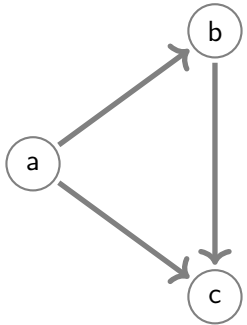
Sayem Shahad Soummo
Mahir Labib Dihan
Souvik Ghosh

TOPOLOGICAL SORT

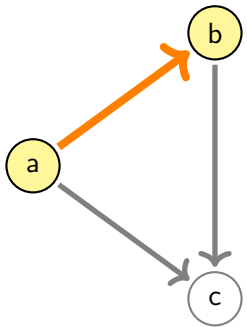
WHAT?

- For every edge in a DAG, the starting vertex is visited before ending vertex..

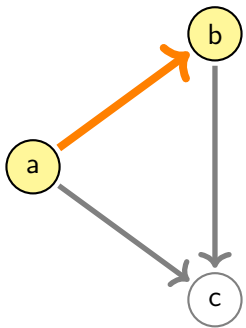
- For every edge in a DAG, the starting vertex is visited before ending vertex..



- For every edge in a DAG, the starting vertex is visited before ending vertex..

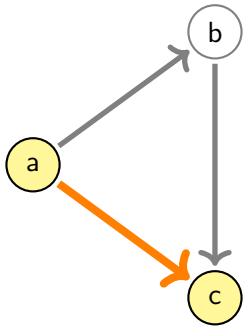


- For every edge in a DAG, the starting vertex is visited before ending vertex..

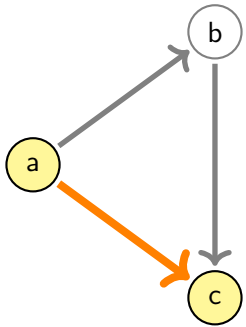


- a is visited before b

- For every edge in a DAG, the starting vertex is visited before ending vertex..

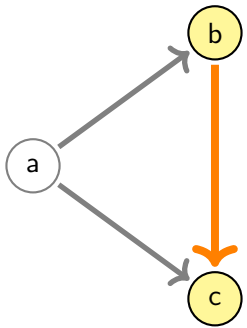


- For every edge in a DAG, the starting vertex is visited before ending vertex..

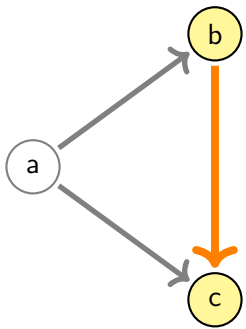


- a is visited before c

- For every edge in a DAG, the starting vertex is visited before ending vertex..



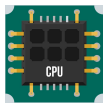
- For every edge in a DAG, the starting vertex is visited before ending vertex..

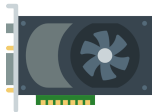


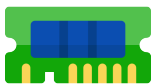
- b is visited before c

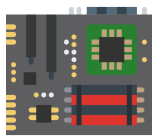
WHY?

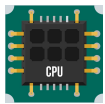
Let's assemble a PC!!!

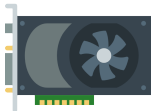

CPU

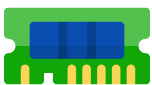
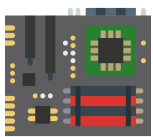




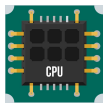


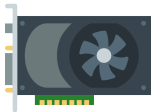


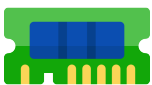
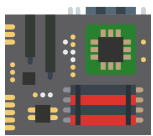


