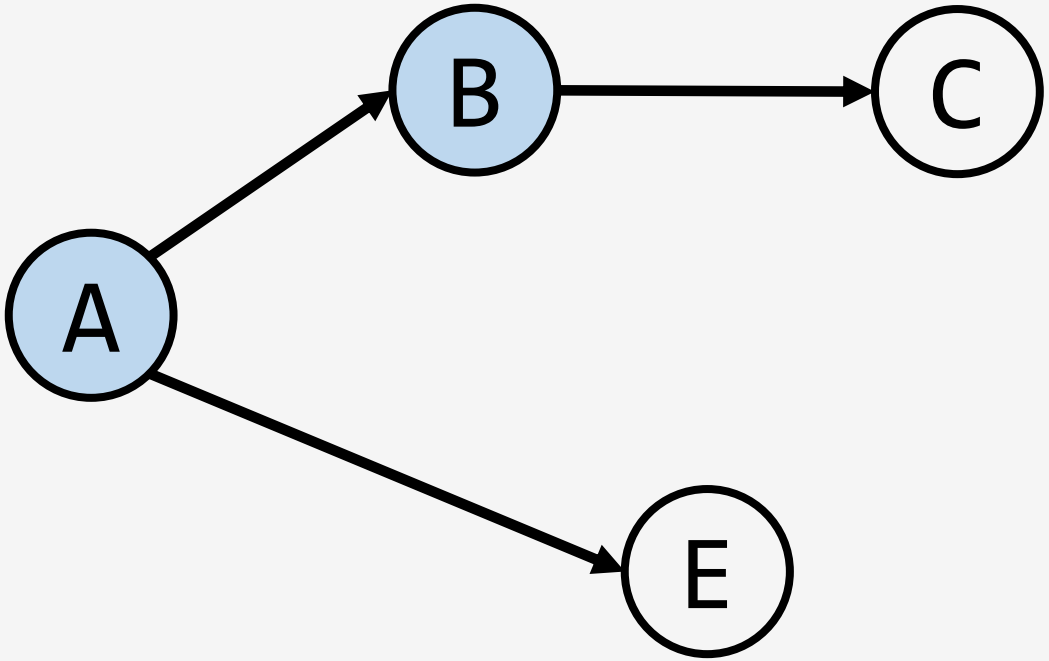
D

Example



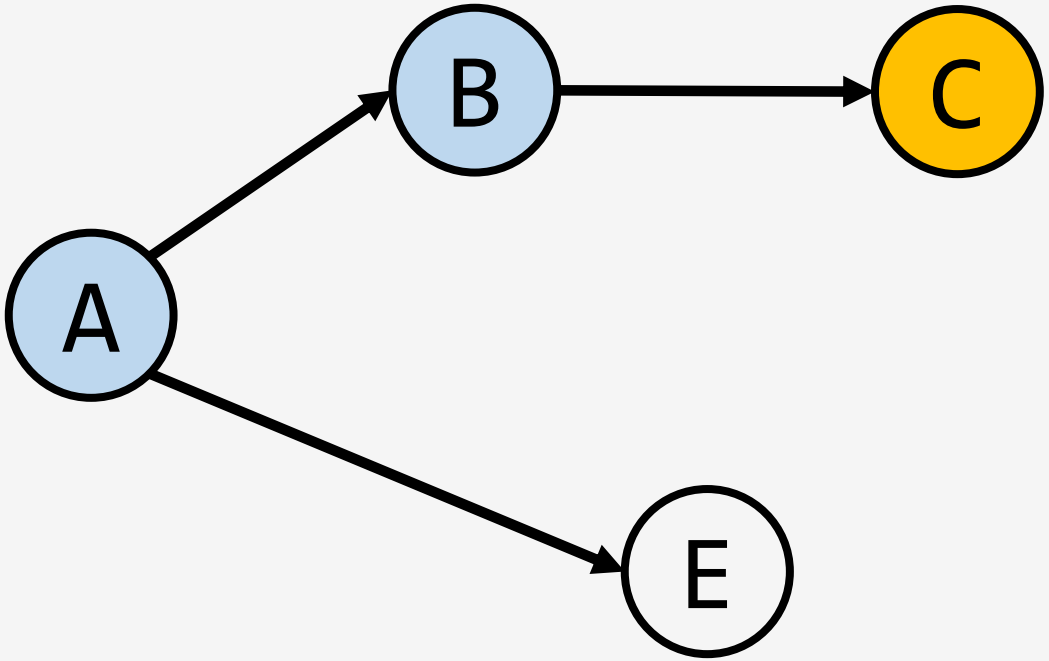
D

Example



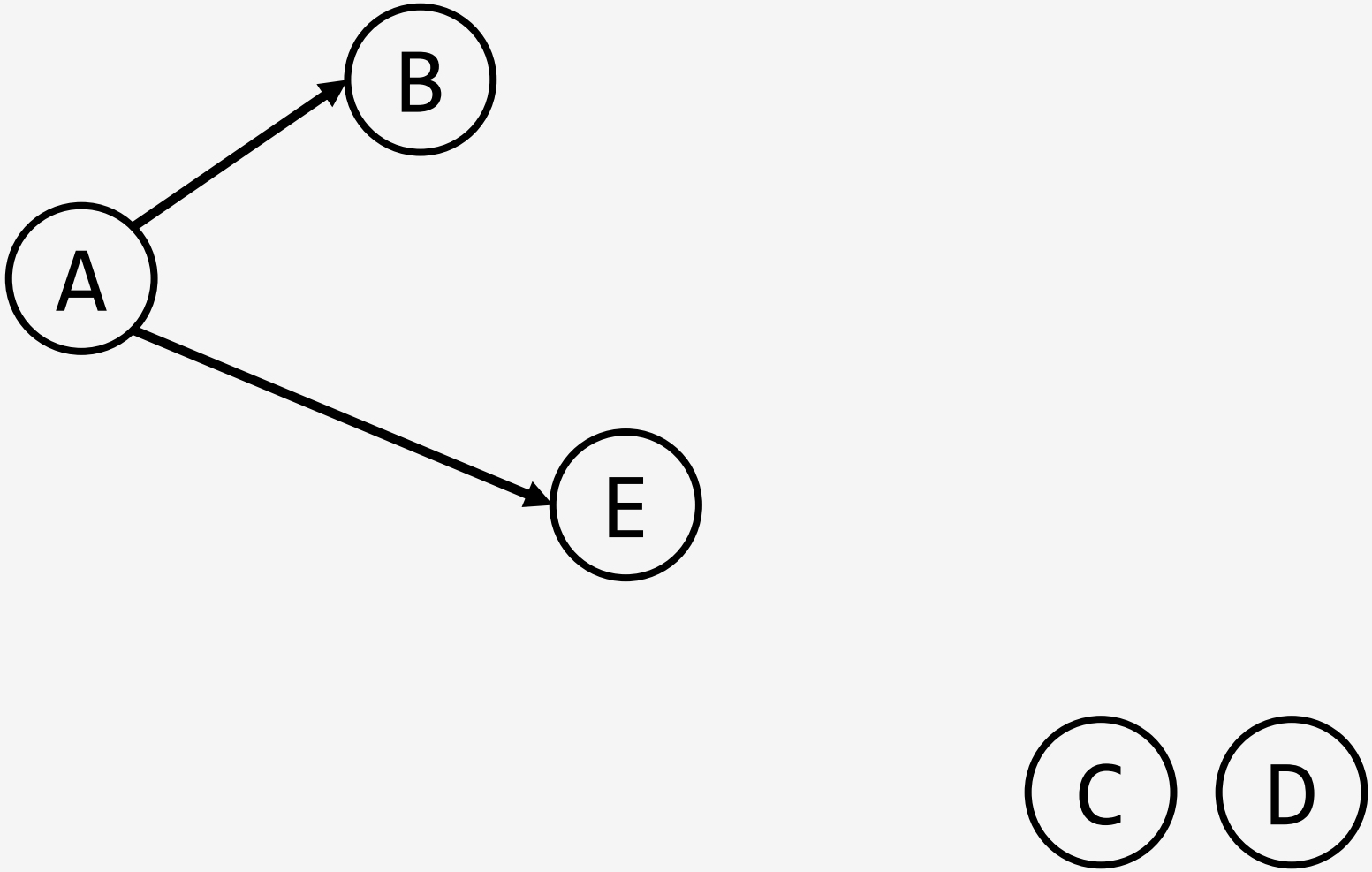
D

Example

