Topological Sorting

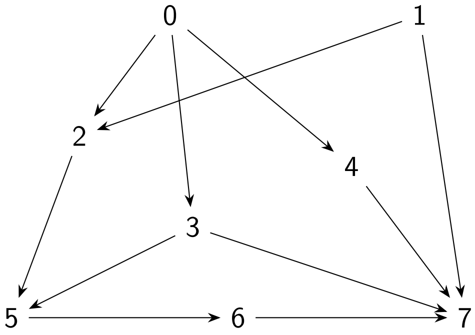
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Mathematics for Data Science 1
Week 11

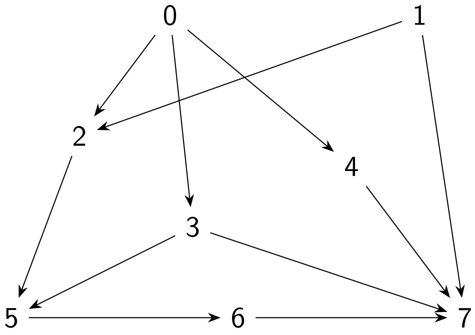
Directed Acyclic Graphs

- $G = (V, E)$, a directed graph without directed cycles



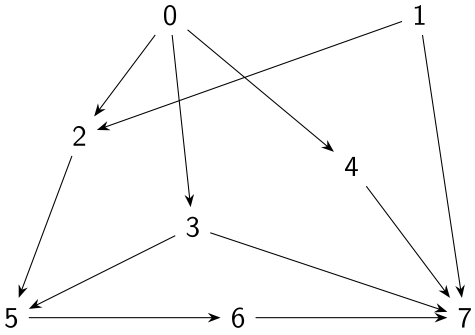
Directed Acyclic Graphs

- $G = (V, E)$, a directed graph without directed cycles
- Topological sorting
  - Enumerate $V = \{0, 1, \ldots, n-1\}$ such that for any $(i, j) \in E$, i appears before j



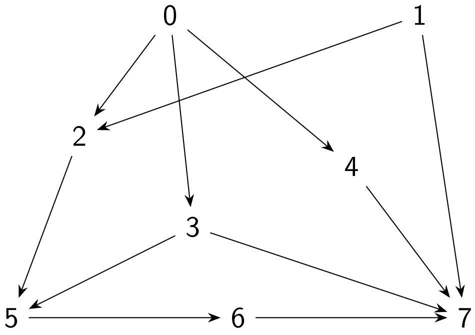
Directed Acyclic Graphs

- $G = (V, E)$, a directed graph without directed cycles
- Topological sorting
  - Enumerate $V = \{0, 1, \dots, n-1\}$ such that for any $(i, j) \in E$, i appears before j
- Represents a feasible schedule



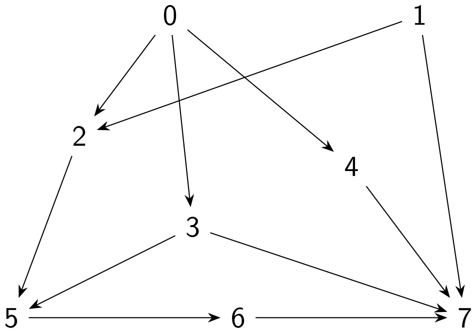
Topological Sort

- A graph with directed cycles cannot be sorted topologically



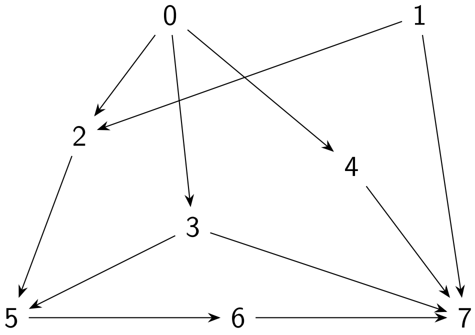
Topological Sort

- A graph with directed cycles cannot be sorted topologically
- Path $i \rightsquigarrow j$ means i must be listed before j



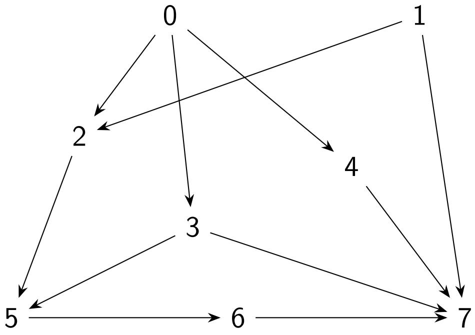
Topological Sort

- A graph with directed cycles cannot be sorted topologically
- Path $i \rightsquigarrow j$ means i must be listed before j
- Cycle $\Rightarrow$ vertices i, j such that there are paths $i \rightsquigarrow j$ and $j \rightsquigarrow i$