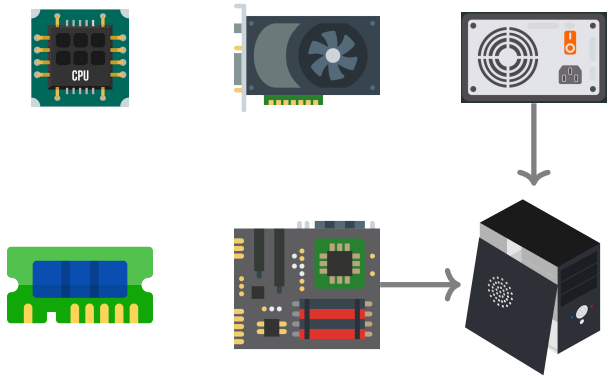

- Lets buy casing first



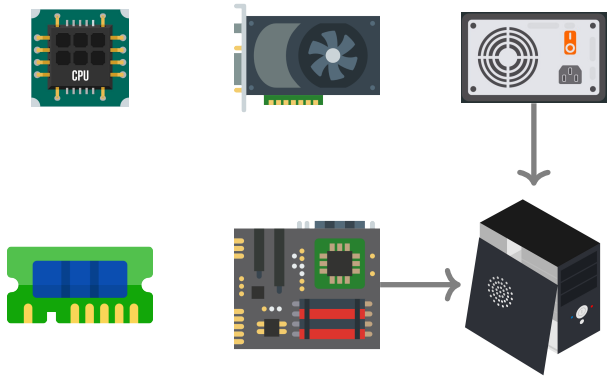


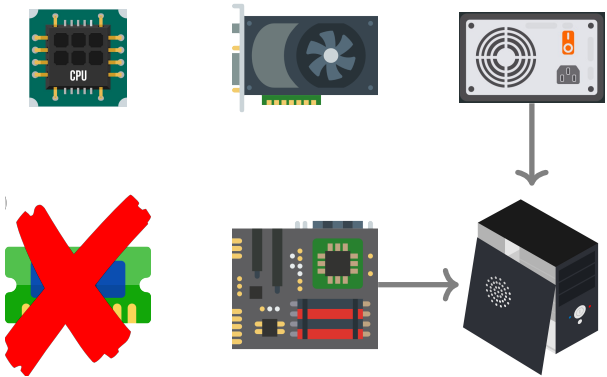

- But we can't!!



