Tutorial 10: Graph algorithms

CSCI2520 - DATA STRUCTURES AND APPLICATIONS

TUTOR: ZHENG CHENGUANG

Outlines

- Shortest-Path Algorithm
- Topological Sort

Shortest-Path Algorithm

- Single Source Shortest Paths Algorithm (SSSP)
  - Dijkstra's algorithm
  - Bellman–Ford algorithm

Dijkstra's algorithm

- Initialize the cost/distance table.
- Pick the unvisited vertex with the min cost and mark it as visited.
- Update the best cost of the adjacent vertices if needed.

- Q: What will happen if there are some negative weight edge? How to solve it?

Bellman–Ford algorithm

- Initialize the cost/distance table.
- Relax edges repeatedly
  - Update the distance table by iterating the edges.
- Check for negative-weight cycles

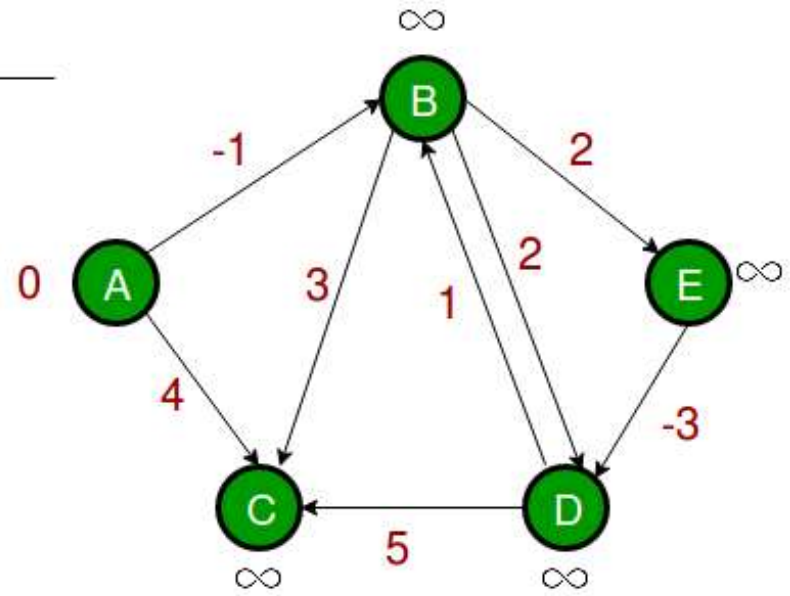
Bellman–Ford algorithm

```
BELLMAN-FORD(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  for i  1 to |V[G]| - 1
       do for each edge (u, v)  E[G]
            if d[u] + w(u, v) < d[v]
              do RELAX(u, v, w)
  for each edge (u, v)  E[G]
       do if d[v] > d[u] + w(u, v)
            then return FALSE
  return TRUE
```