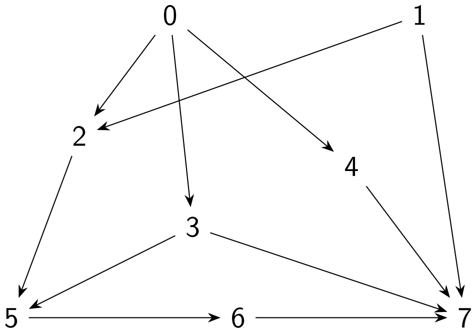
Topological Sort

- A graph with directed cycles cannot be sorted topologically
- Path $i \rightsquigarrow j$ means i must be listed before j
- Cycle $\Rightarrow$ vertices i, j such that there are paths $i \rightsquigarrow j$ and $j \rightsquigarrow i$
- i must appear before j, and j must appear before i, impossible!



Topological Sort

- A graph with directed cycles cannot be sorted topologically
- Path $i \rightsquigarrow j$ means i must be listed before j
- Cycle $\Rightarrow$ vertices i, j such that there are paths $i \rightsquigarrow j$ and $j \rightsquigarrow i$
- i must appear before j, and j must appear before i, impossible!

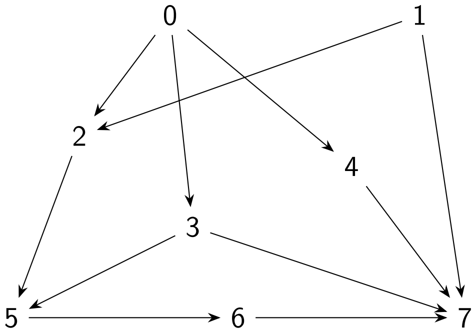


Claim

Every DAG can be topologically sorted

How to topologically sort a DAG?

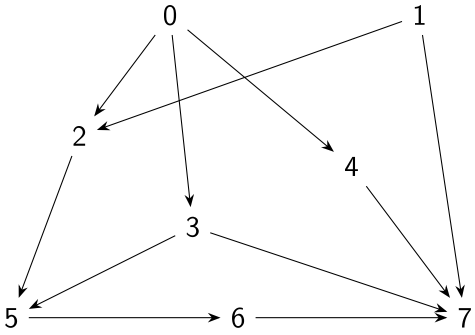
Strategy



How to topologically sort a DAG?

Strategy

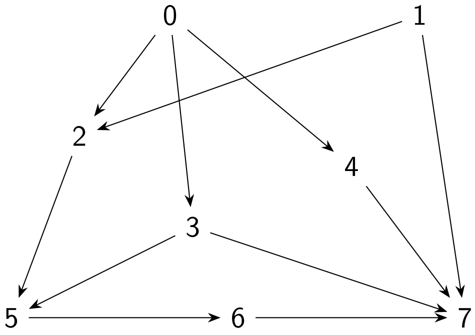
- First list vertices with no dependencies



How to topologically sort a DAG?

Strategy

- First list vertices with no dependencies
- As we proceed, list vertices whose dependencies have already been listed
- ...

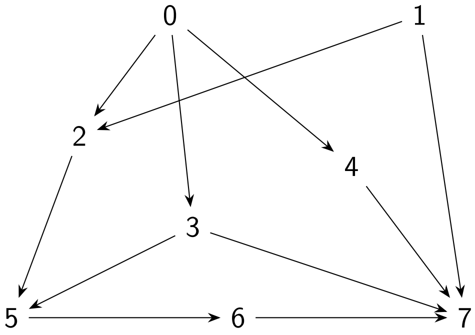


How to topologically sort a DAG?

Strategy

- First list vertices with no dependencies
- As we proceed, list vertices whose dependencies have already been listed
- ...

Questions



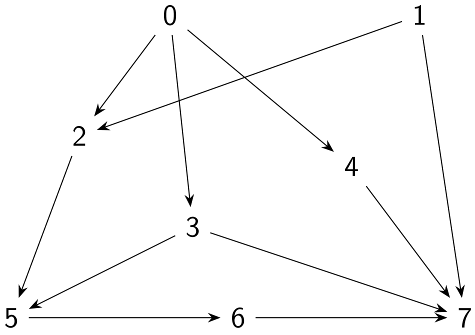
How to topologically sort a DAG?