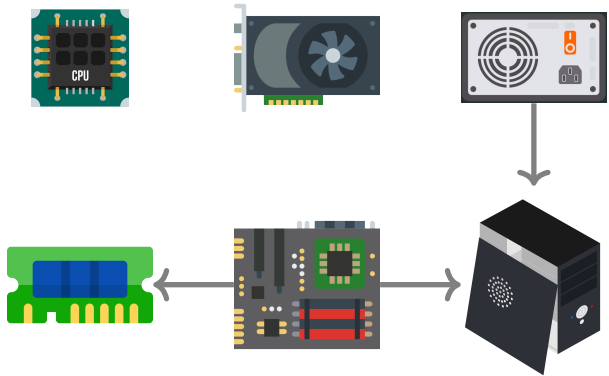
- We need motherboard and PSU model



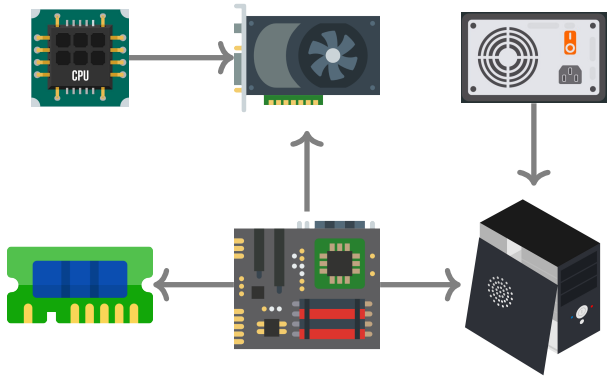
- Lets try to buy RAM

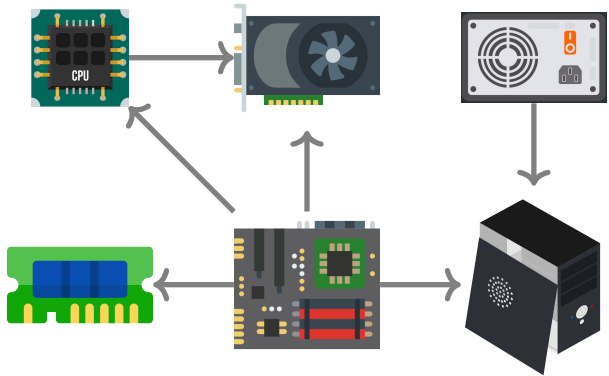


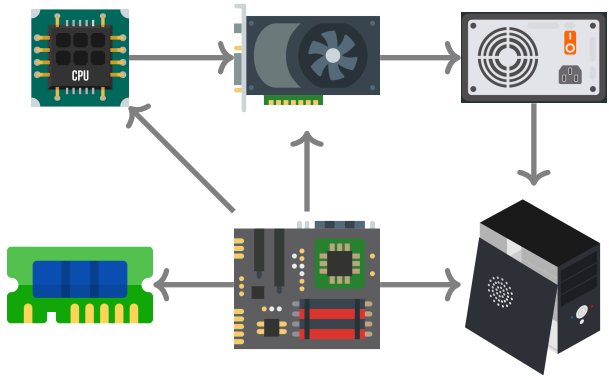
- Again!! We can't



- We need motherboard model


CPU

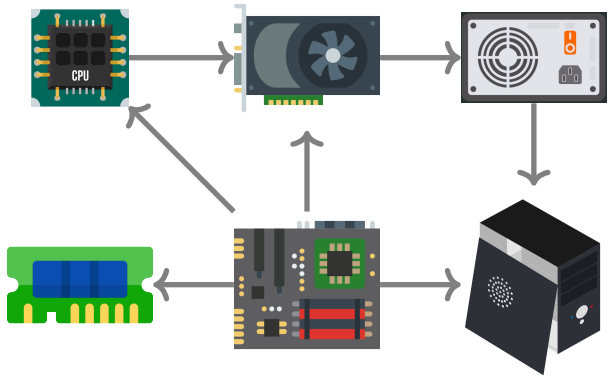

CPU

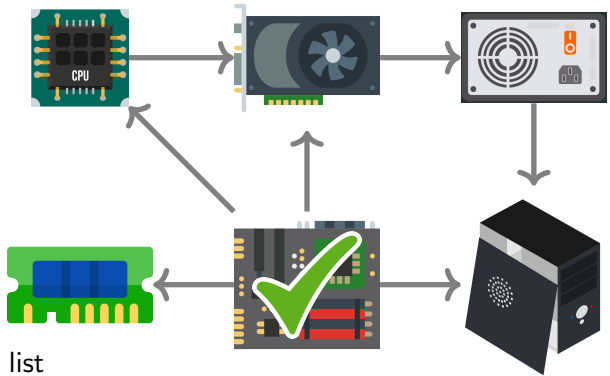

CPU

- We can't buy things in random order

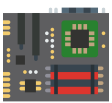
- We can't buy things in random order
- Maintain the topological Order!!!

