Cycles in Graphs

Stepan Kuznetsov

Computer Science Department, Higher School of Economics

Outline

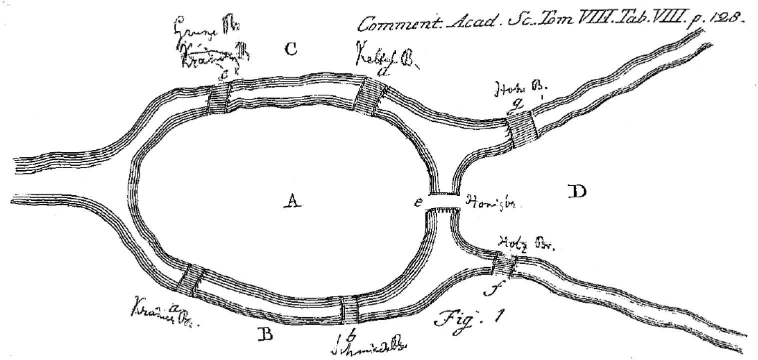
Königsberg Bridges. Euler Cycles

Constructing a Euler Cycle

Hamiltonian Paths

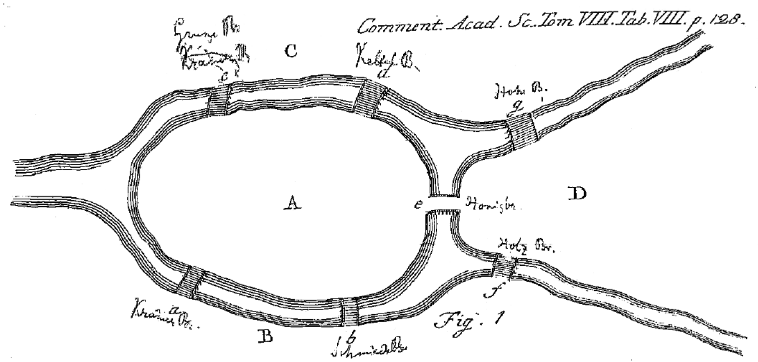
Acyclic Directed Graphs. Topological Sorting

The Seven Bridges of Königsberg



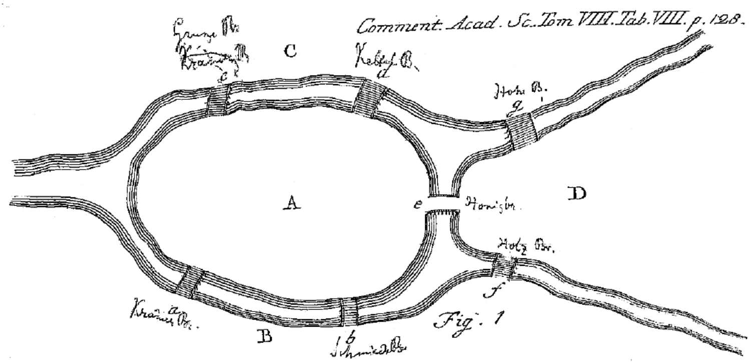
- A notable question by Leonhard Euler (1736).

The Seven Bridges of Königsberg



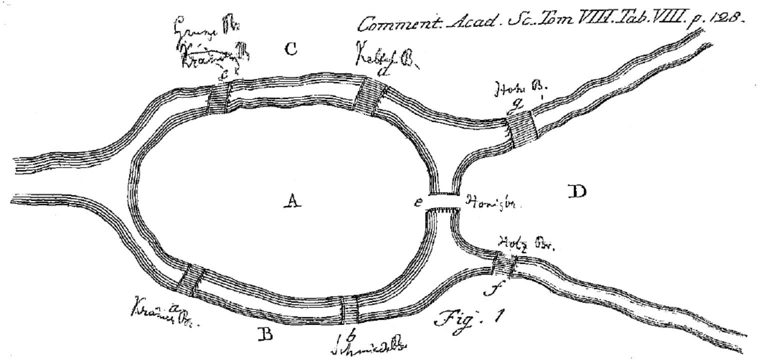
- A notable question by Leonhard Euler (1736).
- You see a map of Königsberg (nowadays Kaliningrad) from the times of Euler.