Strategy

- First list vertices with no dependencies
- As we proceed, list vertices whose dependencies have already been listed
- ...

Questions

- Why will there be a starting vertex with no dependencies?



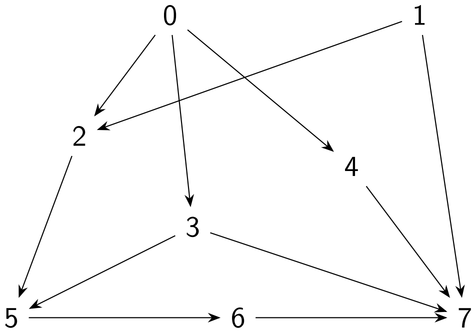
How to topologically sort a DAG?

Strategy

- First list vertices with no dependencies
- As we proceed, list vertices whose dependencies have already been listed
- . . .

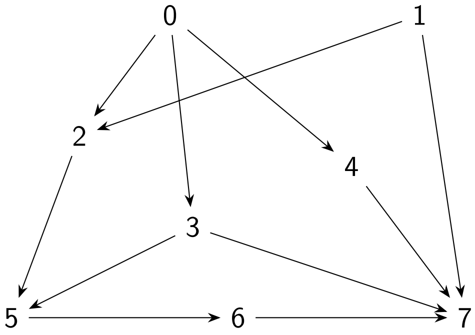
Questions

- Why will there be a starting vertex with no dependencies?
- How do we guarantee we can keep progressing with the listing?



Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

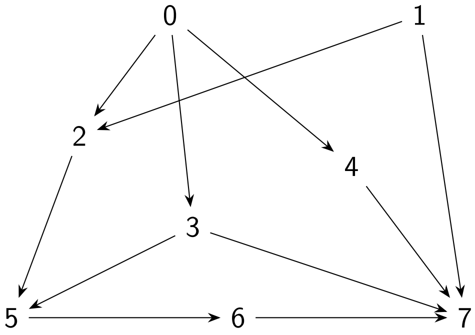


Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

Claim

Every DAG has a vertex with indegree 0



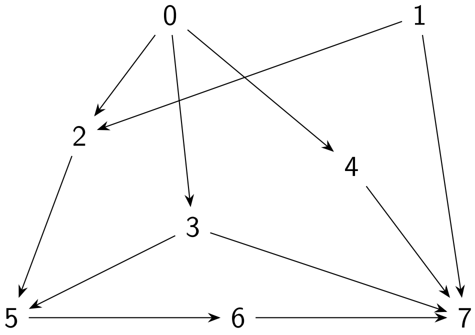
Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

Claim

Every DAG has a vertex with indegree 0

- Start with any vertex with $\text{indegree} > 0$



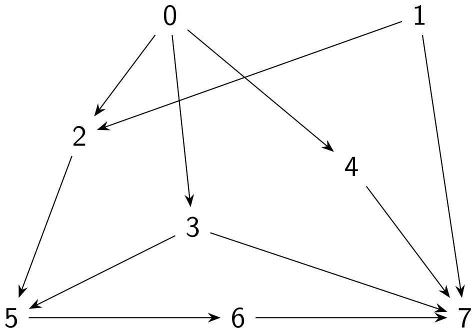
Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

Claim

Every DAG has a vertex with indegree 0

- Start with any vertex with $\text{indegree} > 0$
- Follow edge back to one of its predecessors



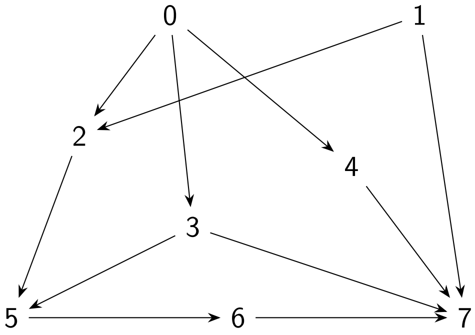
Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

Claim

Every DAG has a vertex with indegree 0

- Start with any vertex with $\text{indegree} > 0$
- Follow edge back to one of its predecessors
- Repeat so long as $\text{indegree} > 0$