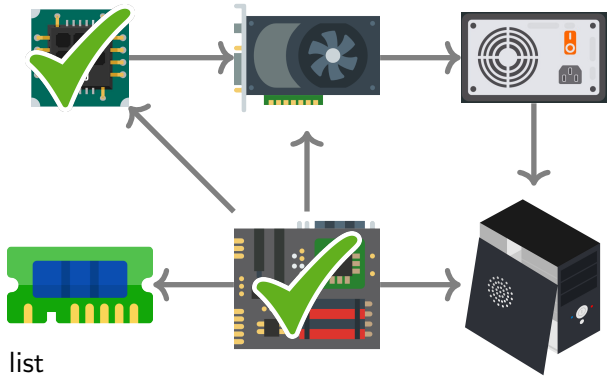
Let's get back to our example


CPU

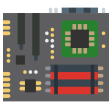


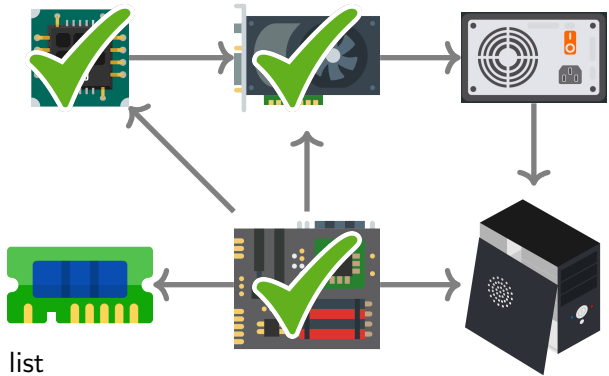
- Our shopping list



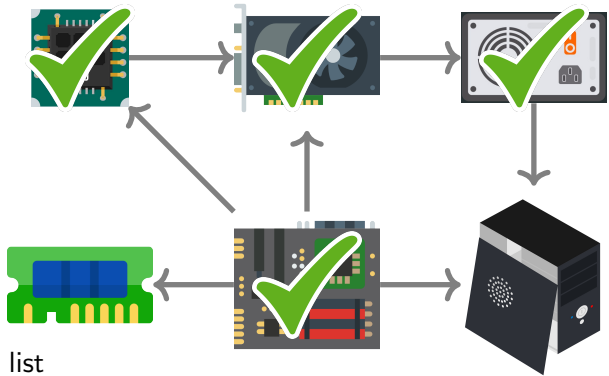


- Our shopping list

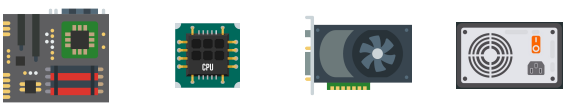


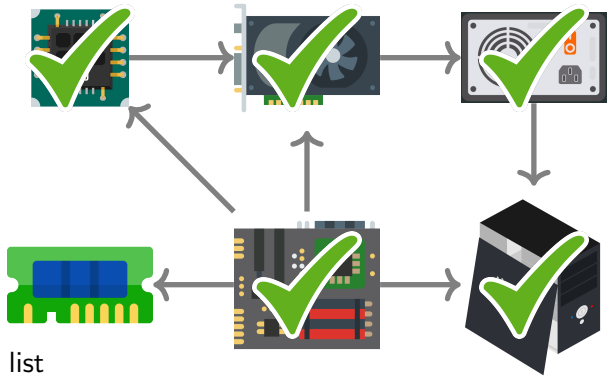


- Our shopping list

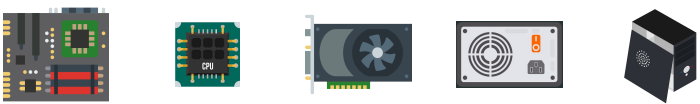


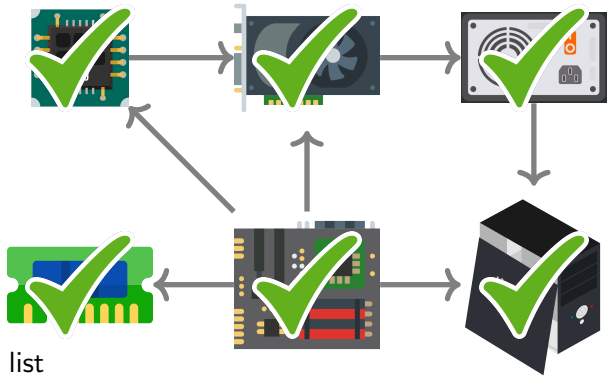
- Our shopping list





- Our shopping list

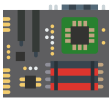


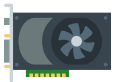


- Our shopping list

- There can exist multiple ordering

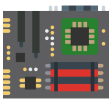
- There can exist multiple ordering

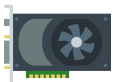




RAM can be bought right after motherboard

- There can exist multiple ordering





  RAM can be bought right after motherboard

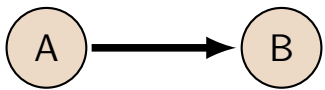
- Dependencies should be taken care beforehand
  In our example, we could not buy casing before motherboard and PSU

We have used topological sorting. But how it works!!!

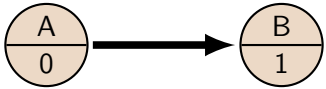
HOW?

- Let's say there are two nodes A and B.

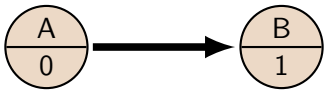
- Let's say there are two nodes A and B.
- And there is an edge from A to B, we can say that B is dependent on A



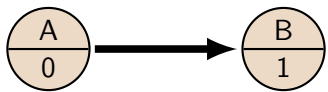
- Focus on indegree (Number of incoming edges) in each node



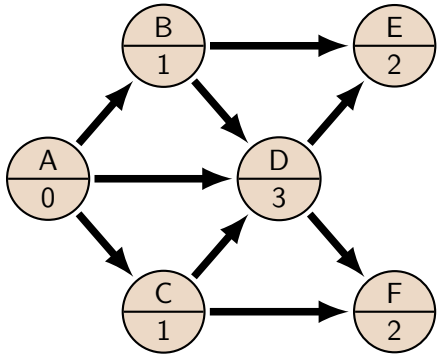
- Focus on indegree (Number of incoming edges) in each node
- Indegree represents dependency. B has 1, A has none.



- Focus on indegree (Number of incoming edges) in each node
- Indegree represents dependency. B has 1, A has none.
- A node can be visited only if it doesn't have any dependencies or we can say if the indegree is 0

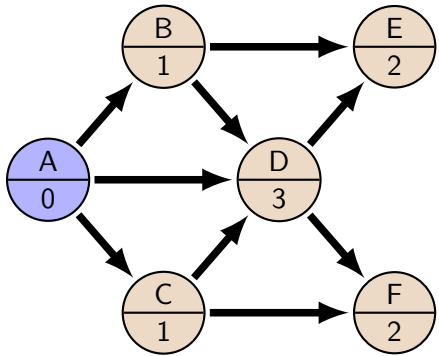


- Add more nodes to the graph.