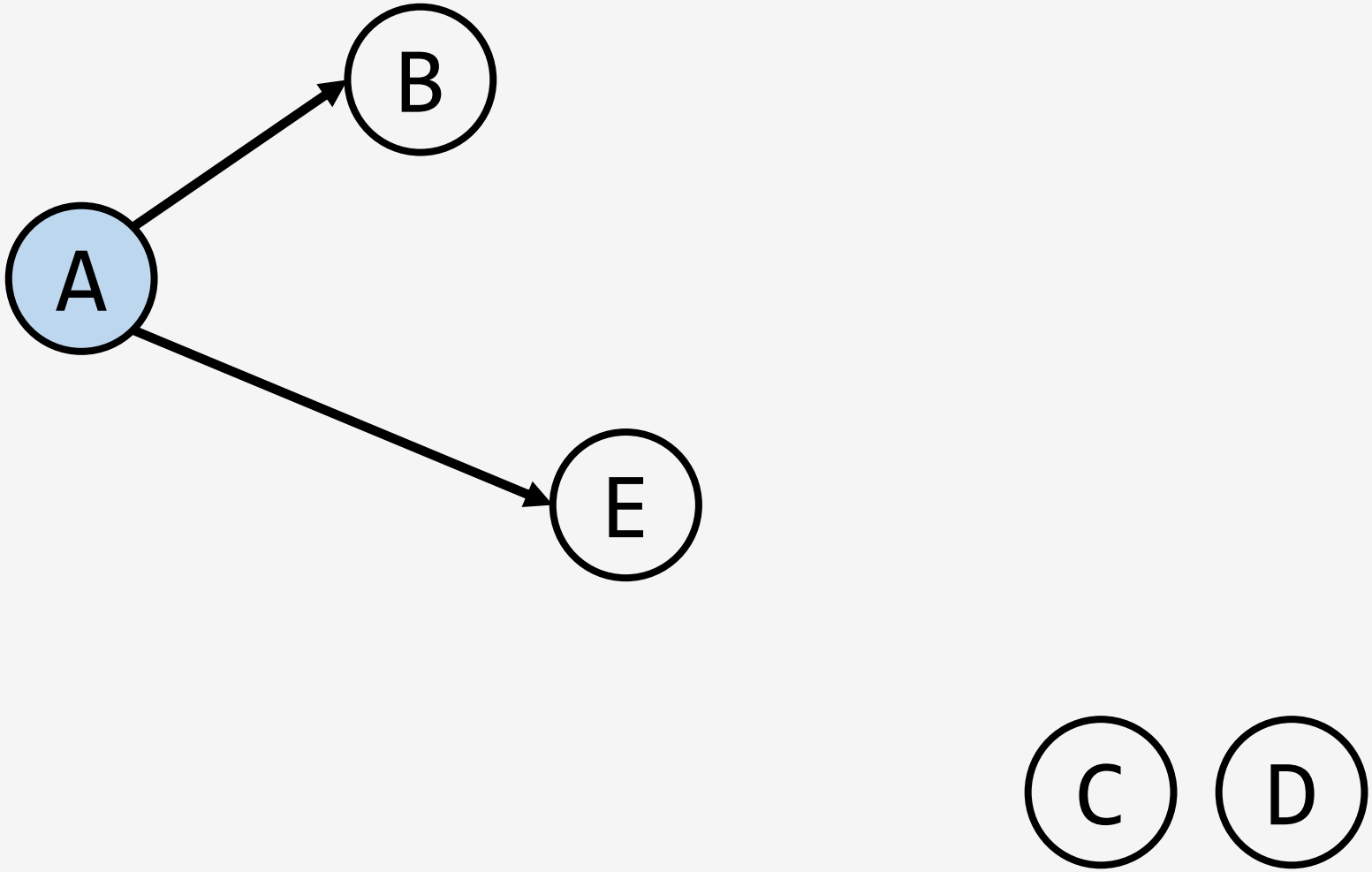


D

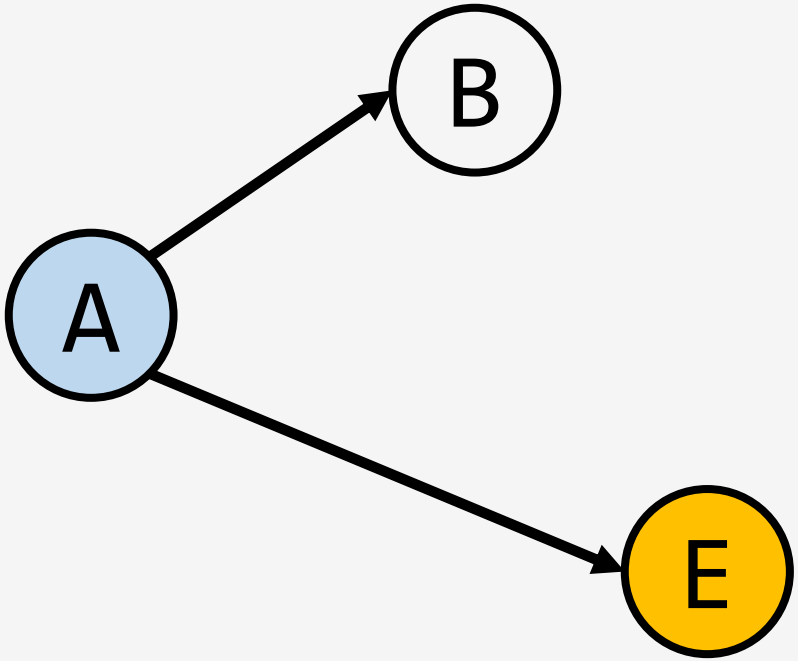
Example



Example

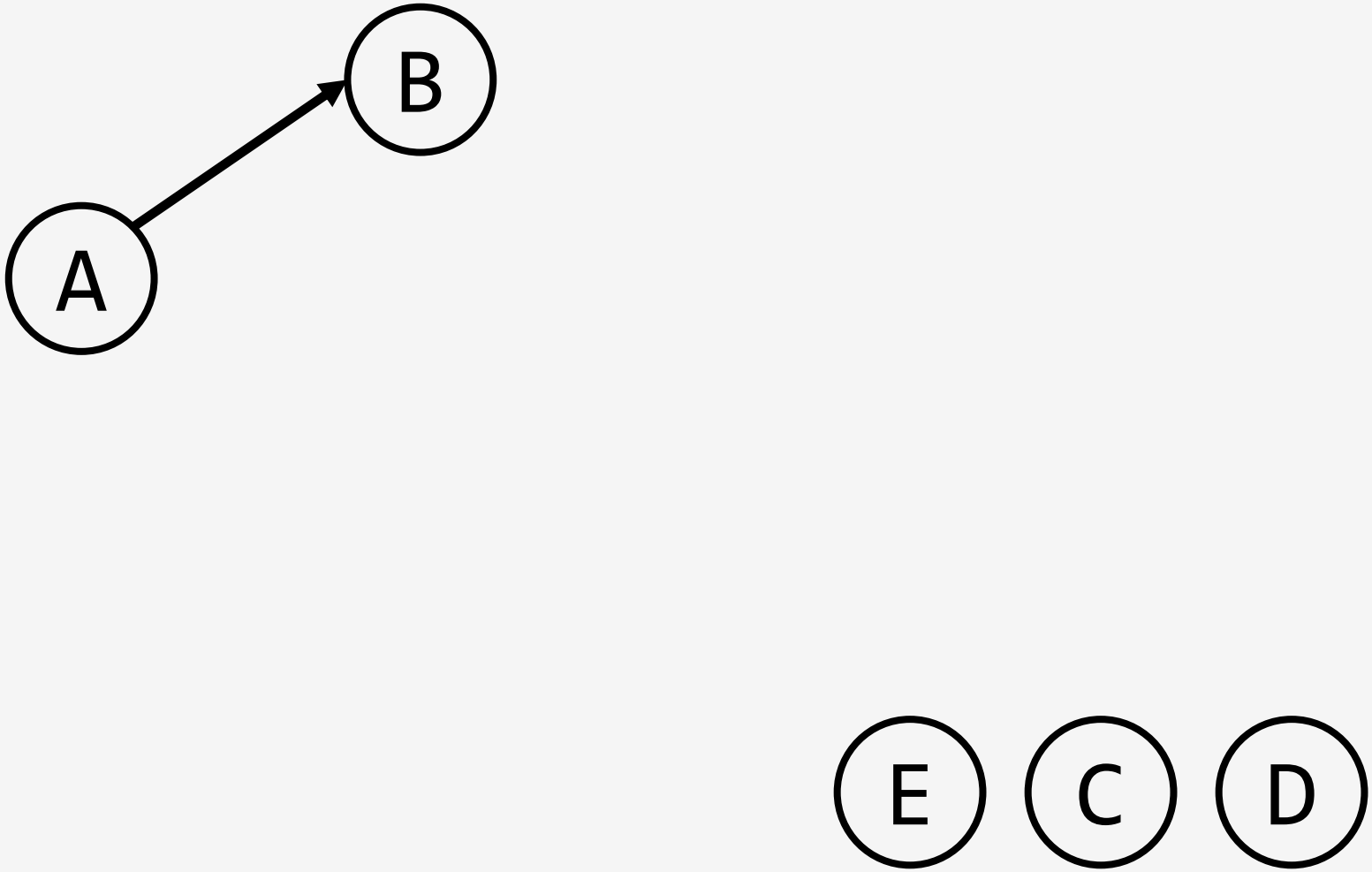


Example