The Seven Bridges of Königsberg



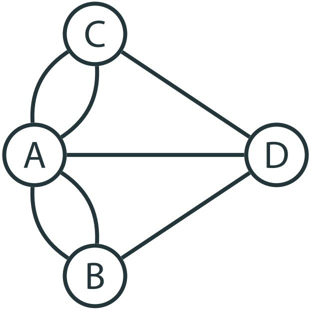
- Two islands (A and D) and two banks of the river (C and B) were connected by 7 bridges.

The Seven Bridges of Königsberg



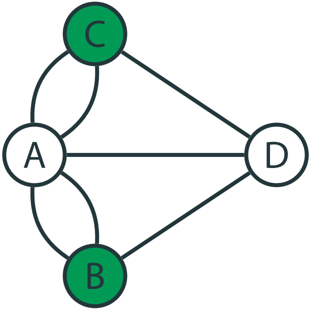
- Two islands (A and D) and two banks of the river (C and B) were connected by 7 bridges.
- Could one visit each bridge **exactly** once?

Königsberg Bridges as a Graph



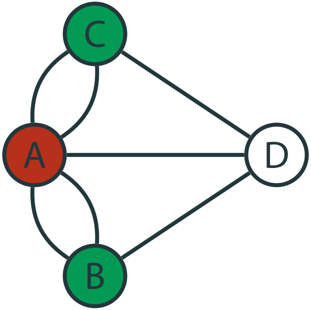
- City areas = vertices; bridges = edges.

Königsberg Bridges as a Graph



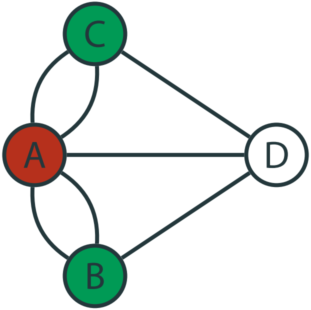
- City areas = vertices; bridges = edges.
- Suppose our tour starts at B, ends at C and visits each edge exactly once.

Königsberg Bridges as a Graph



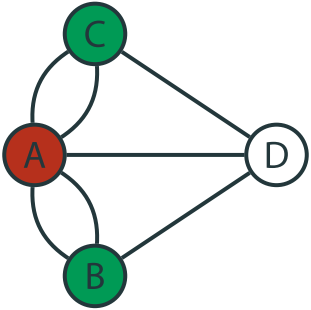
- City areas = vertices; bridges = edges.
- Suppose our tour starts at B, ends at C and visits each edge exactly once.
- Vertex A is visited **in the middle.**

Königsberg Bridges as a Graph



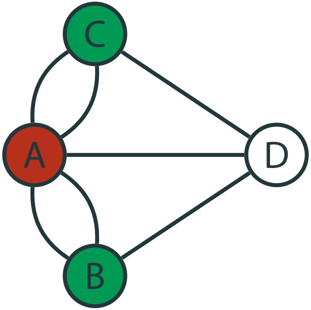
- Vertex A is visited, say, k times.

Königsberg Bridges as a Graph



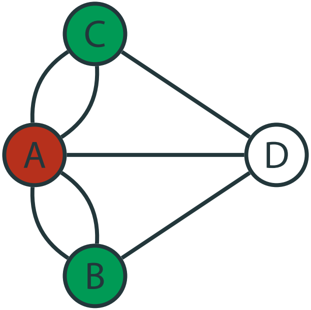
- Vertex A is visited, say, k times.
- The number of edges adjacent to A is $2k$.

Königsberg Bridges as a Graph



- Vertex A is visited, say, k times.
- The number of edges adjacent to A is $2k$.
- **Contradiction:** 5 is odd!

