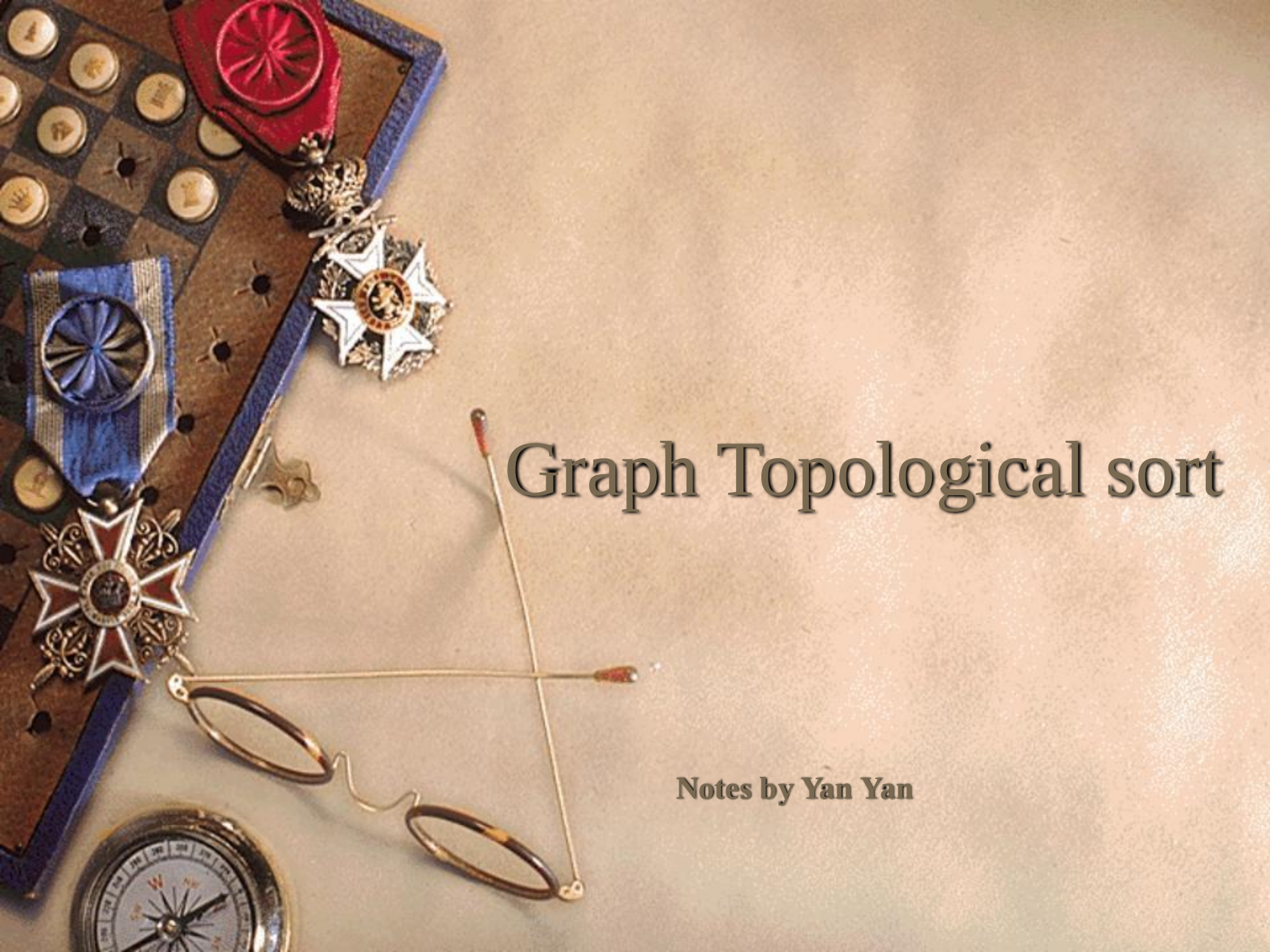


Graph Topological sort

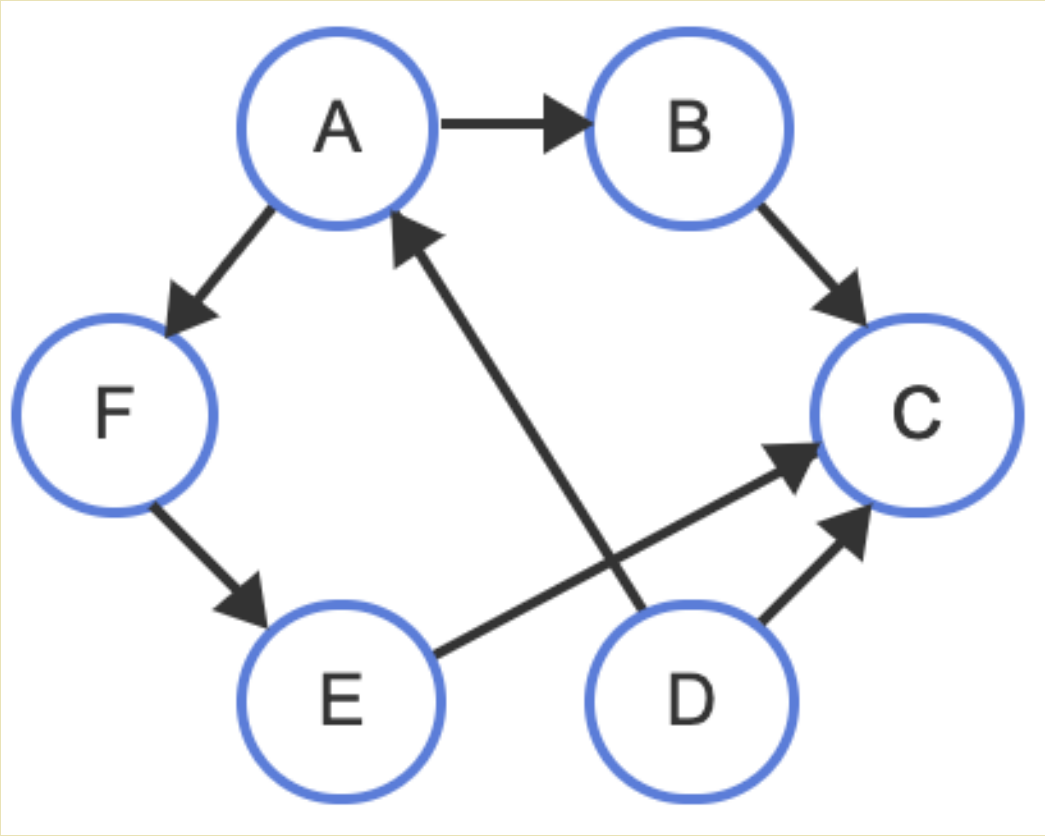
Notes by Yan Yan

Topological Sort

- Topological ordering is an operation on directed acyclic graphs (DAGs).
- A **topological ordering** is a list of the DAG's vertices such that for every edge from a vertex X to a vertex Y, X comes before Y in the list.
  - May not be unique
- A **topological sort** produces a list of topological ordering
- Applications: scheduling of tasks from the given dependencies among tasks.
  - a student can have an ordered list of courses to take.