- Why it can handle this problem and $O(?)$

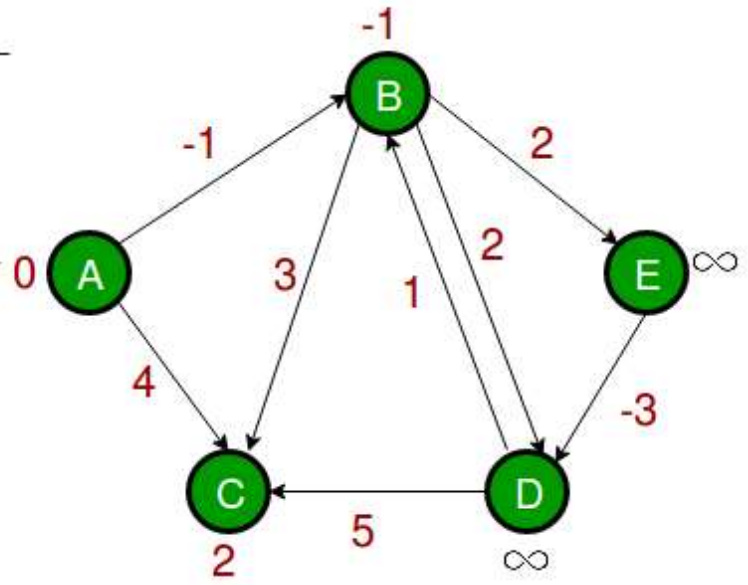
Bellman–Ford algorithm

A	B	C	D	E
0	∞	∞	∞	∞



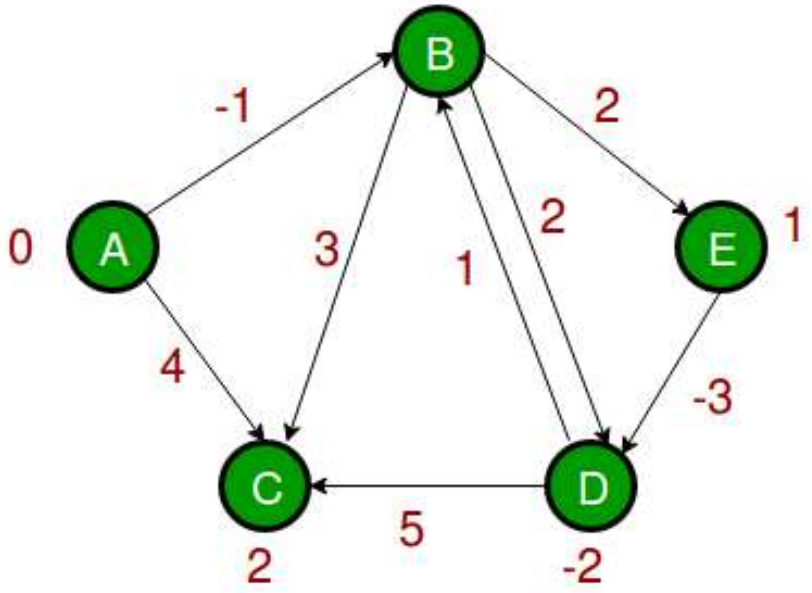
Bellman–Ford algorithm

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



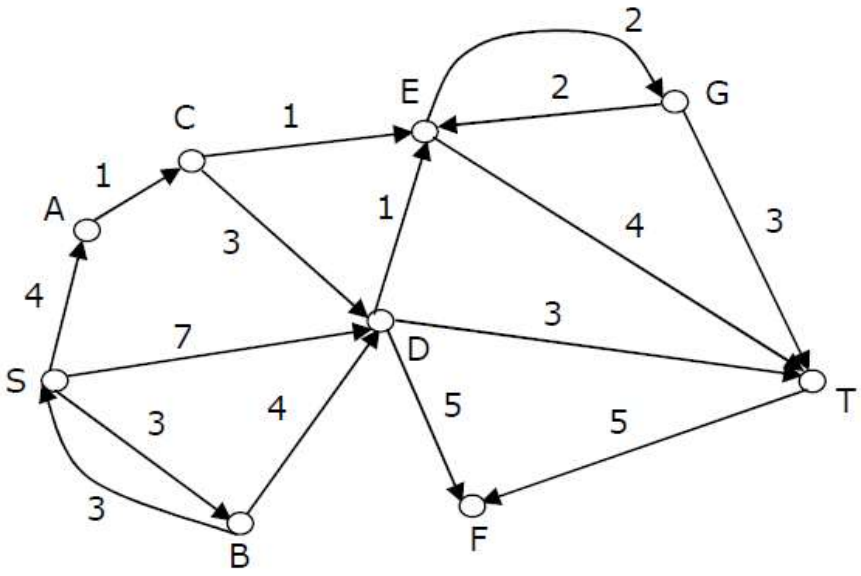
Bellman–Ford algorithm

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Exercise 1

- Consider the directed graph shown in the figure below. Which one will be reported by Dijkstra's shortest path algorithm?
- A: SDT
- B: SBDT
- C: SACDT
- D: SACET