Königsberg Bridges as a Graph



- Vertex A is visited, say, k times.
- The number of edges adjacent to A is $2k$.
- **Contradiction:** 5 is odd!
- All vertices are odd, thus, Euler's problem is unsolvable.

Euler Paths and Cycles

Formal definitions:

- A Euler path is a path in the graph which visits each edge exactly once.

Euler Paths and Cycles

Formal definitions:

- A Euler path is a path in the graph which visits each edge exactly once.
- A Euler cycle is a Euler path which starts and ends at the same vertex.

Euler Paths and Cycles

Formal definitions:

- A Euler path is a path in the graph which visits each edge exactly once.
- A Euler cycle is a Euler path which starts and ends at the same vertex.
- An odd vertex is a vertex which has an odd number of edges adjacent to it.

Euler Paths and Cycles

- If there exists a Euler path, then the graph has at most two odd vertices.

Euler Paths and Cycles

- If there exists a Euler path, then the graph has at most two odd vertices.
- If there exists a Euler cycle, then the graph has no odd vertices.

Euler Paths and Cycles

- If there exists a Euler path, then the graph has at most two odd vertices.
- If there exists a Euler cycle, then the graph has no odd vertices.
- **Question:** can a graph have exactly one odd vertex?

Outline

Königsberg Bridges. Euler Cycles

Constructing a Euler Cycle

Hamiltonian Paths

Acyclic Directed Graphs. Topological Sorting

The Problem

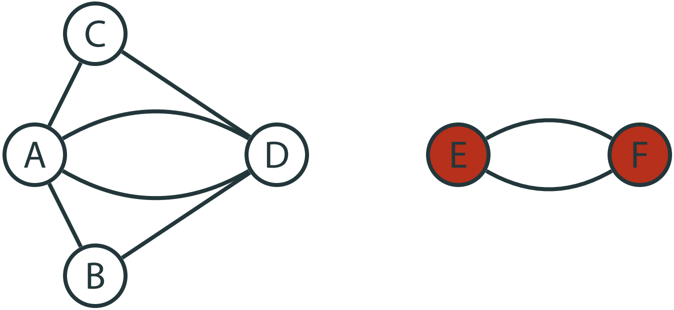
- Suppose our graph has no odd vertices. Does this guarantee existence of a Euler cycle?

The Problem

- Suppose our graph has no odd vertices. Does this guarantee existence of a Euler cycle?
- In general, **no:** there could be other obstacles.

The Problem

- Suppose our graph has no odd vertices. Does this guarantee existence of a Euler cycle?
- In general, **no:** there could be other obstacles.
- E. g., the graph could be **disconnected:**



Greedy Algorithm

- OK, OK, suppose the graph is connected: there is a path between any two vertices...

Greedy Algorithm

- OK, OK, suppose the graph is connected: there is a path between any two vertices...
- ... and there are no odd vertices ...

Greedy Algorithm

- OK, OK, suppose the graph is connected: there is a path between any two vertices...
- ... and there are no odd vertices ...
- ... then let's just start walking!

Greedy Algorithm

- No odd vertices, so we never get stuck ...

Greedy Algorithm

- No odd vertices, so we never get stuck ...
- ... at some point, we return to the starting vertex, completing the cycle ...

Greedy Algorithm

- No odd vertices, so we never get stuck ...
- ... at some point, we return to the starting vertex, completing the cycle ...
- ... but what if there is still something left?

Outline

Königsberg Bridges. Euler Cycles

Constructing a Euler Cycle

Hamiltonian Paths

Acyclic Directed Graphs. Topological Sorting

Hamiltonian Paths and Cycles

- The notion of *Hamiltonian path* is much like the Euler one, but a Hamiltonian path must visit each **vertex** exactly once.