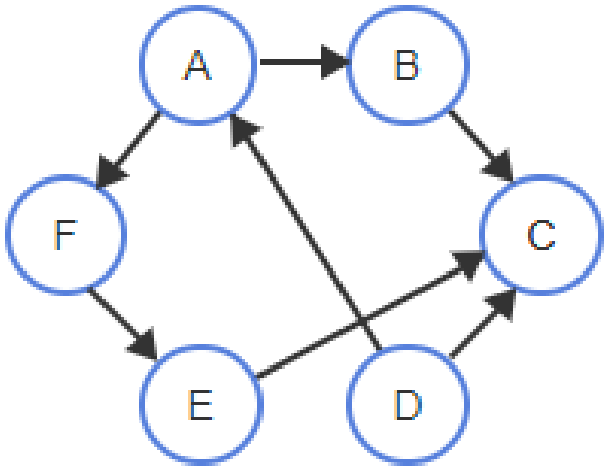
Topological Sort -- Example

- Is it a topological sort?
- C, D, A, F, B, E



Topological Sort -- Example

- Analysis of each edge in the graph determines if an ordering of vertices is a valid topological sort.



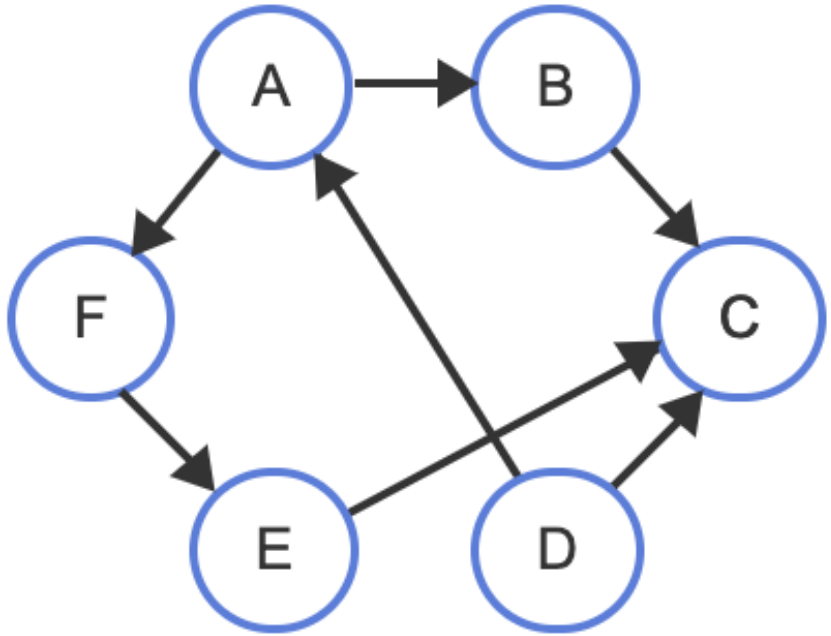
Proposed ordering: C, D, A, F, B, E (1, 2)

Edge (X to Y)	X before Y in ordering?
A to B	Yes
A to F	Yes
B to C	No
D to A	Yes
D to C	No
E to C	No
F to E	Yes