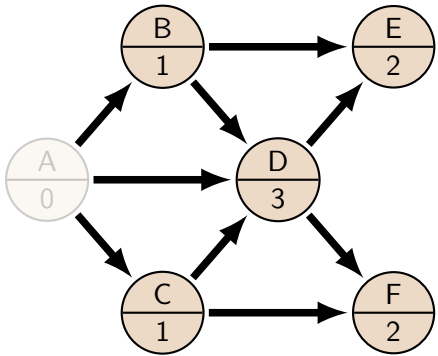
Topological
Order

- A is the only node with indegree = 0



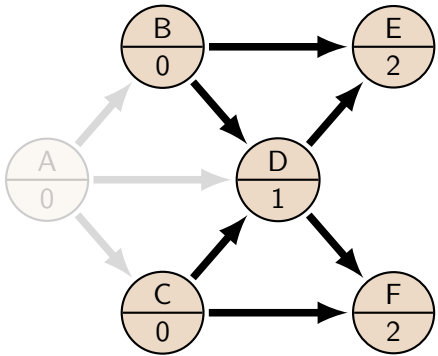
Topological
Order

- Add node A to the topological ordering and remove it from graph



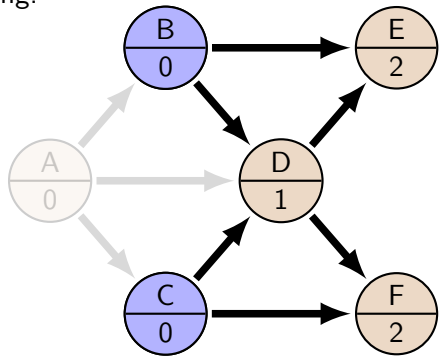
Topological Order

- Outgoing edges of node A also need to be removed which will decrease indegree of B,C,D by 1



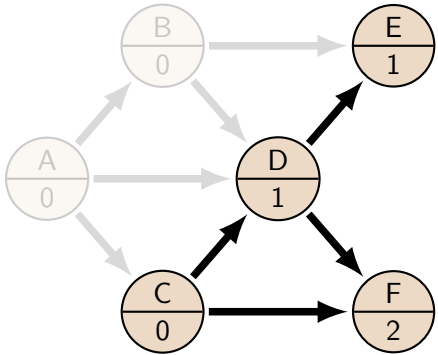
Topological Order

- Now Both B and C have indegree = 0. Select either one to be added to the topological ordering.

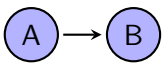


Topological Order

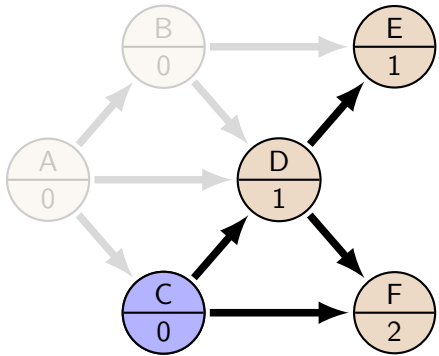
■ Add node B to the topological ordering and remove it from graph



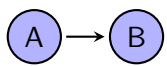
Topological Order



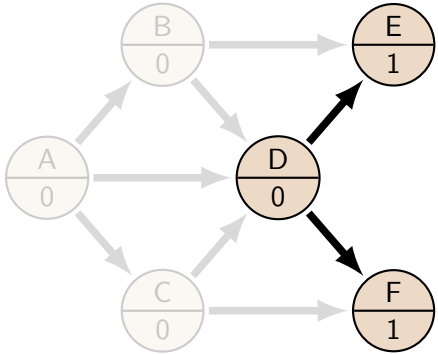
- Node C doesn't have any dependencies



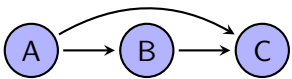
Topological Order



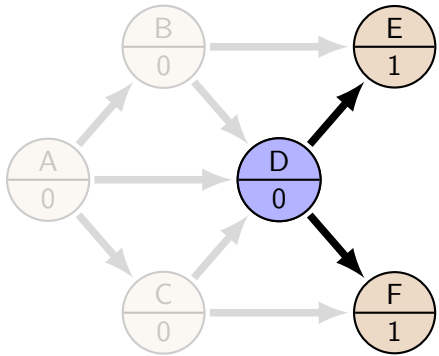
■ Add node C to the topological ordering and remove it from graph