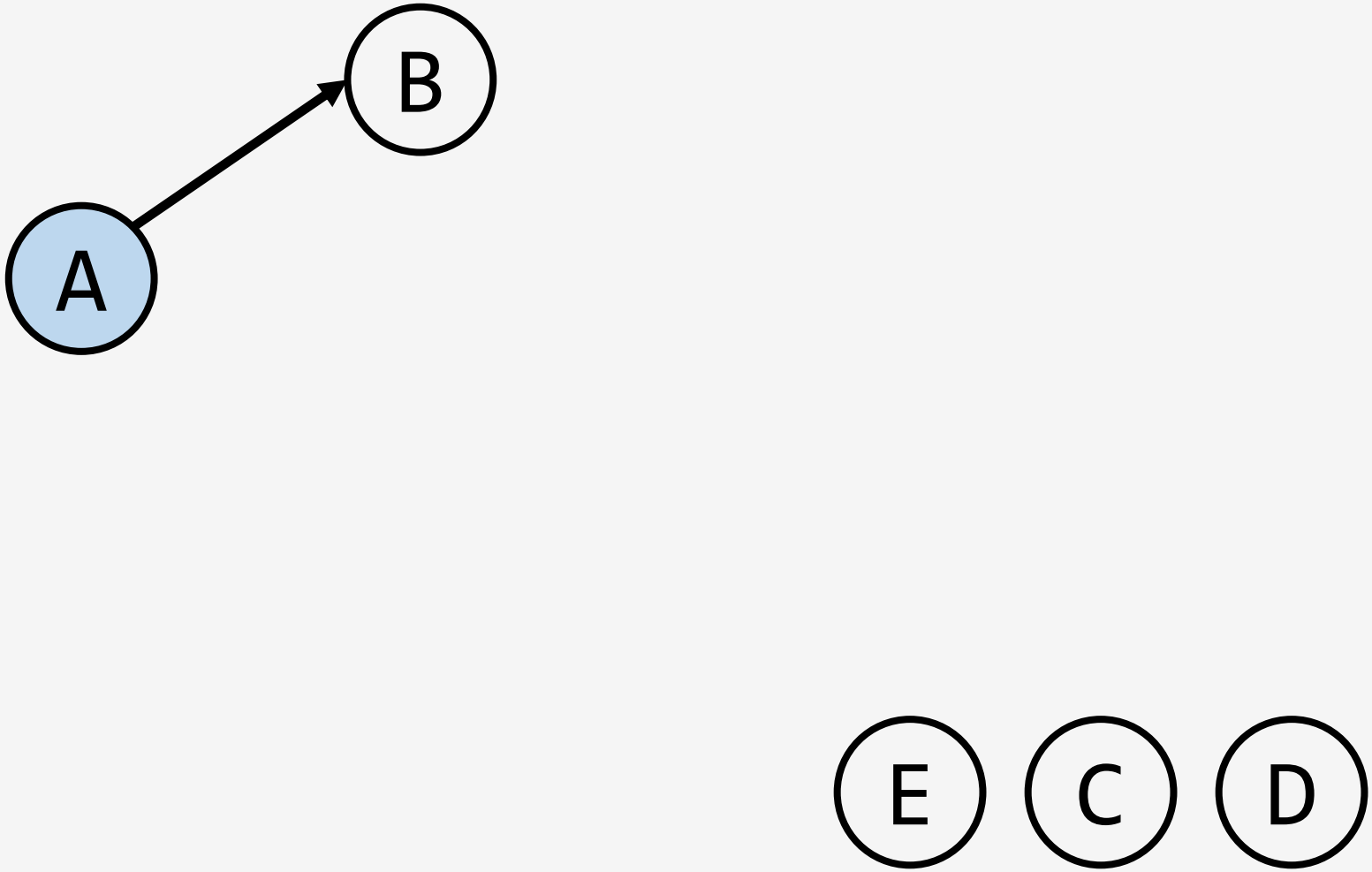


C D

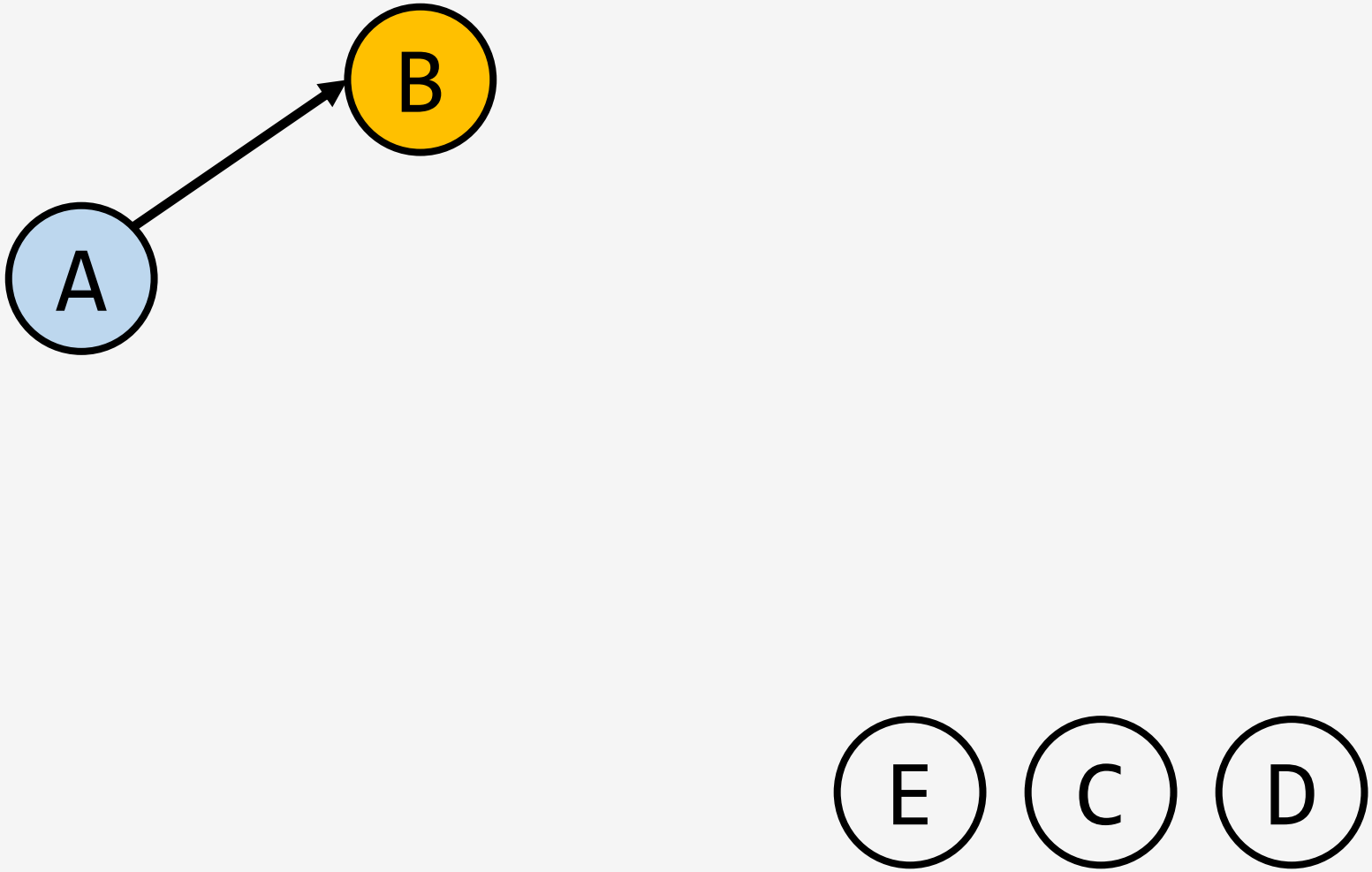
Example



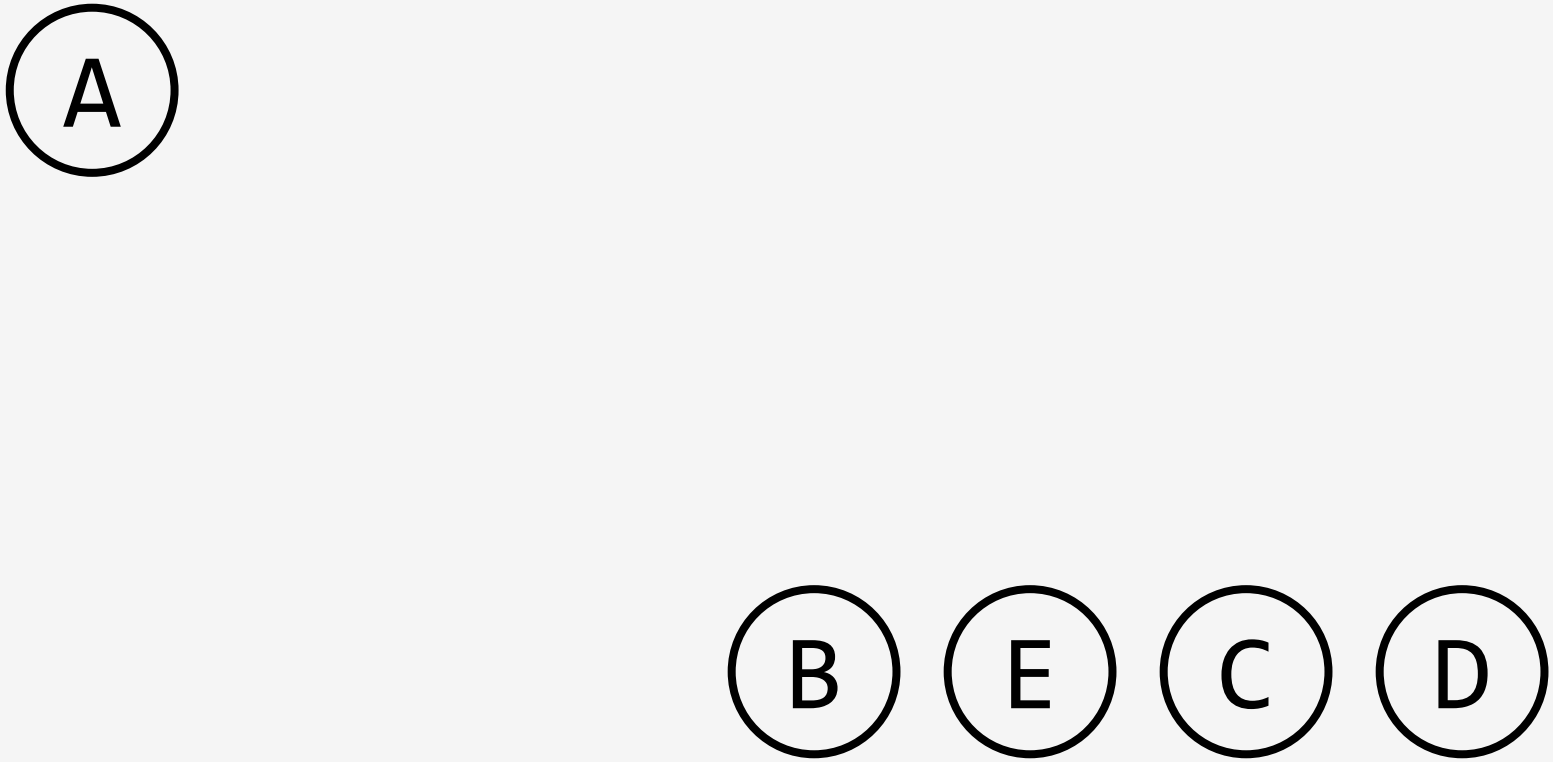
Example



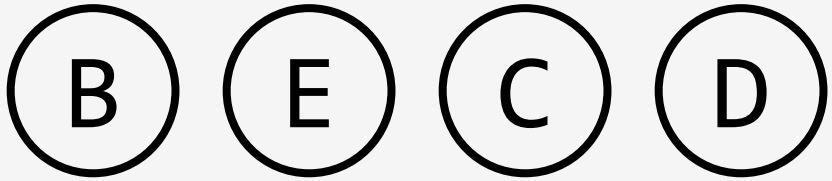