Hamiltonian Paths and Cycles

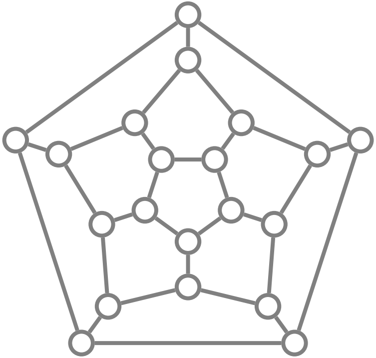
- The notion of *Hamiltonian path* is much like the Euler one, but a Hamiltonian path must visit each **vertex** exactly once.
- A Hamiltonian cycle is a Hamiltonian path which starts and ends at the same vertex (and this is counted as one visit).

Hamiltonian Cycle on a Dodecahedron

- A dodecahedron is a regular (Platonic) solid with 12 pentagonal faces.

Hamiltonian Cycle on a Dodecahedron

- A dodecahedron is a regular (Platonic) solid with 12 pentagonal faces.
- Using the polar projection, the dodecahedron can be represented as a graph on the plane:



Hamiltonian Cycles

- In contrast to Euler cycles, constructing Hamiltonian cycles requires some creativity.

Hamiltonian Cycles

- In contrast to Euler cycles, constructing Hamiltonian cycles requires some creativity.
- No “good” criterion whether a Hamiltonian cycle exists (in a given graph) is known.

Hamiltonian Cycles

- In contrast to Euler cycles, constructing Hamiltonian cycles requires some creativity.
- No “good” criterion whether a Hamiltonian cycle exists (in a given graph) is known.
- Unless P=NP, there is also no efficient algorithm for finding Hamiltonian cycles.

Knight's Tour

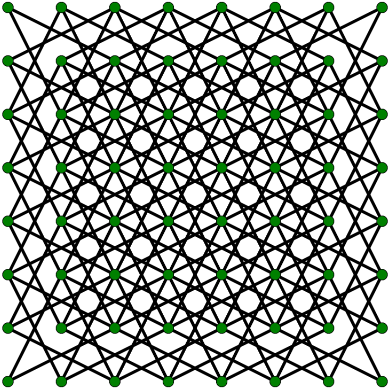
- A specific case of Hamiltonian cycle is the *knight tour question.*

Knight’s Tour

- A specific case of Hamiltonian cycle is the *knight tour question.*
- The challenge is to find a walk of a chess knight over the board, in which each square is visited exactly once.

Knight's Tour

As one can easily see, a knight tour is exactly a Hamiltonian path in the following *knight's graph:*



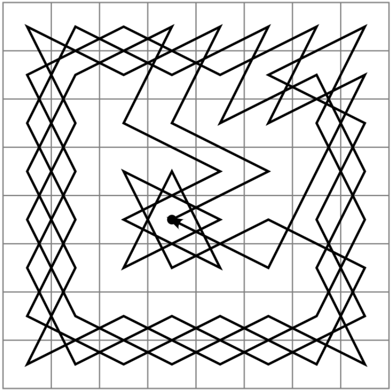
Wikipedia

Knight's Tour

- If it is a Hamiltonian cycle, then the knight tour is called *closed*.
- Otherwise, if the tour does not return to the initial square, it is *open*.

Knight's Tour

Example of a closed knight's tour on the standard 8×8 chessboard:



Wikipedia

Knight's Tour

- In contrast to the general case, there exist efficient algorithms for constructing knight's tours on an arbitrary $n \times n$ board.

Knight's Tour

- In contrast to the general case, there exist efficient algorithms for constructing knight's tours on an arbitrary $n \times n$ board.
- Good criteria whether such a tour exists are also available.

Applications of Hamiltonian Paths

- Besides toy examples like knight's tour, there are serious applications of Hamiltonian paths.

Applications of Hamiltonian Paths

- Besides toy examples like knight's tour, there are serious applications of Hamiltonian paths.
- In genomics, Hamiltonian paths are used for reconstructing the genome from its fragments: vertices = fragments, edges connect overlapping fragments, each fragment should be used exactly once.