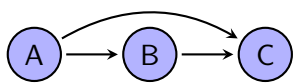
Topological Order



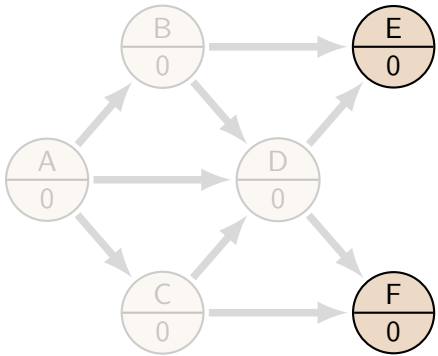
- Node D doesn't have any dependencies



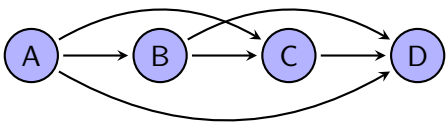
Topological Order



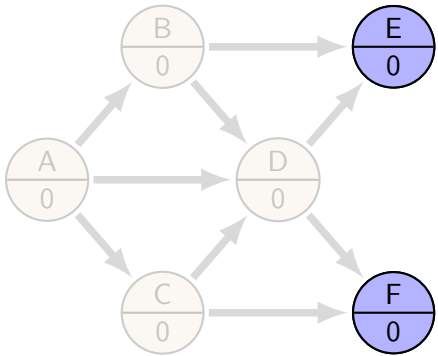
■ Add node D to the topological ordering and remove it from graph



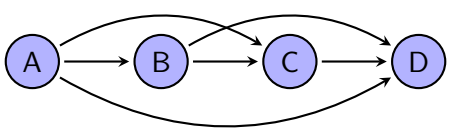
Topological Order



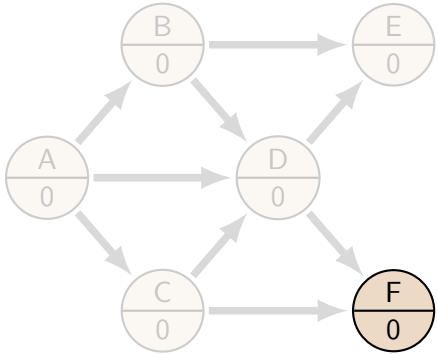
- Now Both E and F have indegree = 0. Select either to be added to the topological ordering.



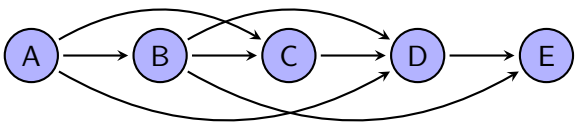
Topological Order



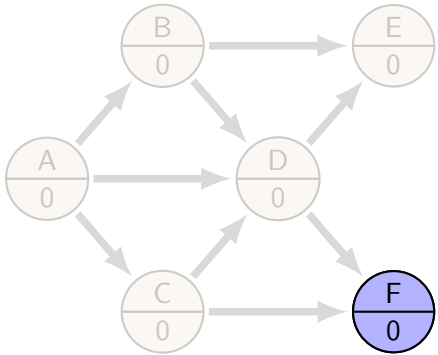
■ Add node E to the topological ordering and remove it from graph



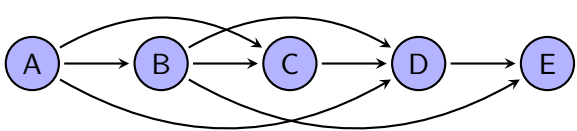
Topological Order



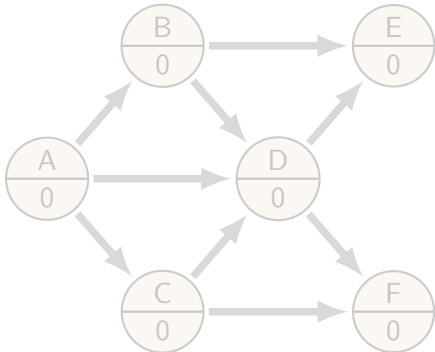
- F is the final node



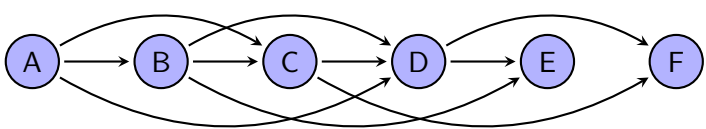
Topological Order



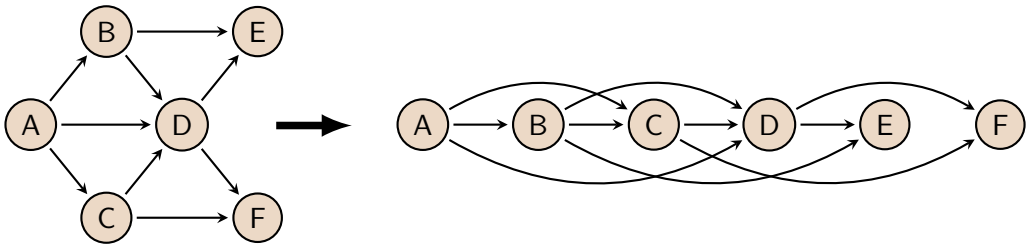
■ Add node F to the topological ordering and remove it from graph