Example



Example



Example



Example

Runtime

O(|V|) paths.

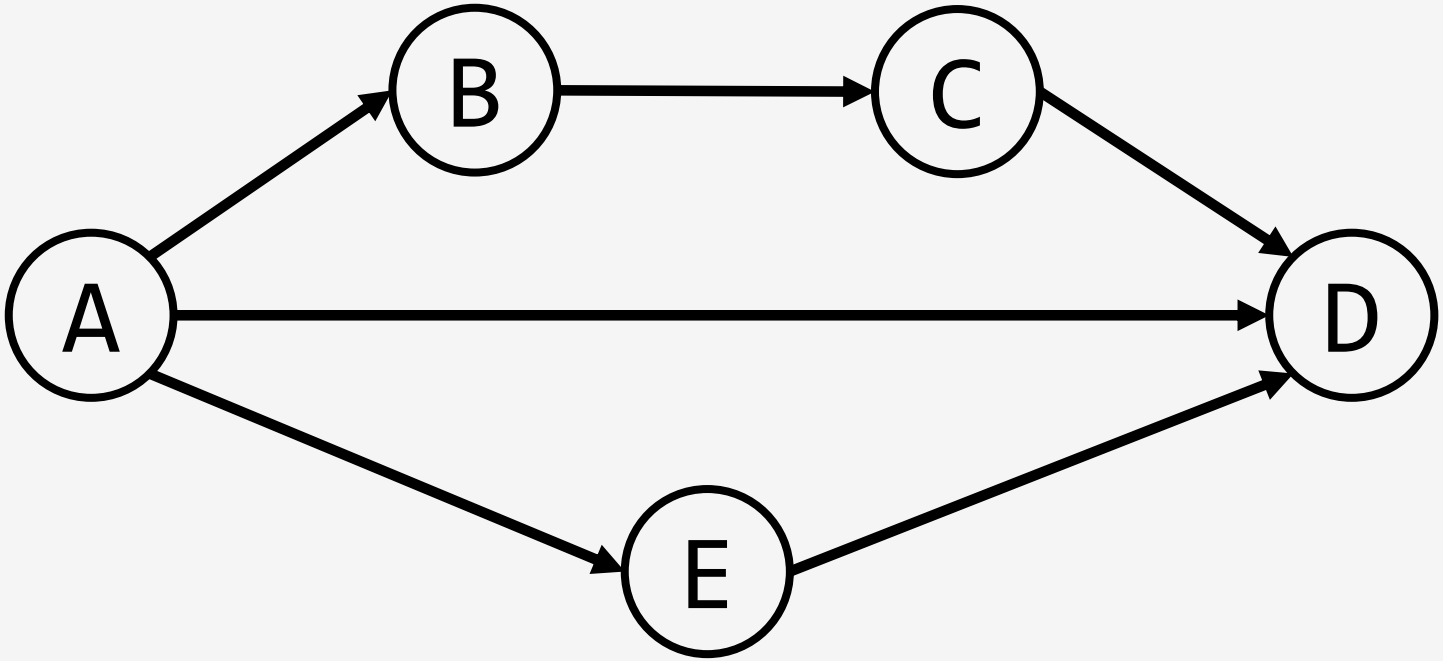
Each takes O(|V|) time.

Runtime $O(|V|^2)$.