Applications of Hamiltonian Paths

- Besides toy examples like knight's tour, there are serious applications of Hamiltonian paths.
- In genomics, Hamiltonian paths are used for reconstructing the genome from its fragments: vertices = fragments, edges connect overlapping fragments, each fragment should be used exactly once.
- In electronic circuit design, they are used for implementing efficient power gating.

Applications of Hamiltonian Paths

- Besides toy examples like knight's tour, there are serious applications of Hamiltonian paths.
- In genomics, Hamiltonian paths are used for reconstructing the genome from its fragments: vertices = fragments, edges connect overlapping fragments, each fragment should be used exactly once.
- In electronic circuit design, they are used for implementing efficient power gating.
- In computer graphics, Hamiltonian paths help in compact representation of data.

Outline

Königsberg Bridges. Euler Cycles

Constructing a Euler Cycle

Hamiltonian Paths

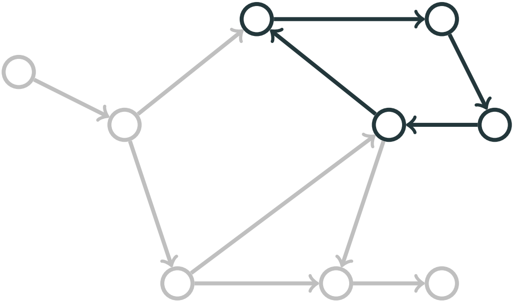
Acyclic Directed Graphs. Topological Sorting

Cycles in Directed Graphs

In directed graphs, cycles should also be directed!

Cycles in Directed Graphs

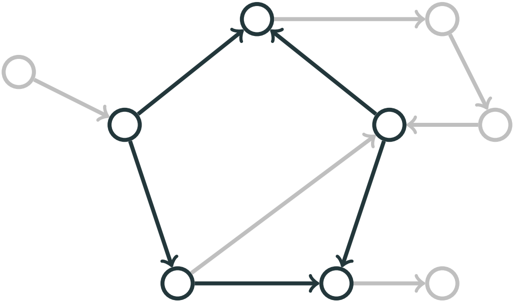
In directed graphs, cycles should also be directed!



this is a directed cycle

Cycles in Directed Graphs

In directed graphs, cycles should also be directed!



this is not a directed cycle

Directed Acyclic Graphs

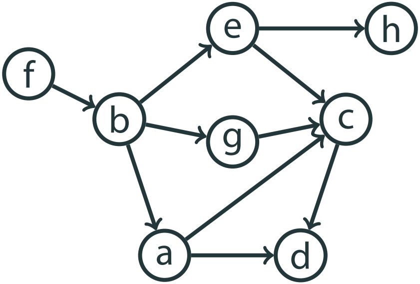
- A directed graph without directed cycles is called *acyclic.*

Directed Acyclic Graphs

- A directed graph without directed cycles is called *acyclic.*
- Directed acyclic graphs are called DAGs.

Directed Acyclic Graphs

- A directed graph without directed cycles is called *acyclic.*
- Directed acyclic graphs are called DAGs.
- Example:



Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.
- Examples:

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.
- Examples:
    - automated software installers;

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.
- Examples:
  - automated software installers;
  - software build scripts (like `make`);

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.
- Examples:
  - automated software installers;
  - software build scripts (like `make`);
  - job scheduling;

Application: Dependencies of Actions

- Let vertices represent *actions* to be performed.
- There is an edge from A to B if action B cannot be performed before A.
- Examples:
  - automated software installers;
  - software build scripts (like `make`);
  - job scheduling;
  - ...

Topological Sorting

- In a DAG, vertices can be enumerated in such a way that all edges go forward.

Topological Sorting

- In a DAG, vertices can be enumerated in such a way that all edges go forward.
- That is, if there is an edge from vertex i to vertex j, then $j > i$.

Topological Sorting

- In a DAG, vertices can be enumerated in such a way that all edges go forward.
- That is, if there is an edge from vertex i to vertex j, then $j > i$.
- This is called *topological sorting* of a DAG.