Topological Order



- We have found the topological order



Unsorted Graph

Topologically Sorted Graph

- Kahn's Algorithm

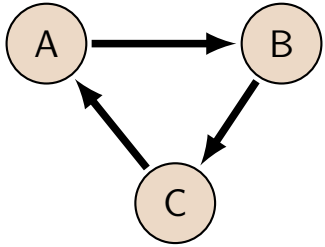
```
1:  procedure TopoSort(G)
2:      create a list L
3:      create a queue Q
4:      for each vertex v ∈ G do
5:          if the indegree of v = 0 then
6:              put v into the Q
7:      while Q is not empty do
8:          pop a vertex v out of Q
9:          add v to the end of L
10:         for each edge (u, v) ∈ G do
11:             decrement the indegree of u
12:             if the indegree of u = 0 then
13:                 put u into the Q
14:     return L
```

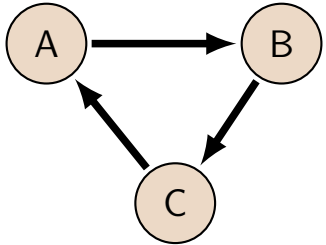
- Kahn's Algorithm
- BFS

```
1:  procedure TopoSort(G)
2:      create a list L
3:      create a queue Q
4:      for each vertex v ∈ G do
5:          if the indegree of v = 0 then
6:              put v into the Q
7:      while Q is not empty do
8:          pop a vertex v out of Q
9:          add v to the end of L
10:         for each edge (u, v) ∈ G do
11:             decrement the indegree of u
12:             if the indegree of u = 0 then
13:                 put u into the Q
14:     return L
```

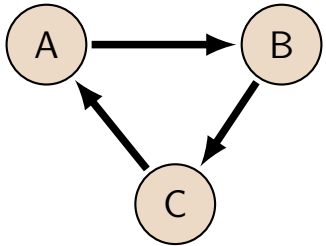
- Kahn's Algorithm
- BFS
- $O(|V| + |E|)$

```
1:  procedure TopoSort(G)
2:      create a list L
3:      create a queue Q
4:      for each vertex v ∈ G do
5:          if the indegree of v = 0 then
6:              put v into the Q
7:      while Q is not empty do
8:          pop a vertex v out of Q
9:          add v to the end of L
10:         for each edge (u, v) ∈ G do
11:             decrement the indegree of u
12:             if the indegree of u = 0 then
13:                 put u into the Q
14:     return L
```


A
B
C

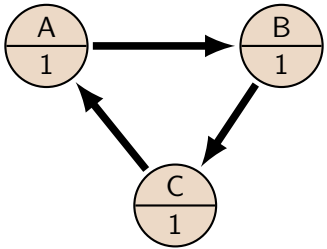


What is the topological order?



What is the topological order?

Not possible :(



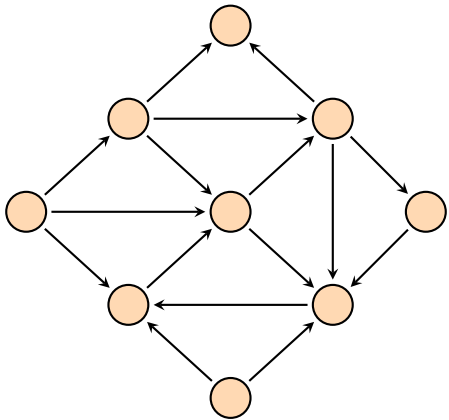
What is the topological order?

Not possible :(

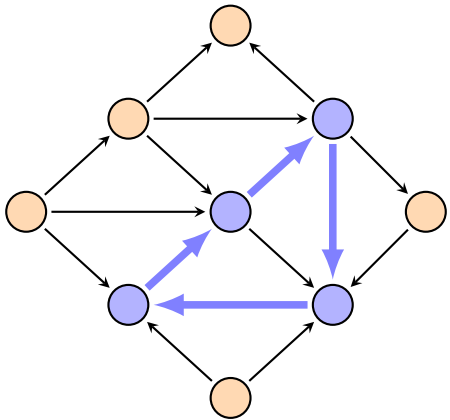
Topological sorting is applicable for Directed Acyclic Graph (DAG).

WHERE?

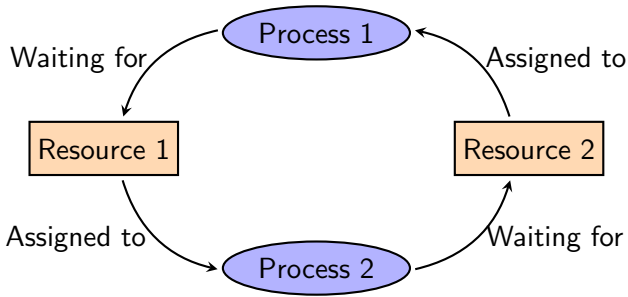
- Detecting cycle in a graph.