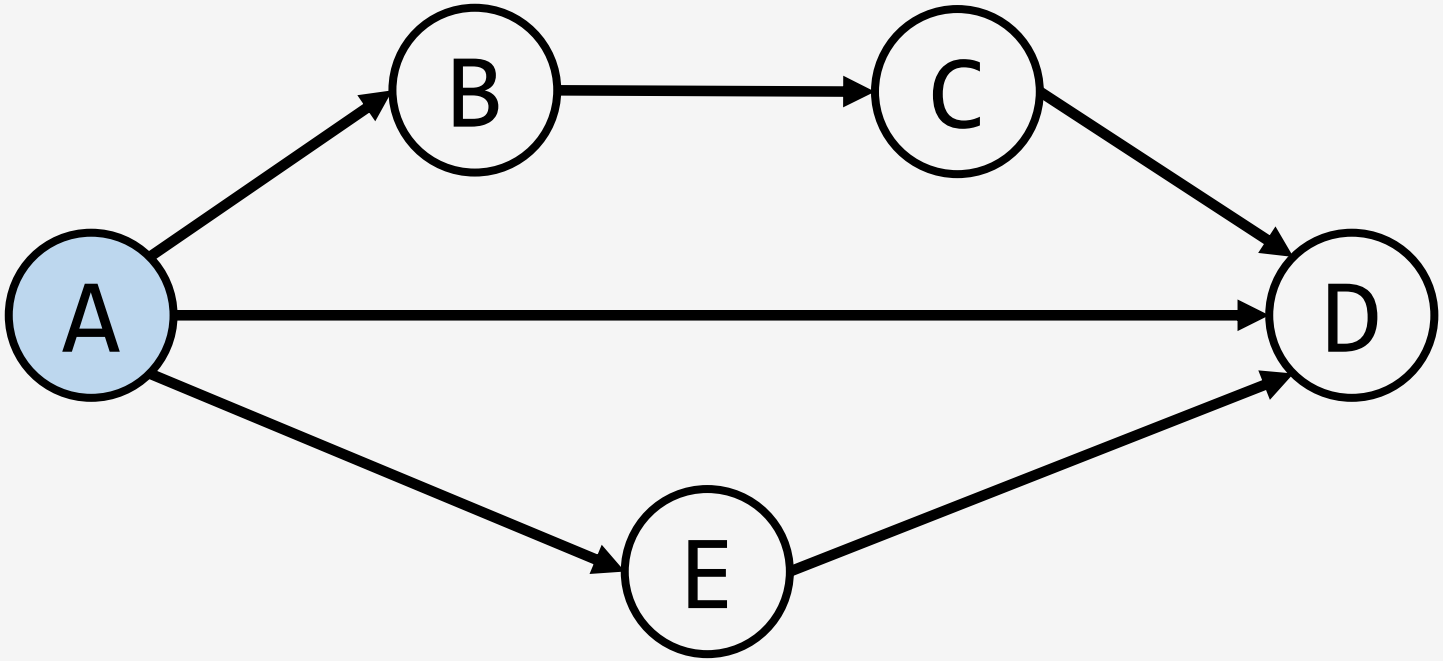
Speed Up

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

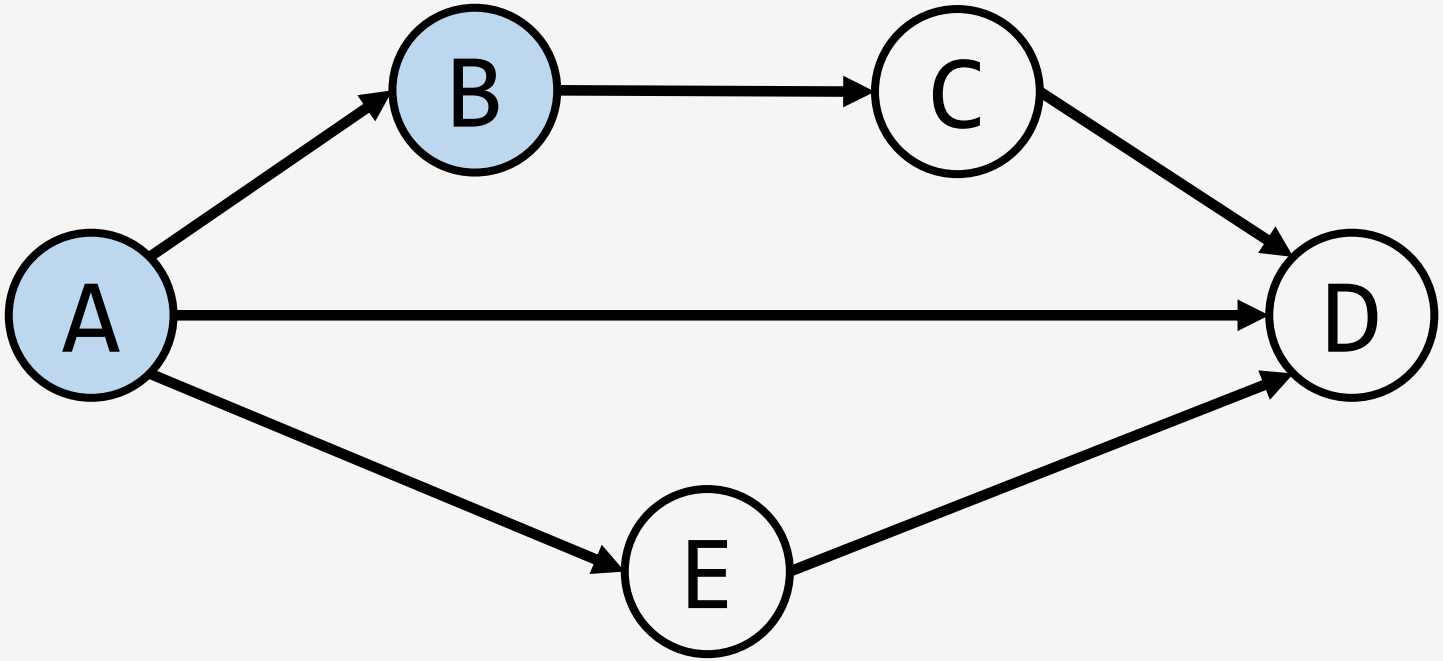
Example



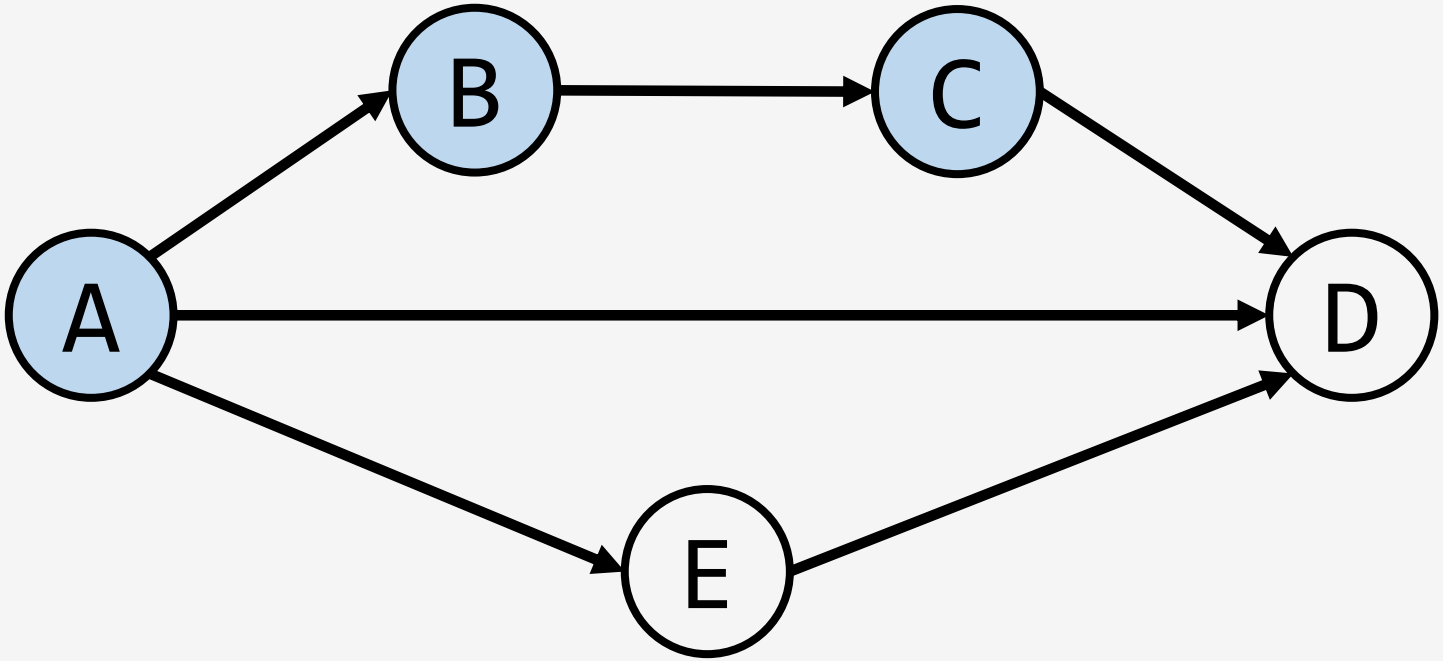
Example