Topological Sorting

- In a DAG, vertices can be enumerated in such a way that all edges go forward.
- That is, if there is an edge from vertex i to vertex j, then $j > i$.
- This is called *topological sorting* of a DAG.
- For graphs with (directed) cycles topological sorting is impossible.

Topological Sorting

- In a DAG, vertices can be enumerated in such a way that all edges go forward.
- That is, if there is an edge from vertex i to vertex j, then $j > i$.
- This is called *topological sorting* of a DAG.
- For graphs with (directed) cycles topological sorting is impossible.
- In dependency graphs, topological sorting represents *correct execution order* of actions.

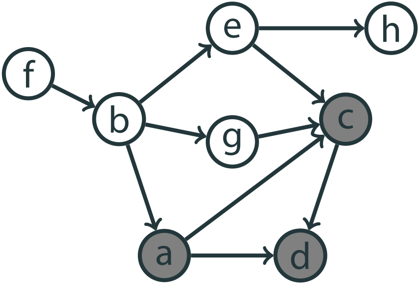
Topological Sorting Using DFS

- We prove existence of topological sorting of an arbitary DAG by presenting an algorithm which finds it.

Topological Sorting Using DFS

- We prove existence of topological sorting of an arbitary DAG by presenting an algorithm which finds it.
- This algorithm is based on depth-first search.

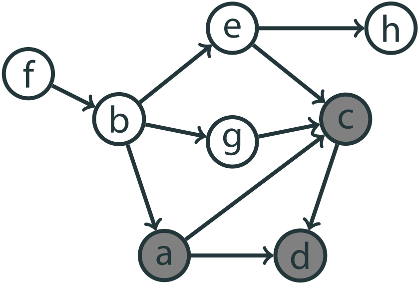
Topological Sorting Using DFS



a — c — d

Suppose we have already correctly sorted some of the vertices...

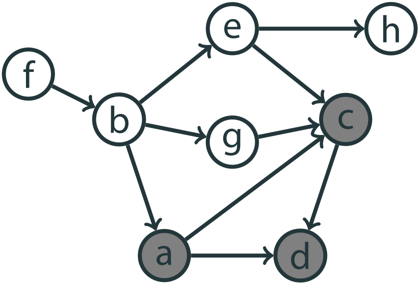
Topological Sorting Using DFS



a — c — d

...and wish to append a new one *to the beginning* of the list.

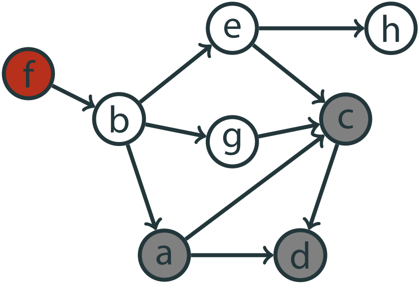
Topological Sorting Using DFS



? — a — c — d

...and wish to append a new one *to the beginning* of the list.

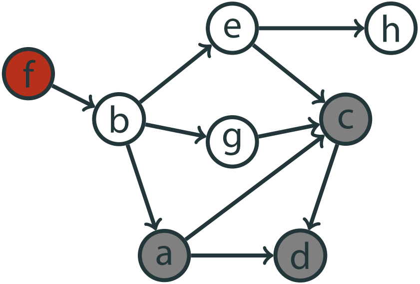
Topological Sorting Using DFS



? — a — c — d

We pick a random vertex...

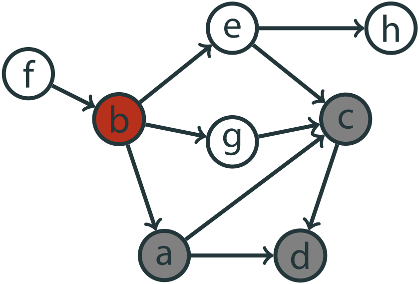
Topological Sorting Using DFS



? — a — c — d

...and follow arrows until a vertex from which no edge leads to a vertex not previously sorted.