Topological Sort -- Example

- Is it a topological sort?
- D, A, F, E, B, C

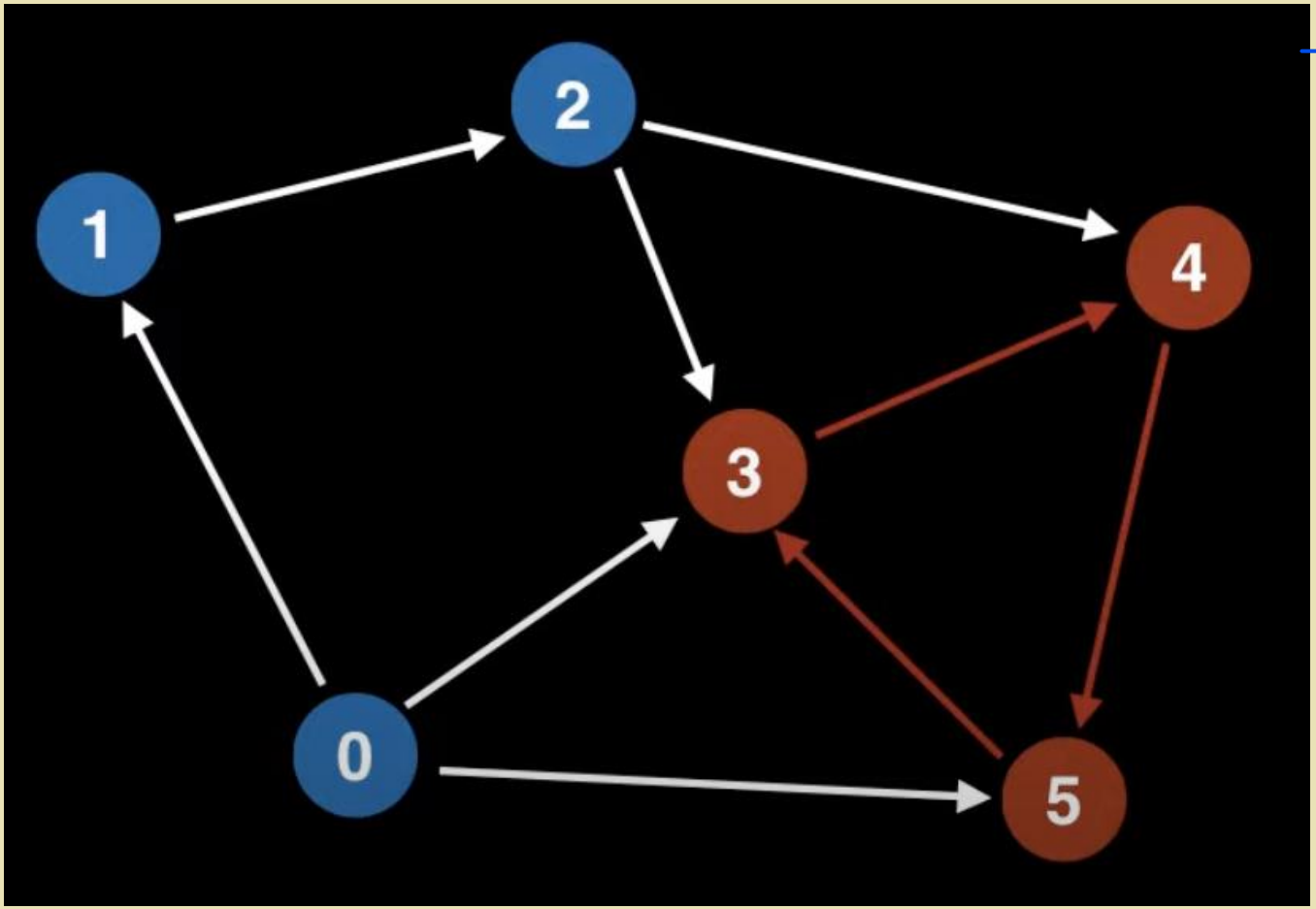
Yes



- If you check the edges of D and C, what can you find?

Topological Sort -- Example

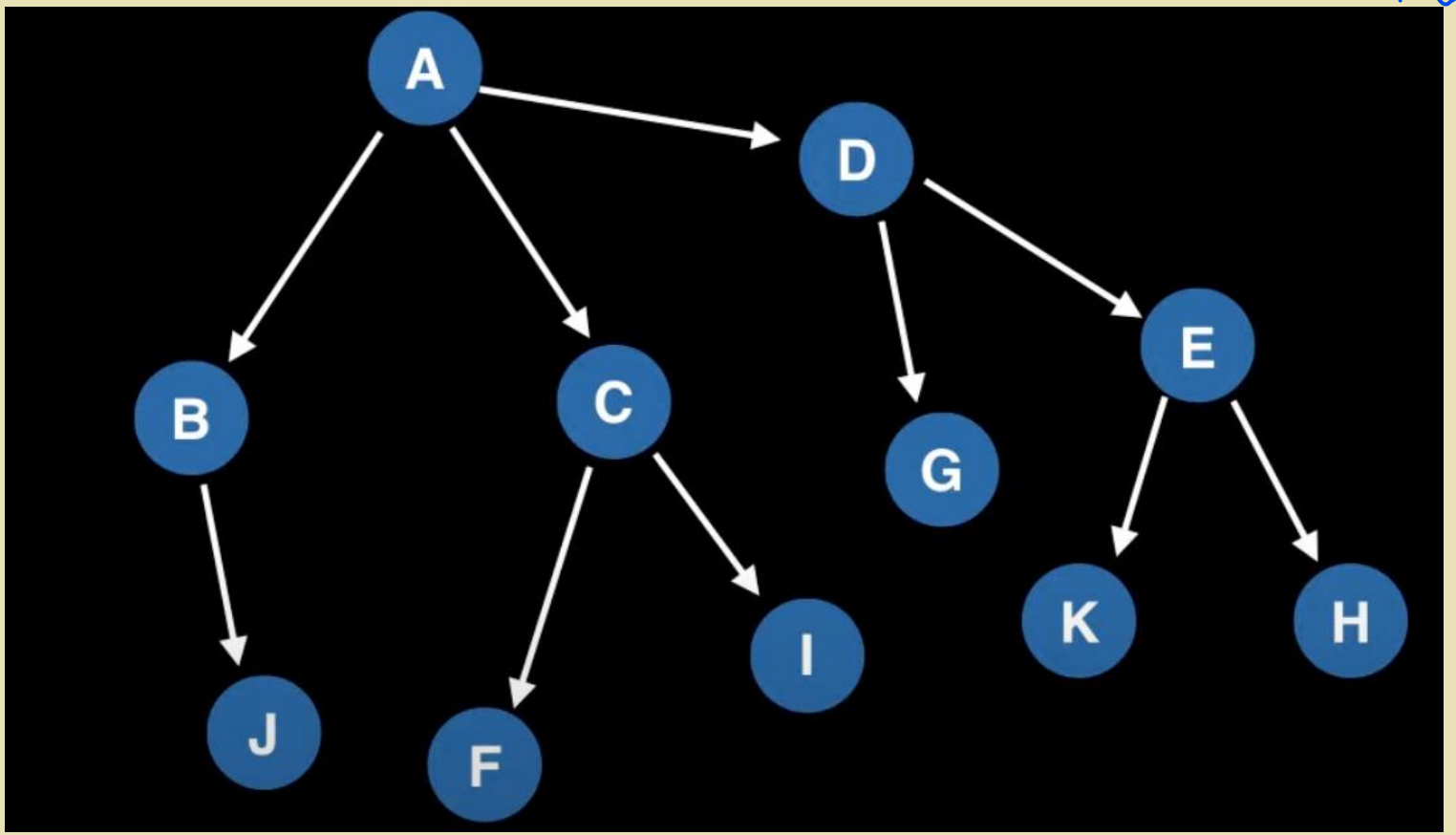
- Will cyclic graph have topological ordering? NO



https://www.youtube.com/watch?v=eL-KzMXSXXI

Topological Sort -- Example

- Will a (directed) tree have topological ordering? Yes

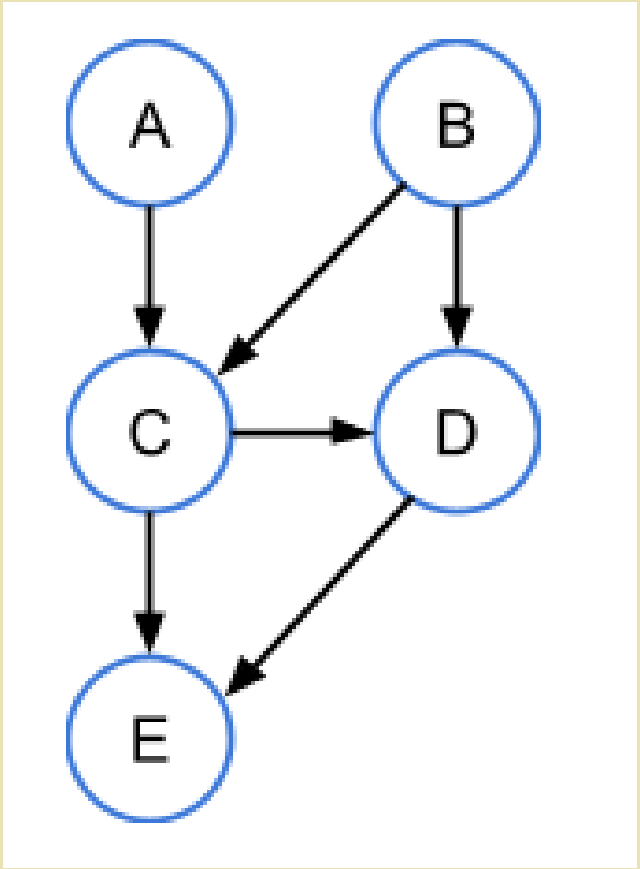


https://www.youtube.com/watch?v=eL-KzMXSXXI

Topological Sort -- Exercise

- Determine if each of the following is a topological sort for the given graph

1. A, B, C, D, E ✓
2. E, D, C, B, A ✓
3. D, E, A, B, C ✗
4. B, A, C, D, E ✓



- Check each edge!