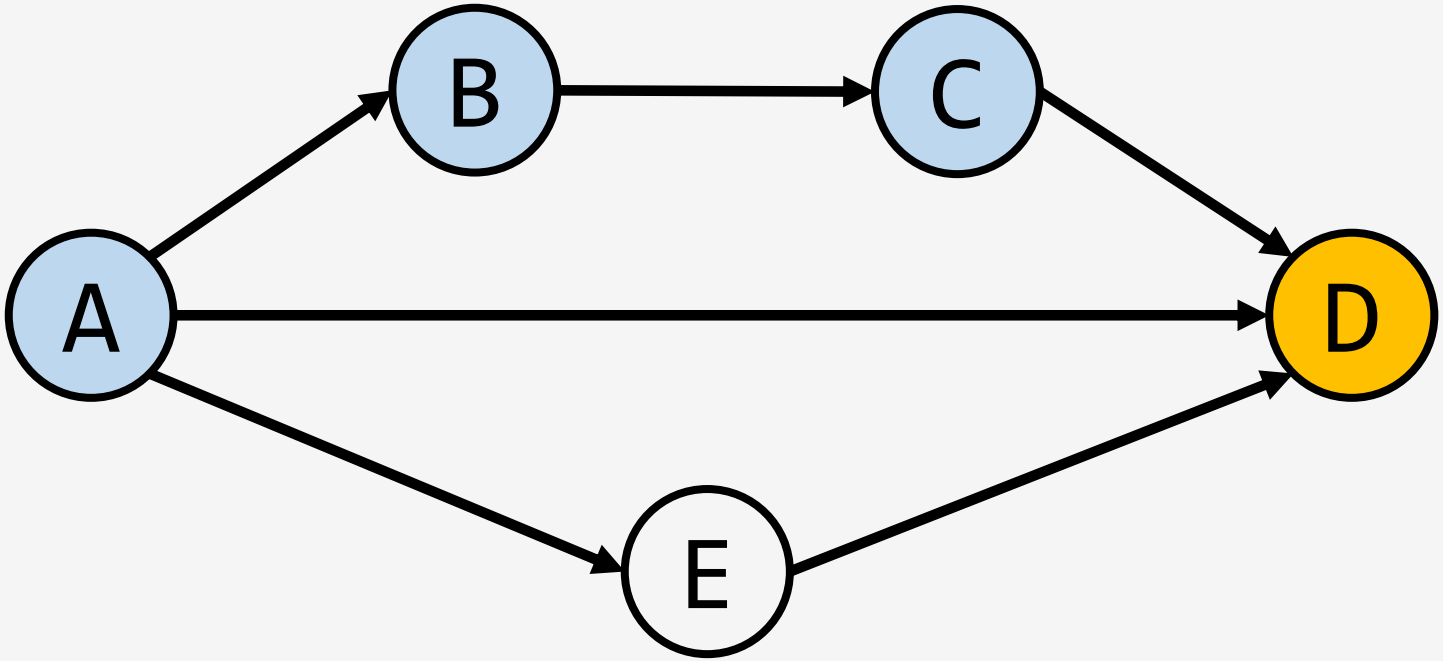
Example



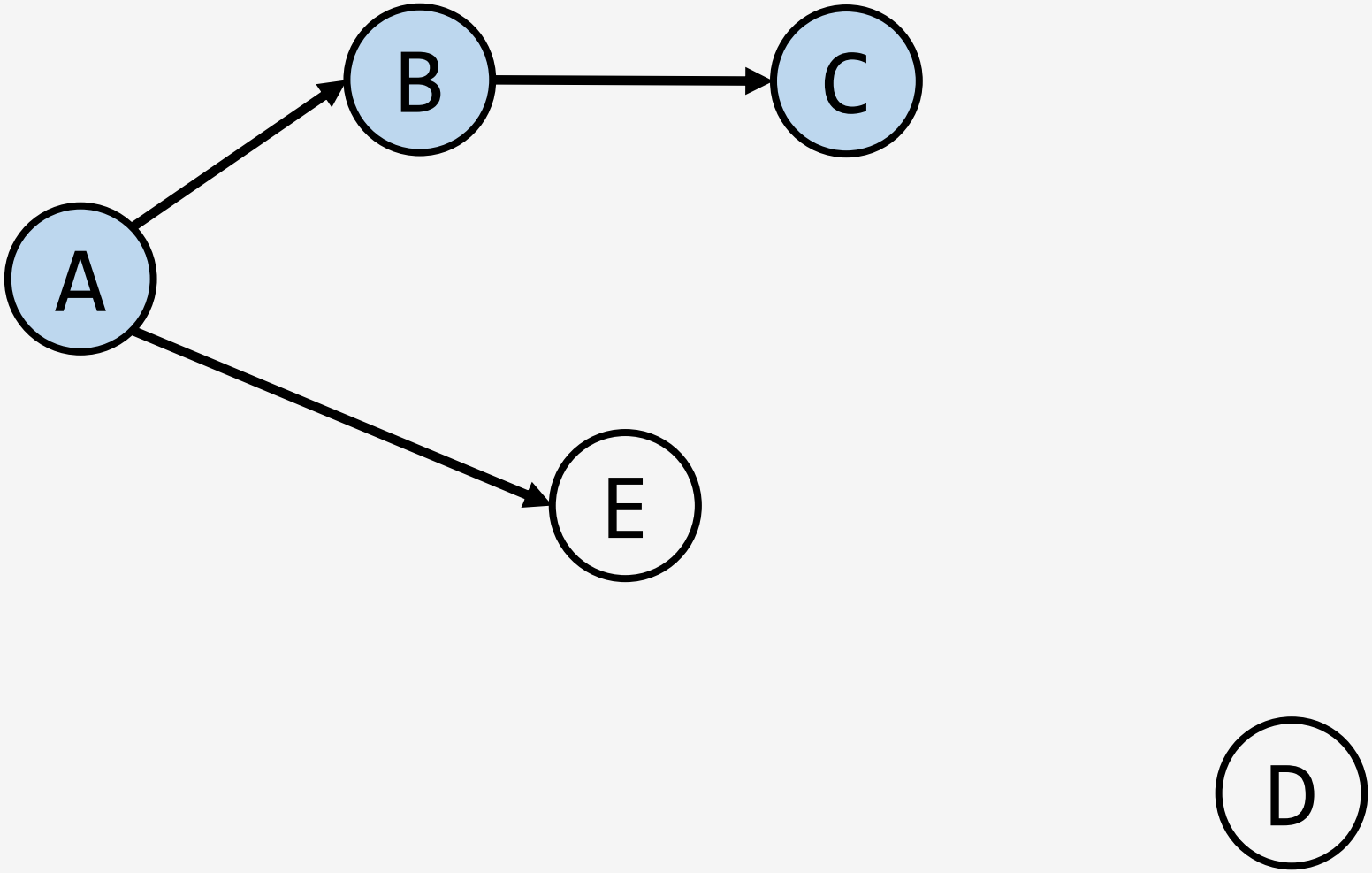
Example



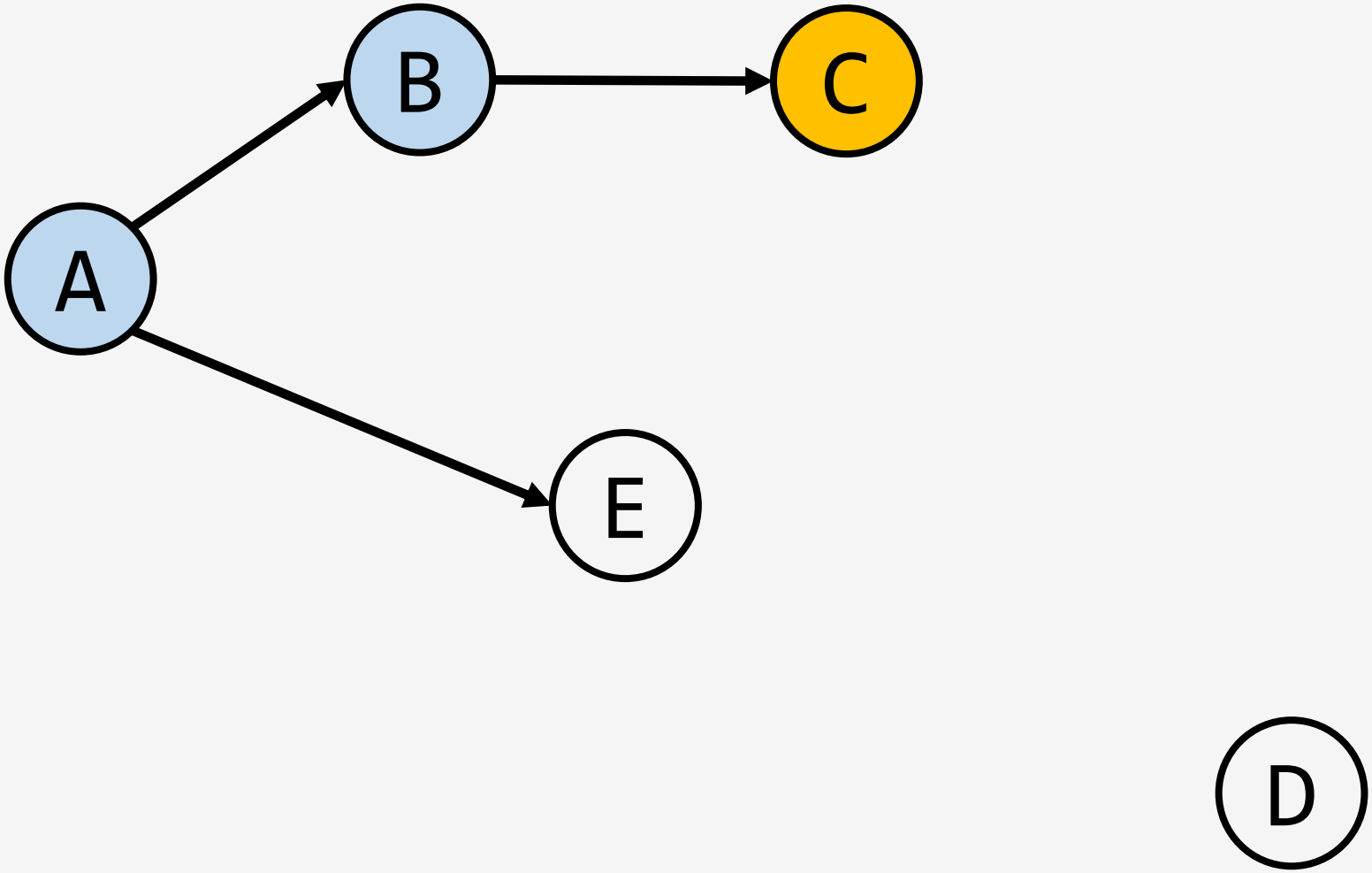
Example



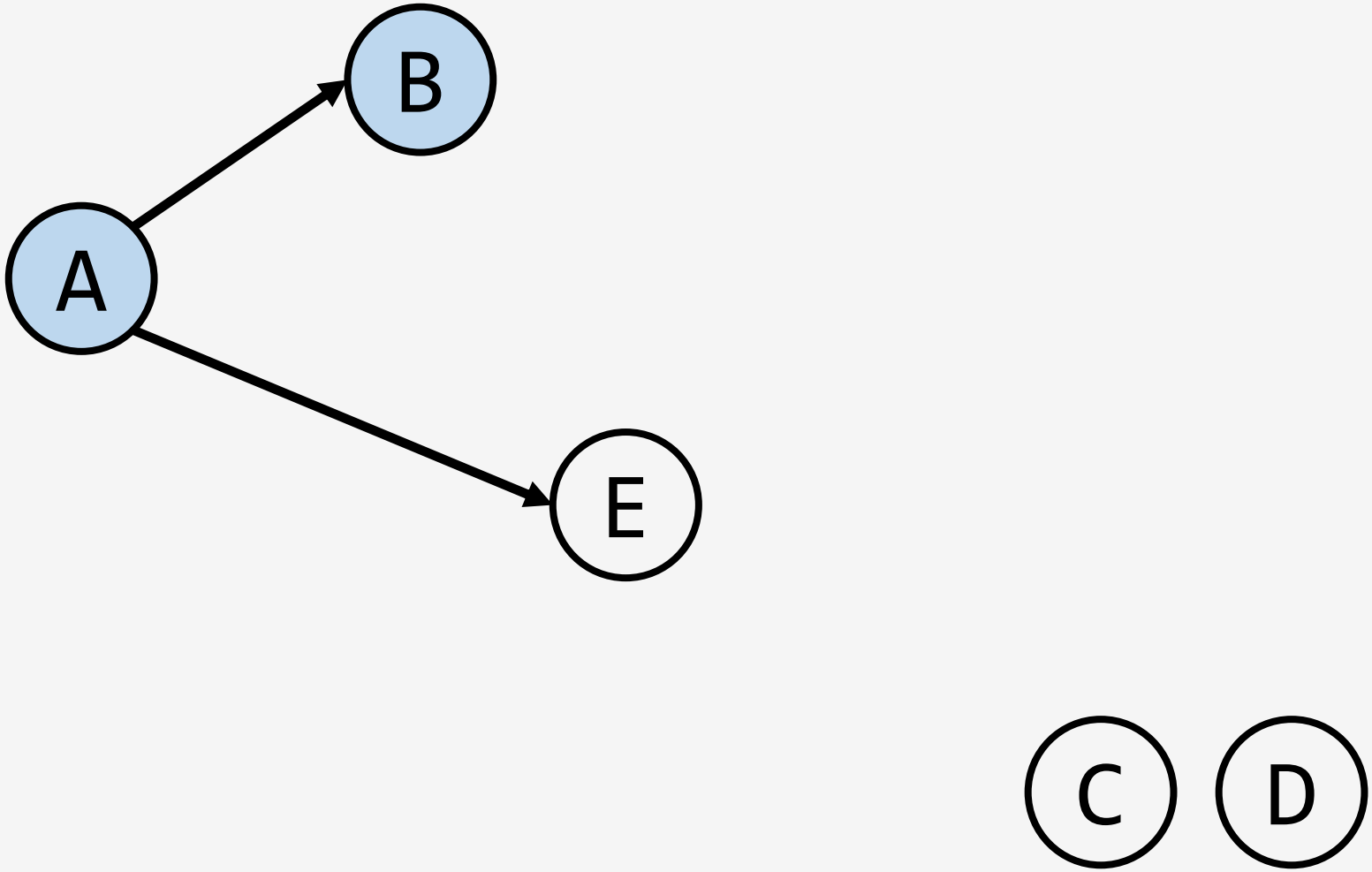
Example