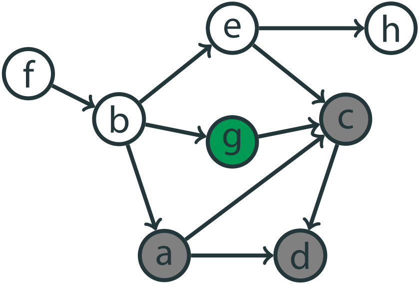
Topological Sorting Using DFS



? — a — c — d

...and follow arrows until a vertex from which no edge leads to a vertex not previously sorted.

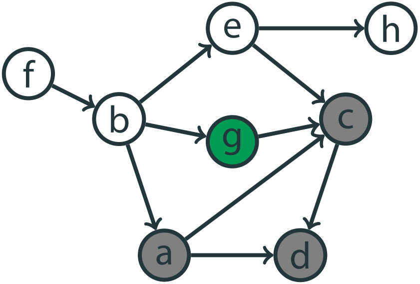
Topological Sorting Using DFS



? — a — c — d

...and follow arrows until a vertex from which no edge leads to a vertex not previously sorted.

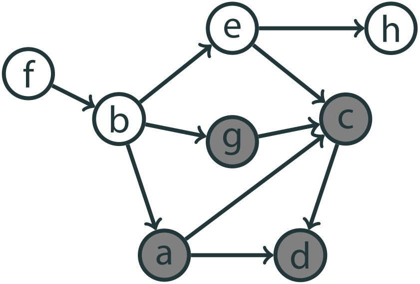
Topological Sorting Using DFS



? — a — c — d

This vertex is safe to add!

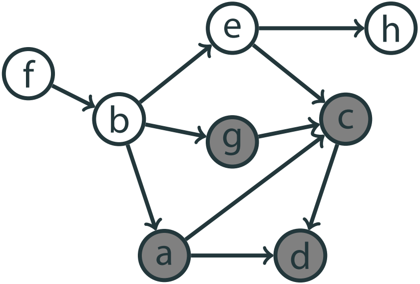
Topological Sorting Using DFS



g — a — c — d

This vertex is safe to add!

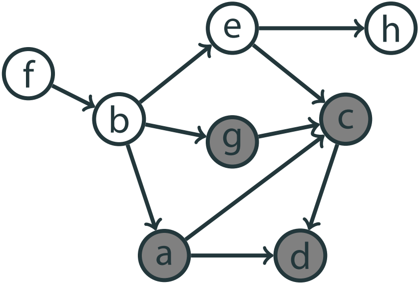
Topological Sorting Using DFS



g — a — c — d

Vertices a, c, d were added before, in the same manner...

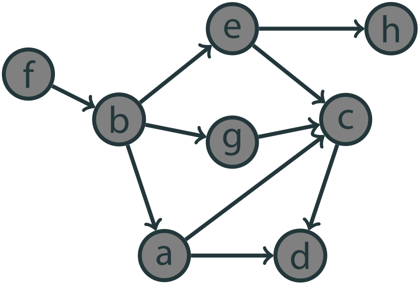
Topological Sorting Using DFS



g — a — c — d

... and we can continue this process, until all vertices are listed.

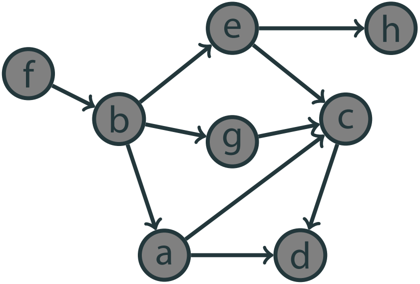
Topological Sorting Using DFS



f — b — e — h — g — a — c — d

... and we can continue this process, until all vertices are listed.

Topological Sorting Using DFS



f — b — e — h — g — a — c — d

This is the necessary topological sorting.