Topological Sort Algorithm

- Can be done using **source removal**.
- A source is a vertex with no incoming edges.
- Each step, a source is identified. The source is removed from the graph along with all its outgoing edges. The vertex is then added at the end of the list.
- The process continues until all vertices are removed from the graph.

Topological Sort Algorithm

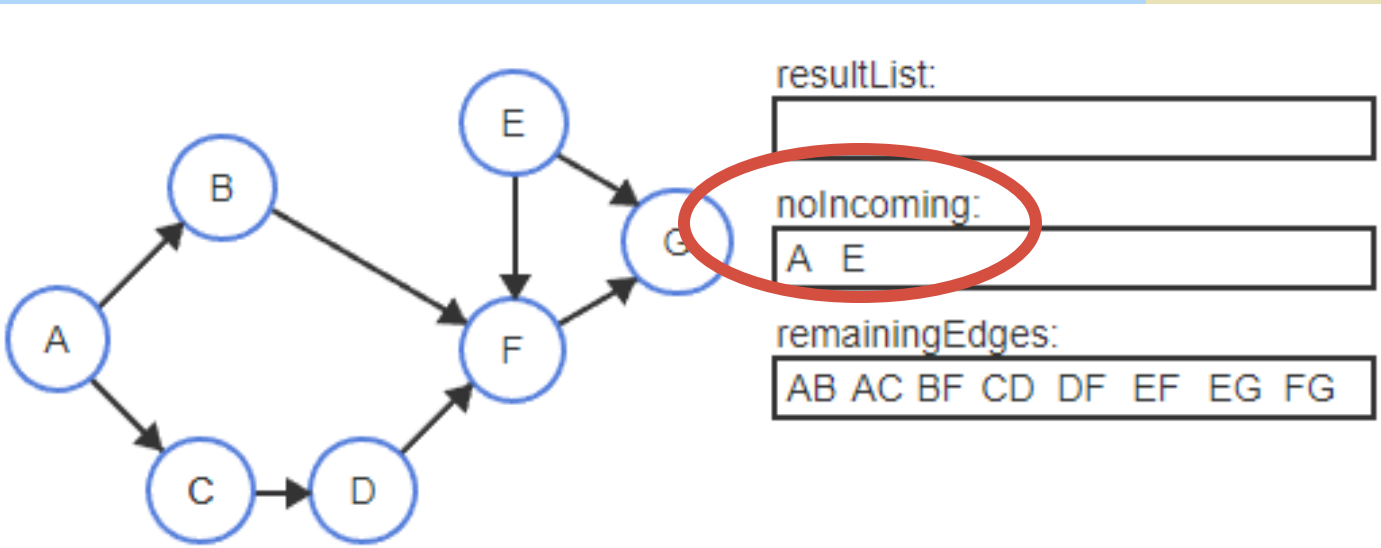
- The topological sort algorithm uses three lists:
  - a **results** list that will contain a topological sort of vertices: start as empty
  - a **no-incoming-edges** list of vertices with no incoming edges (source)
  - a remaining-edges list: start as all edges

Topological Sort Algorithm

- while the no-incoming-edges vertex list is not empty
  - a vertex is removed from the no-incoming-edges list and added to the result list.
  - a temporary list is built by removing all edges in the remaining-edges list that are outgoing from the removed vertex.
  - For each edge currentE in the temporary list, the number of edges in the remaining-edges list that are incoming to currentE's terminating vertex are counted.
  - If the incoming edge count is 0, then currentE's terminating vertex is added to the no-incoming-edges vertex list.

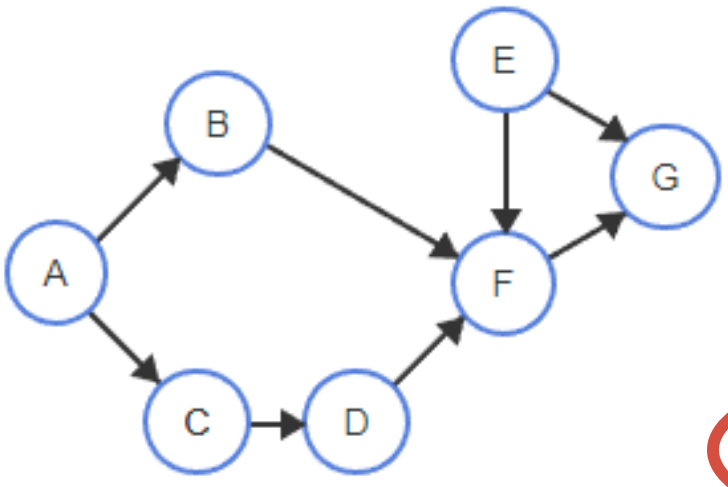
Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E

noIncoming:

A

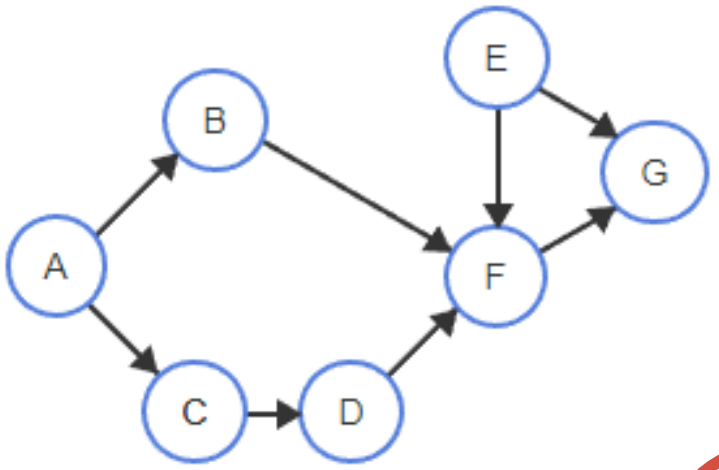
remainingEdges:

AB AC BF CD DF EF EG FG

currentV: E

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:
E

noIncoming:
A

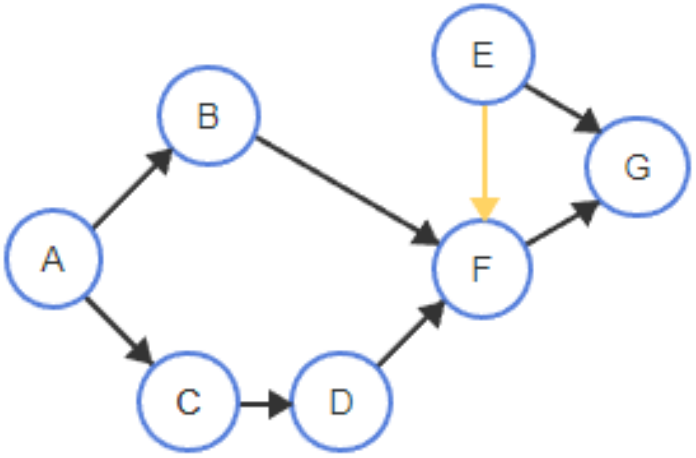
remainingEdges:
AB AC BF CD DF FG

currentV: E

outgoingEdges:
EF EG

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E

noIncoming:

A

remainingEdges:

AB AC BF CD DF FG

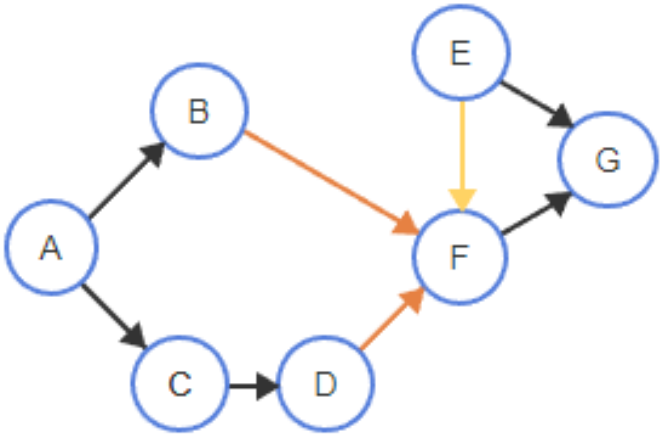
currentV: E

outgoingEdges:

EF EG

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E

noIncoming:

A

remainingEdges:

AB AC BF CD DF FG

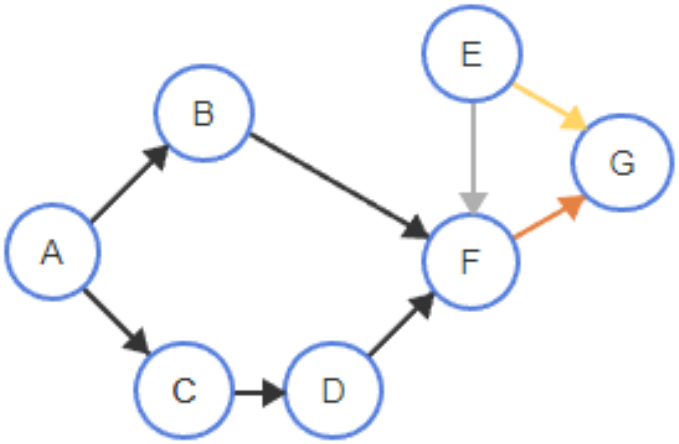
currentV: E inCount: 2

outgoingEdges:

EF EG

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E

noIncoming:

A

remainingEdges:

AB AC BF CD DF FG

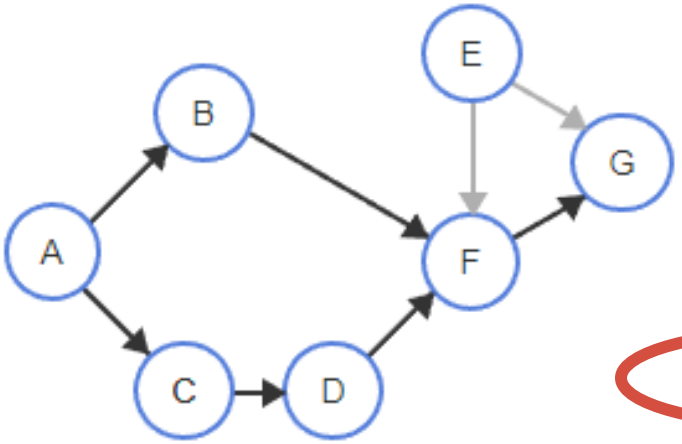
currentV: E inCount: 1

outgoingEdges:

EF EG

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A

noIncoming:

remainingEdges:

BF CD DF FG

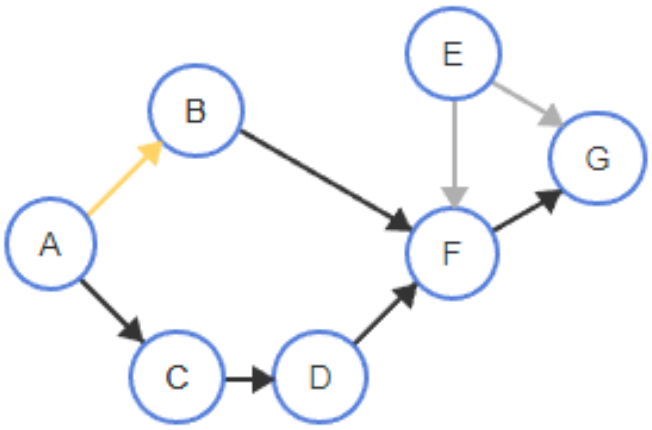
currentV: A

outgoingEdges:

AB AC

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A

noIncoming:

remainingEdges:

BF CD DF FG

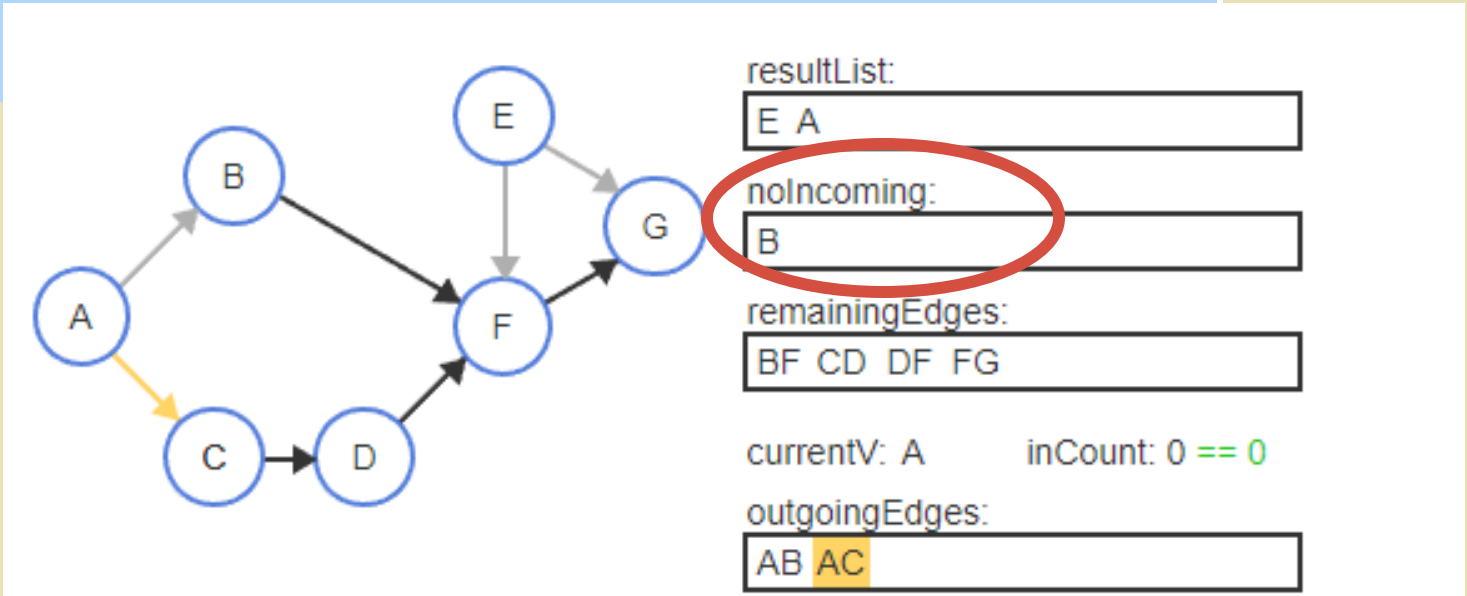
currentV: A inCount: 0 == 0

outgoingEdges:

AB AC

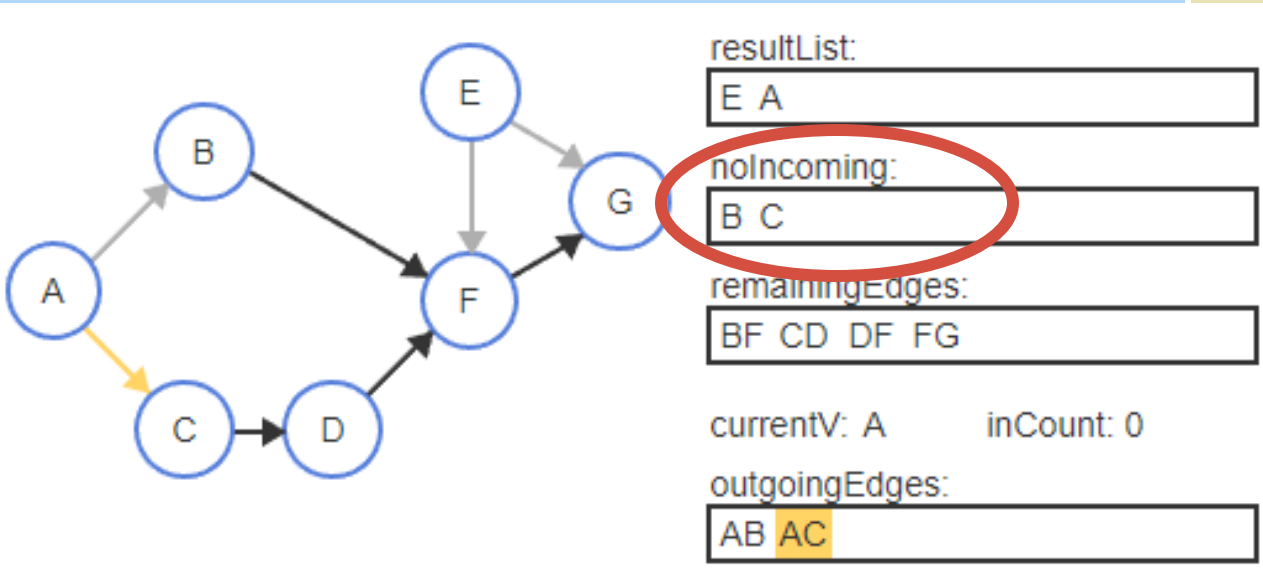
Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



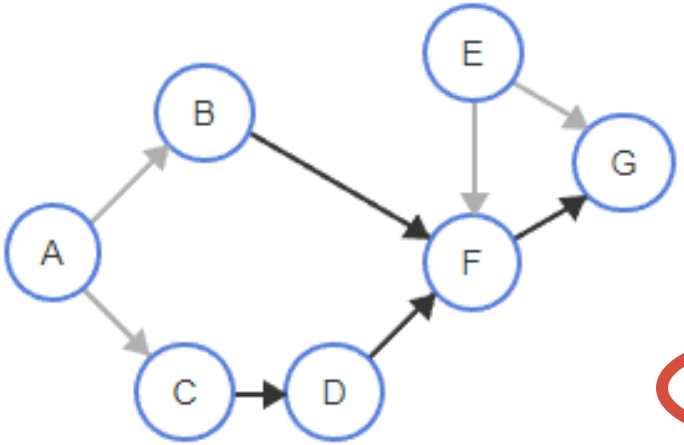
Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:
E A C

noIncoming:
B

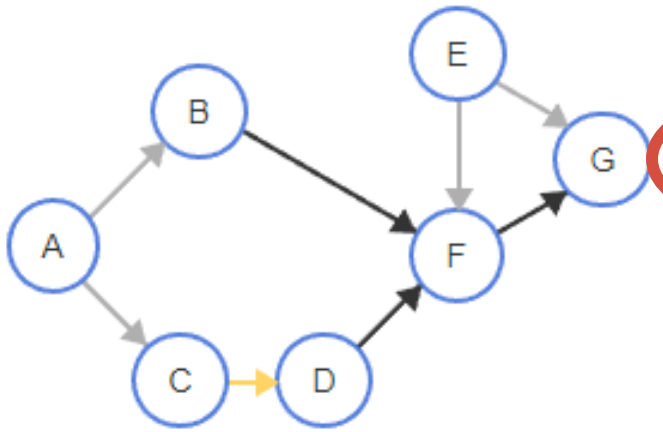
remainingEdges:
BF DF FG

currentV: C

outgoingEdges:
CD

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A C

noIncoming:

B D

remainingEdges:

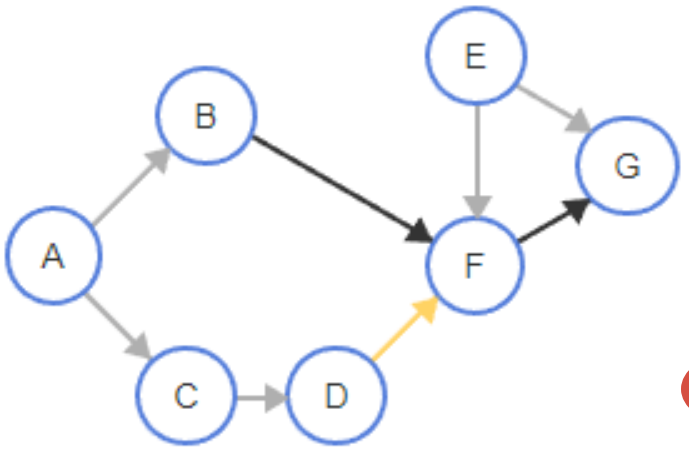
BF DF FG

currentV:

outgoingEdges:

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:
E A C D

noIncoming:
B

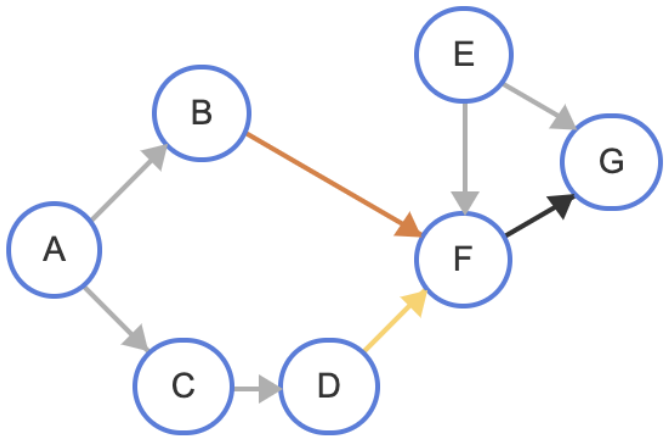
remainingEdges:
BF FG

currentV: D

outgoingEdges:
DF

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A C D

noIncoming:

B

remainingEdges:

BF FG

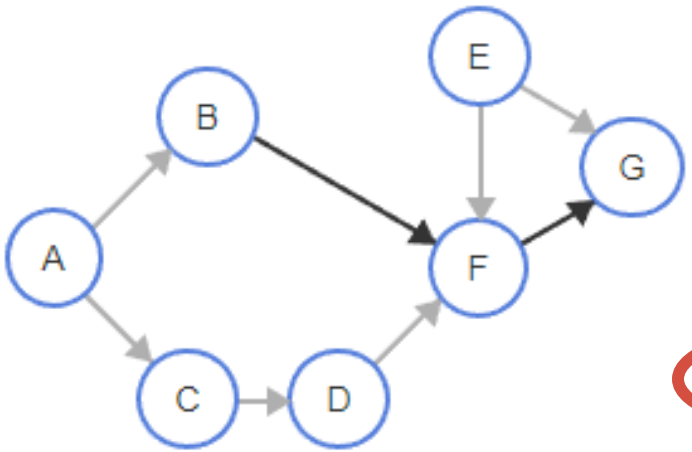
currentV: D inCount: 1 != 0

outgoingEdges:

DF

Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A C D B

noIncoming:

remainingEdges:

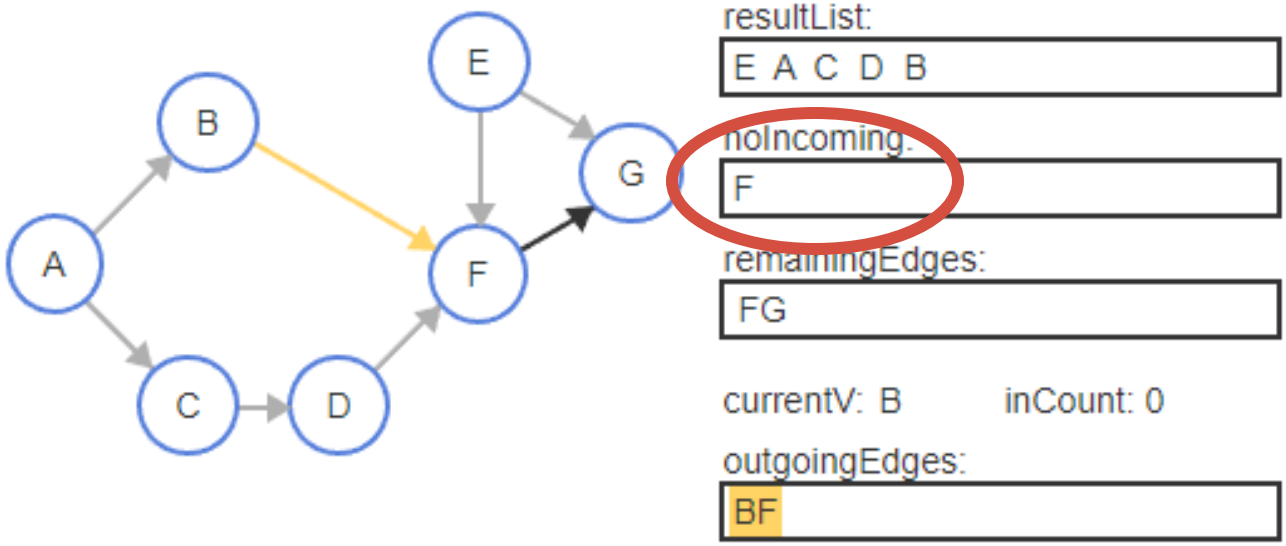
BF FG

currentV: B

outgoingEdges:

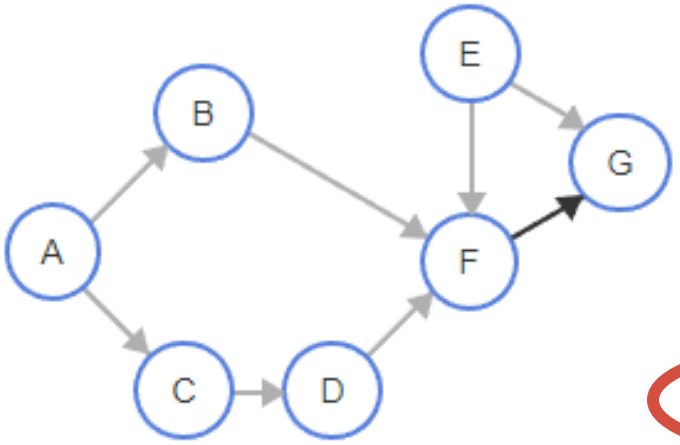
Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:
E A C D B F

noIncoming:

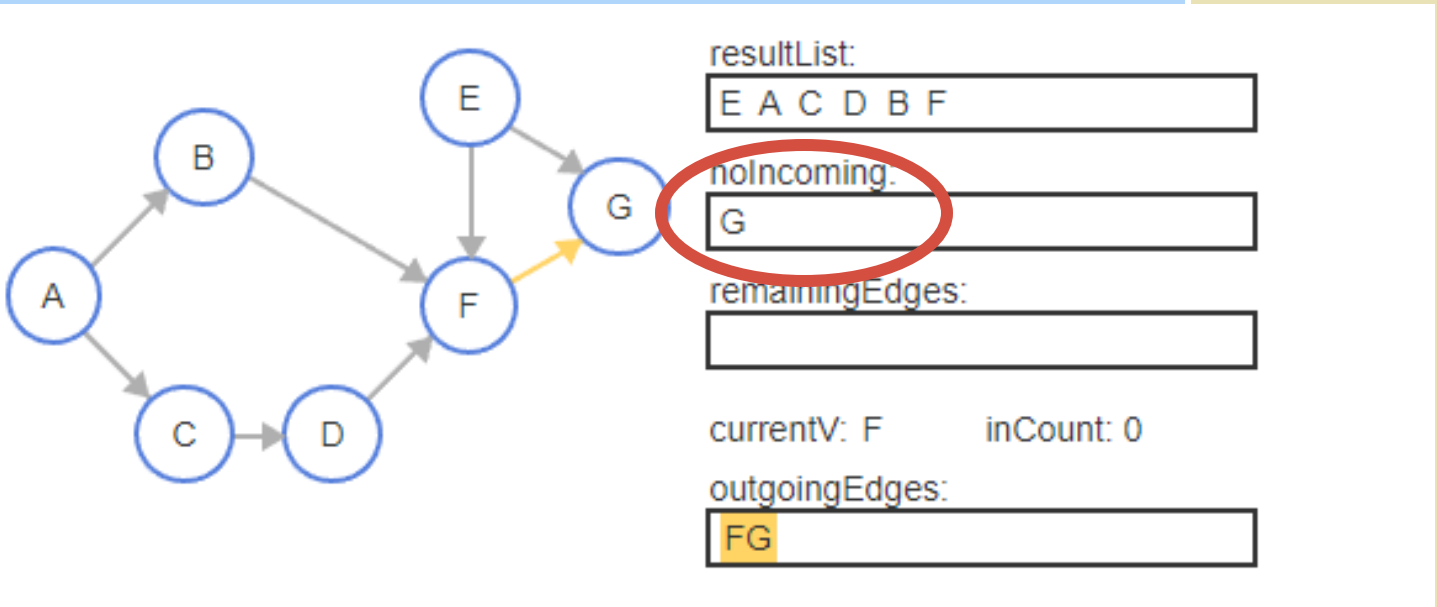
remainingEdges:

currentV: F

outgoingEdges:
FG

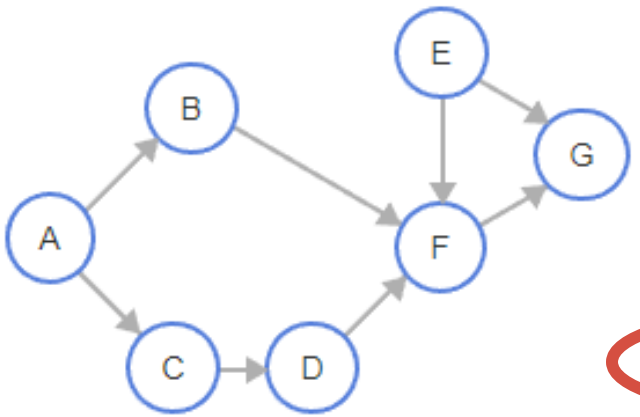
Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



Topological Sort Algorithm

```
GraphTopologicalSort(graph) {
   resultList = empty list of vertices
   noIncoming = list of all vertices with no incoming edges
   remainingEdges = list of all edges in the graph

   while (noIncoming is not empty) {
      currentV = remove any vertex from noIncoming
      Add currentV to resultList
      outgoingEdges = remove currentV's outgoing edges from remainingEdges
      for each edge currentE in outgoingEdges {
         inCount = GraphGetIncomingEdgeCount(remainingEdges, currentE→toVertex)
         if (inCount == 0)
            Add currentE→toVertex to noIncoming
      }
   }
   return resultList
}
```



resultList:

E A C D B F G

noIncoming:

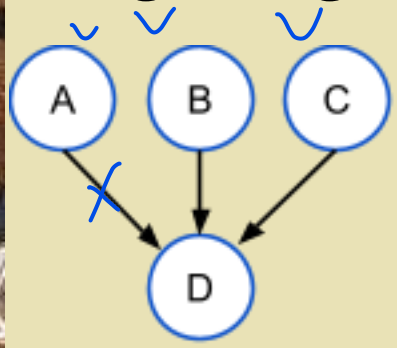
remainingEdges:

currentV: G

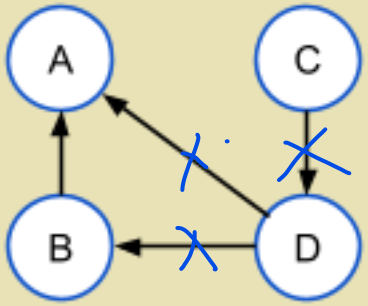
outgoingEdges:

Topological Sort -- Example

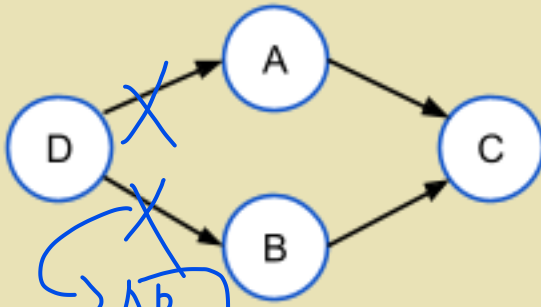
- Write down a valid topological sort for each of the given graph.



1



2



3

Algorithm efficiency

- The two vertex lists used in the topological sort algorithm will at most contain all the vertices in the graph. The remaining-edge list will at most contain all edges in the graph.
- For a graph with a set of vertices V and a set of edges E, the space complexity of topological sorting is O(|V|+|E|).
- If a graph implementation allows for retrieval of a vertex's incoming and outgoing edges in constant time, then the time complexity of topological sorting is also O(|V|+|E|).

That's about this lecture!