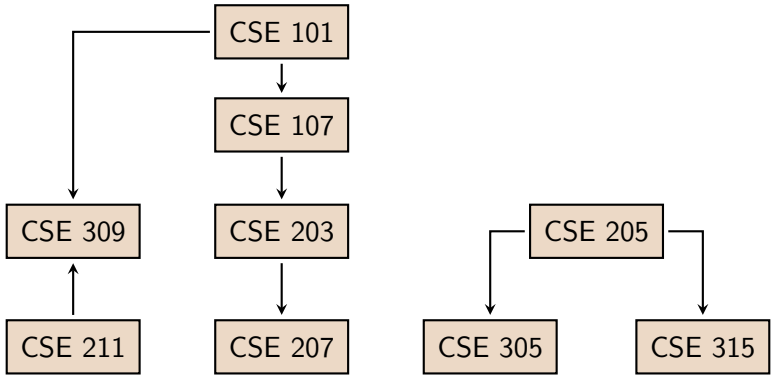
- Detecting cycle in a graph.



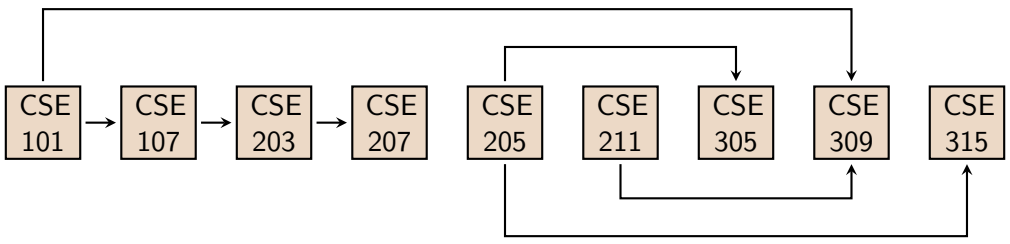
- Operation System deadlock detection.



- Course Schedule problem.

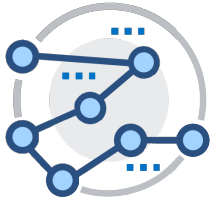


- Course Schedule problem.



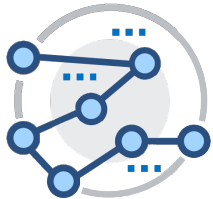
Dependency resolution

Dependency resolution

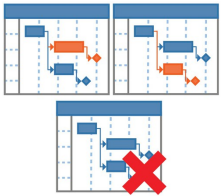


Manufacturing workflow

Dependency resolution

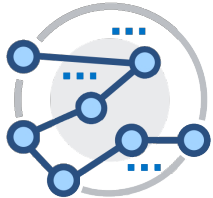


Manufacturing workflow

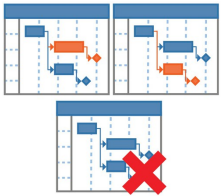


Critical path analysis

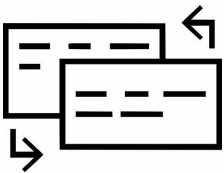
Dependency resolution



Manufacturing workflow

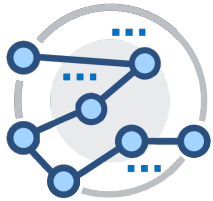


Critical path analysis

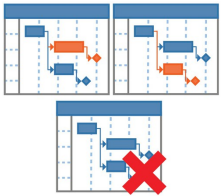


Sentence ordering

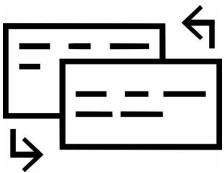
Dependency resolution



Manufacturing workflow



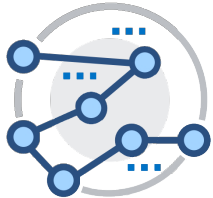
Critical path analysis



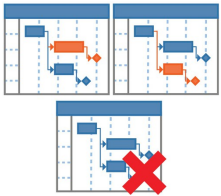
Sentence ordering

Task scheduling

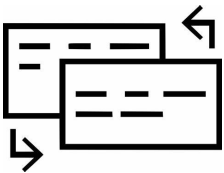
Dependency resolution



Manufacturing workflow



Critical path analysis



Sentence ordering

Task scheduling

Data serialization

THANK YOU