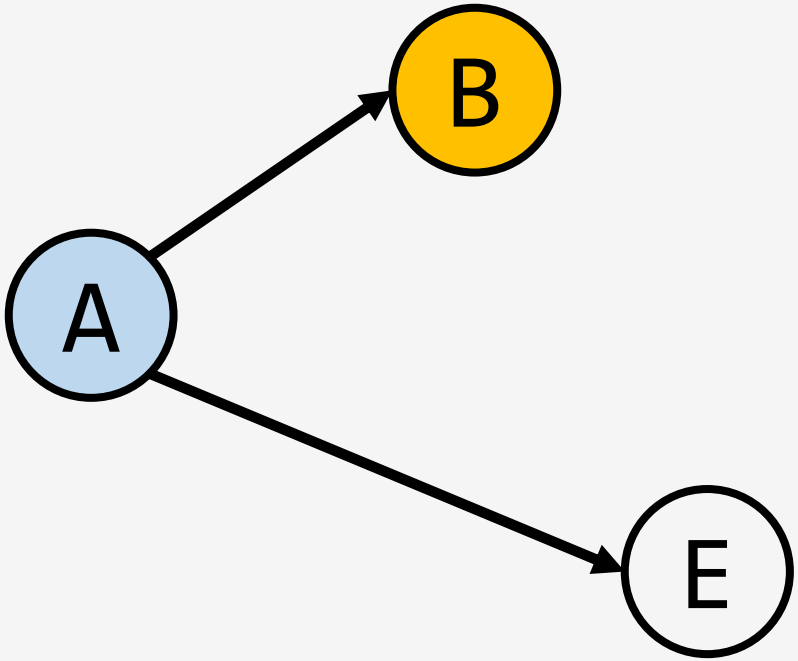
Example



Example

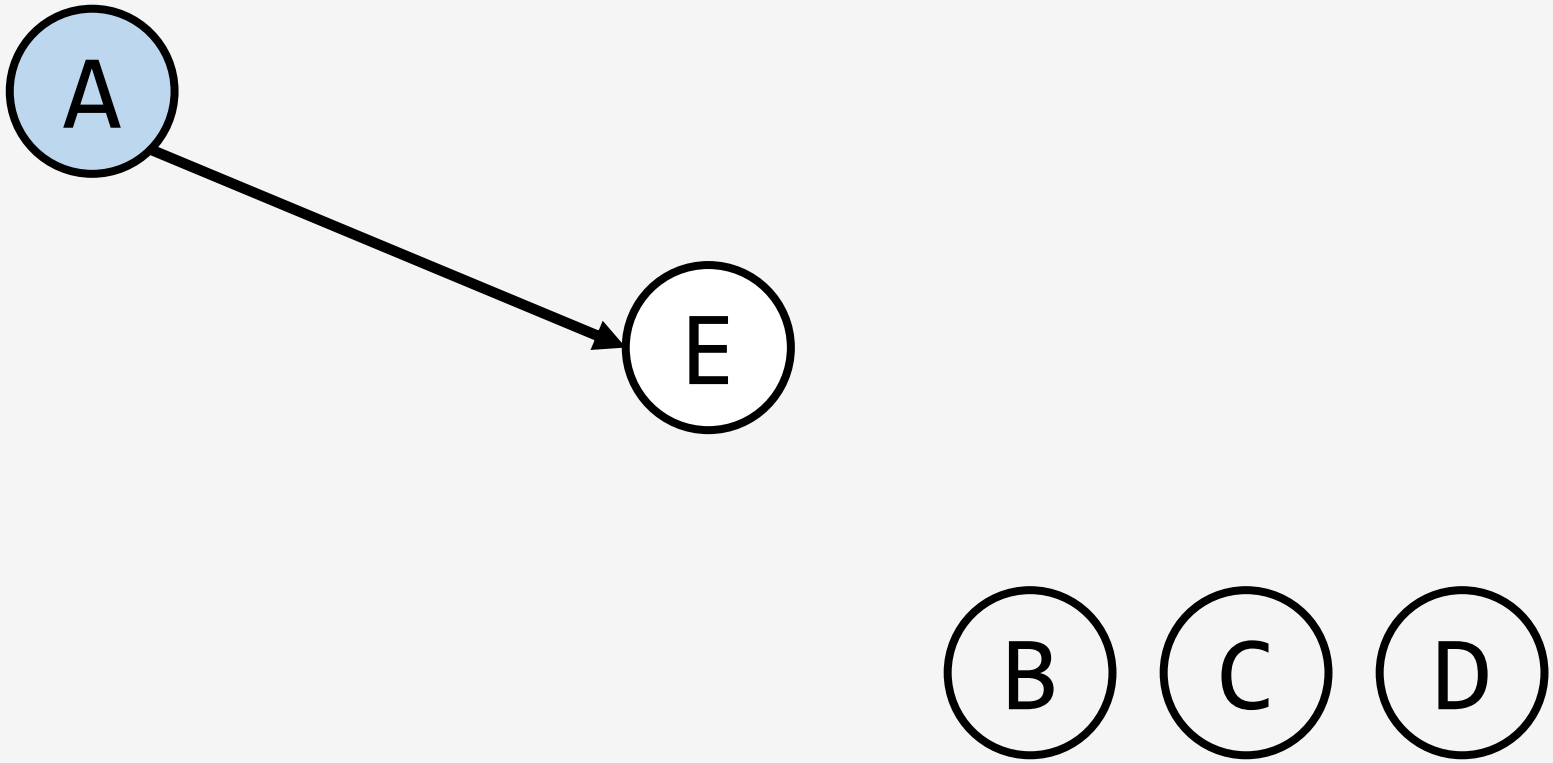


Example

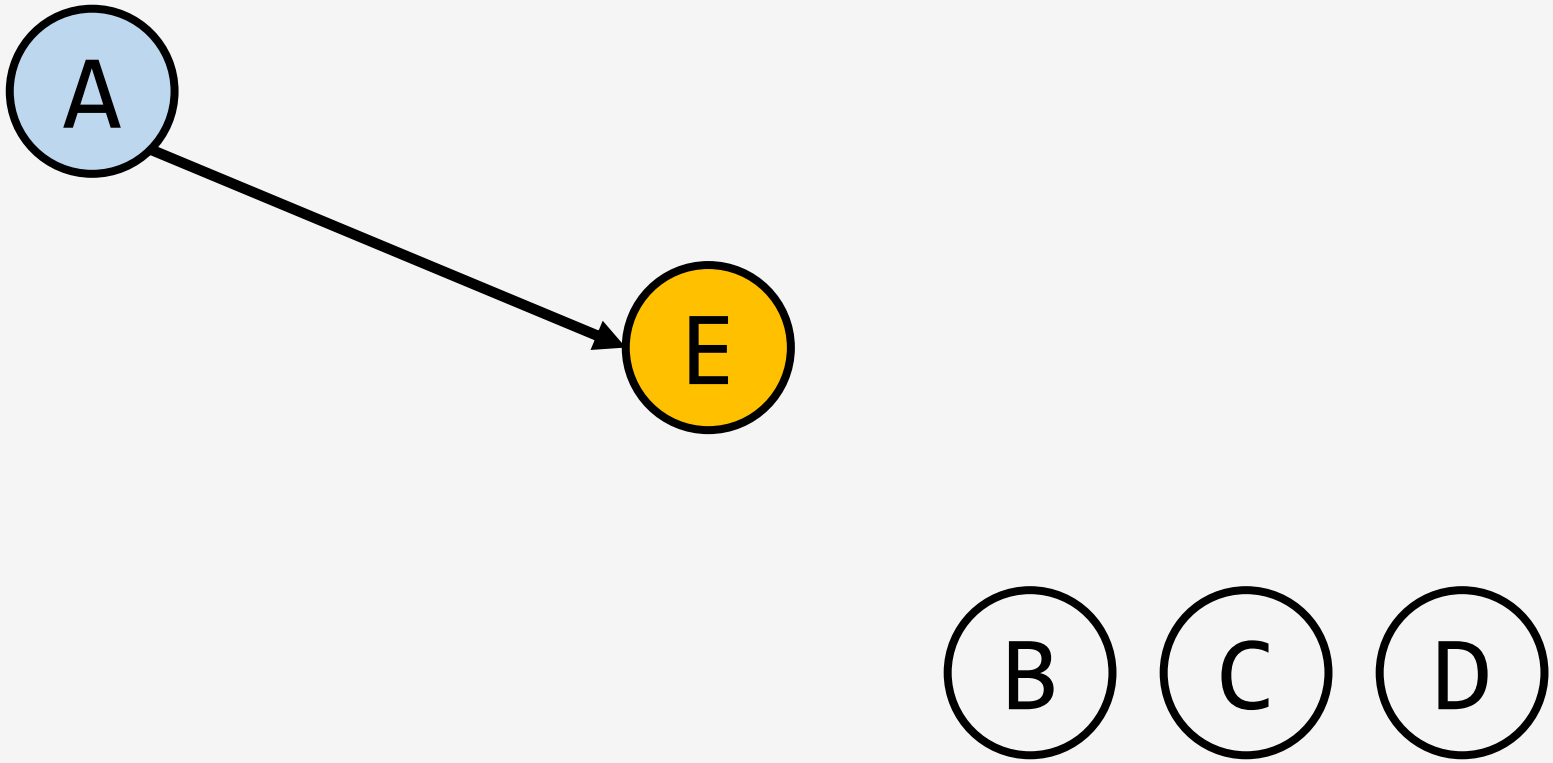


C D

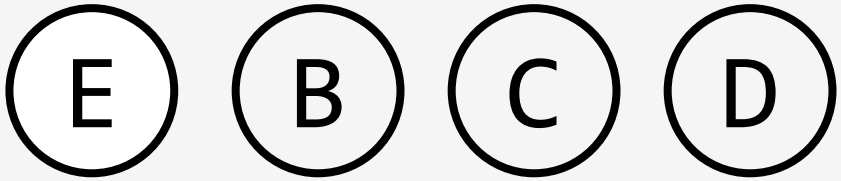
Example



Example



Example



Example

Example

Observation

This is just DFS!

We are sorting vertices based in post-order!

Better algorithm

```
TopologicalSort(G)
    DFS(G)
    sort vertices by reverse post-order
```