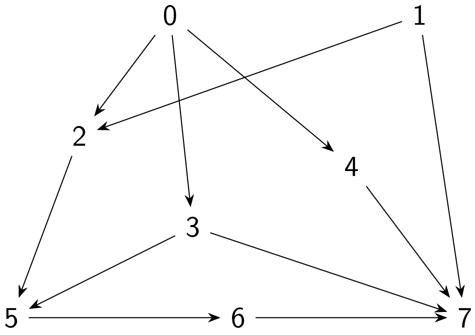
Algorithm for topological sort

- A vertex with no dependencies has no incoming edges, $\text{indegree}(v) = 0$

Claim

Every DAG has a vertex with indegree 0

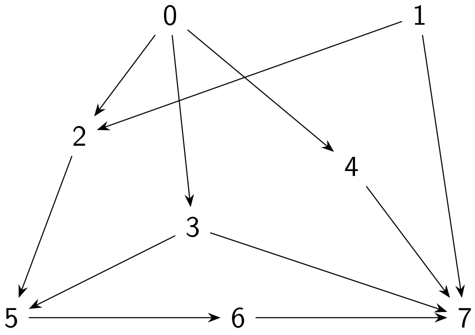
- Start with any vertex with $\text{indegree} > 0$
- Follow edge back to one of its predecessors
- Repeat so long as $\text{indegree} > 0$
- If we repeat n times, we must have a cycle, which is impossible in a DAG



Topological sort algorithm

Fact

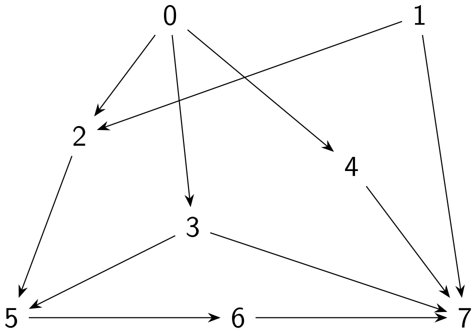
Every DAG has a vertex with indegree 0



Fact

Every DAG has a vertex with indegree 0

- List out a vertex j with indegree $= 0$

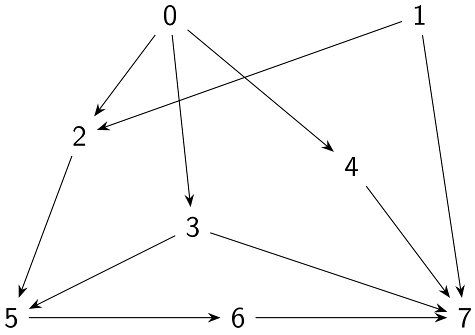


Topological sort algorithm

Fact

Every DAG has a vertex with indegree 0

- List out a vertex j with indegree $= 0$
- Delete j and all edges from j

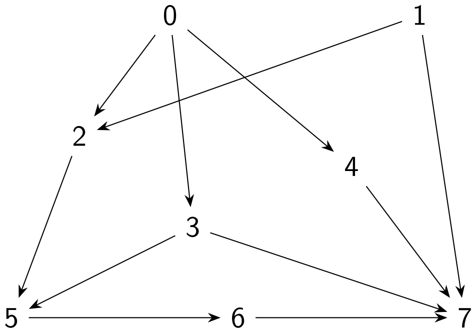


Topological sort algorithm

Fact

Every DAG has a vertex with indegree 0

- List out a vertex j with indegree $= 0$
- Delete j and all edges from j
- What remains is again a DAG!

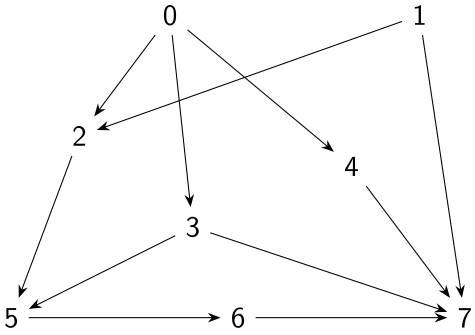


Topological sort algorithm

Fact

Every DAG has a vertex with indegree 0

- List out a vertex j with indegree $= 0$
- Delete j and all edges from j
- What remains is again a DAG!
- Can find another vertex with indegree $= 0$ to list and eliminate

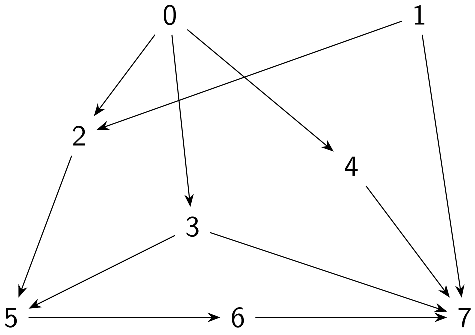


Topological sort algorithm

Fact

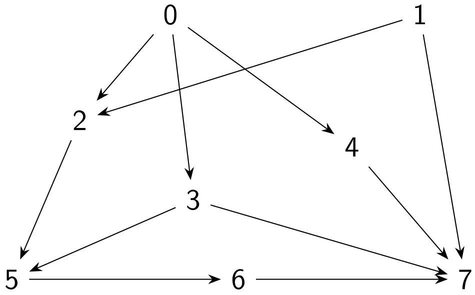
Every DAG has a vertex with indegree 0

- List out a vertex j with indegree $= 0$
- Delete j and all edges from j
- What remains is again a DAG!
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed



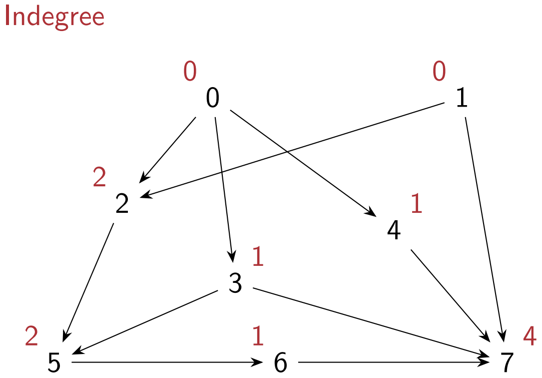
Topological sort algorithm

- Compute indegree of each vertex



Topological sort algorithm

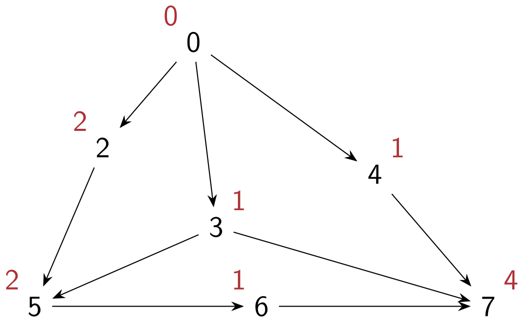
- Compute indegree of each vertex
  - Scan each column of the adjacency matrix



Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG

Indegree



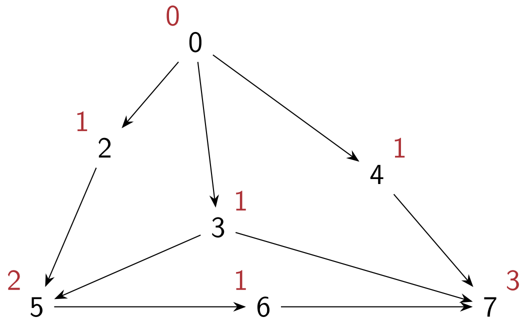
Topologically sorted sequence

1,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees

Indegree



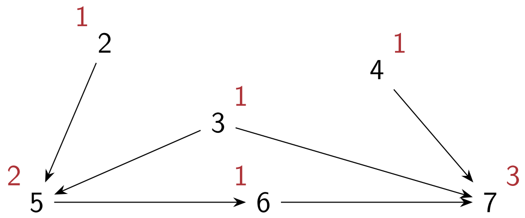
Topologically sorted sequence

1,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate

Indegree



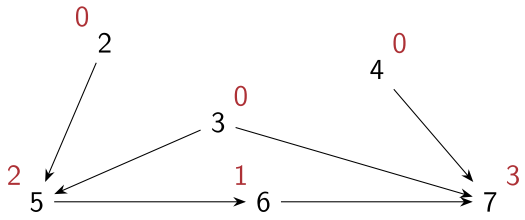
Topologically sorted sequence

1, 0,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate

Indegree



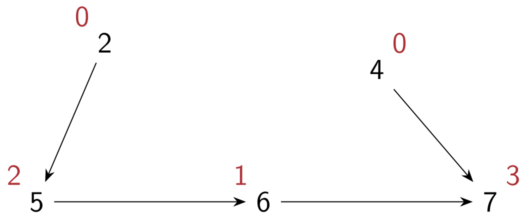
Topologically sorted sequence

1, 0,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



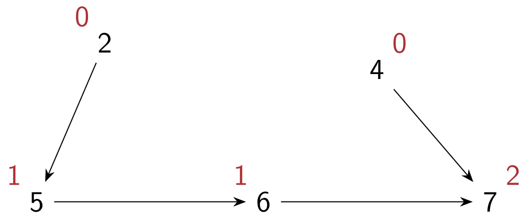
Topologically sorted sequence

1, 0, 3,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



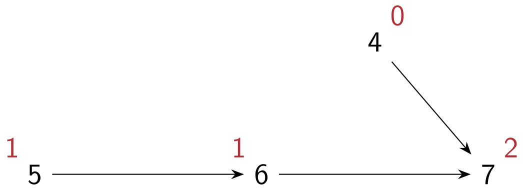
Topologically sorted sequence

1, 0, 3,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



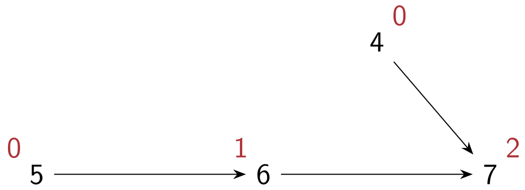
Topologically sorted sequence

1, 0, 3, 2,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



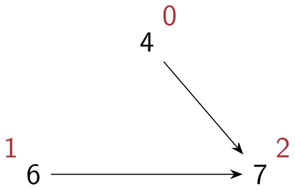
Topologically sorted sequence

1, 0, 3, 2,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



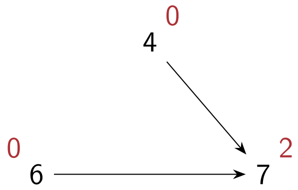
Topologically sorted sequence

1, 0, 3, 2, 5,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



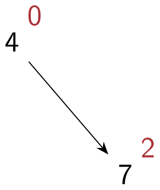
Topologically sorted sequence

1, 0, 3, 2, 5,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



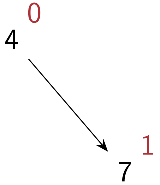
Topologically sorted sequence

1, 0, 3, 2, 5, 6,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree



Topologically sorted sequence

1, 0, 3, 2, 5, 6,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree

7 1

Topologically sorted sequence

1, 0, 3, 2, 5, 6, 4,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

Indegree

7 0

Topologically sorted sequence

1, 0, 3, 2, 5, 6, 4,

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

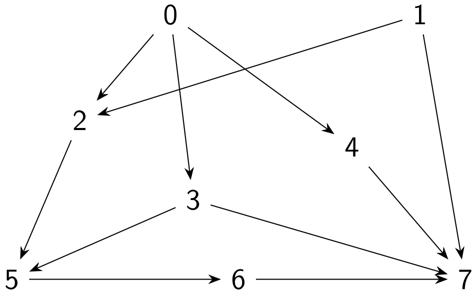
Indegree

Topologically sorted sequence

1, 0, 3, 2, 5, 6, 4, 7

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed

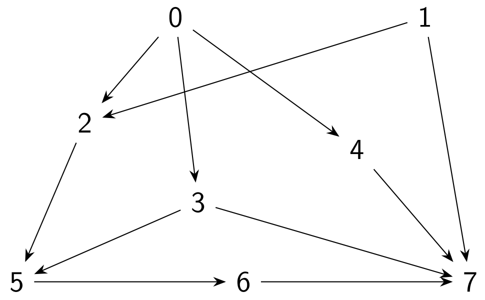


Topologically sorted sequence

1, 0, 3, 2, 5, 6, 4, 7

Topological sort algorithm

- Compute indegree of each vertex
  - Scan each column of the adjacency matrix
- List a vertex with indegree 0 and remove it from the DAG
- Update indegrees
- Can find another vertex with indegree $= 0$ to list and eliminate
- Repeat till all vertices are listed
- Using adjacency lists?
  - Scan each list $i \rightarrow [j_1, j_2, \ldots, j_k]$
  - Increment indegree(j_ℓ) for each j_ℓ



Topologically sorted sequence

1, 0, 3, 2, 5, 6, 4, 7

Summary

- Directed acyclic graphs are a natural way to represent dependencies
- Topological sort gives a feasible schedule that represents dependencies
  - At least one vertex with no dependencies, indegree 0
  - Eliminating such a vertex retains DAG structure
  - Repeat the process till all vertices are listed
- More than one topological sort is possible
  - Choice of which vertex with indegree 0 to list next