Exercise 2

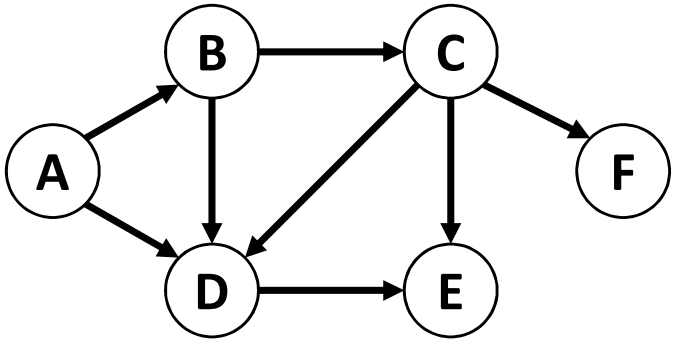
- To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, the data structure to be used is:
- A:Queue
- B:Stack
- C:Heap

Exercise 3

- In a weighted graph, assume that the shortest path from a source 's' to a destination 't' is correctly calculated using a shortest path algorithm. Is the following statement true? If we increase weight of every edge by 1, the shortest path always remains same.
- A: YES
- B: NO

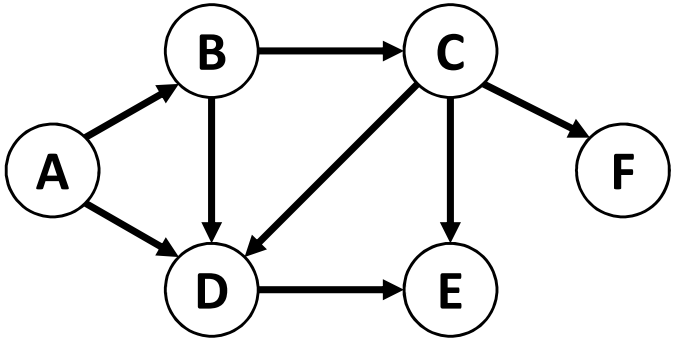
Topological Sort

- Given a directed acyclic graph(DAG) G = (V , E), find an ordering of V, such that, for each edge (u, v) ∈ E, u precedes v in the ordering. The ordering is called a **topological order**.

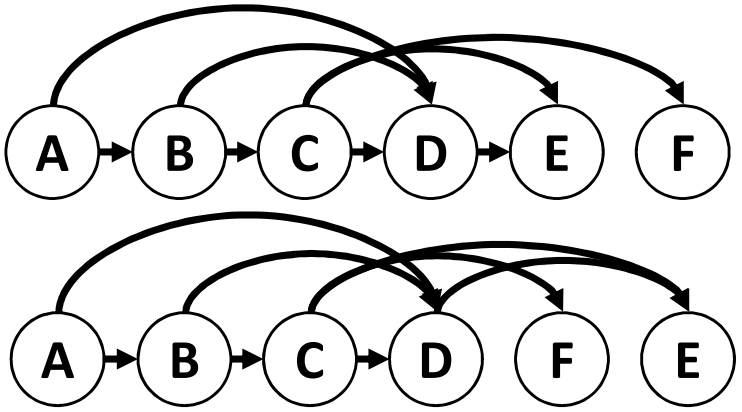


Topological Sort

- Any ordering in which all the arrows go to the right is a valid solution (may not unique).



Both are valid!

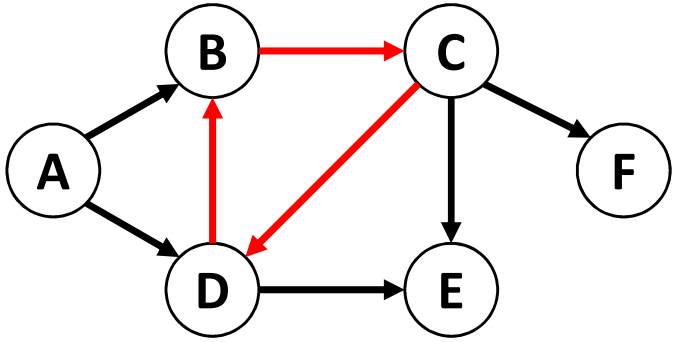


Toposort Examples

- Taking courses
  - In order to take a course, you must take all of its prerequisites first
- In computer science,
  - Instruction scheduling
  - Determining the order of compilation tasks

Why DAG?

- A directed graph with a cycle cannot be topologically sorted.

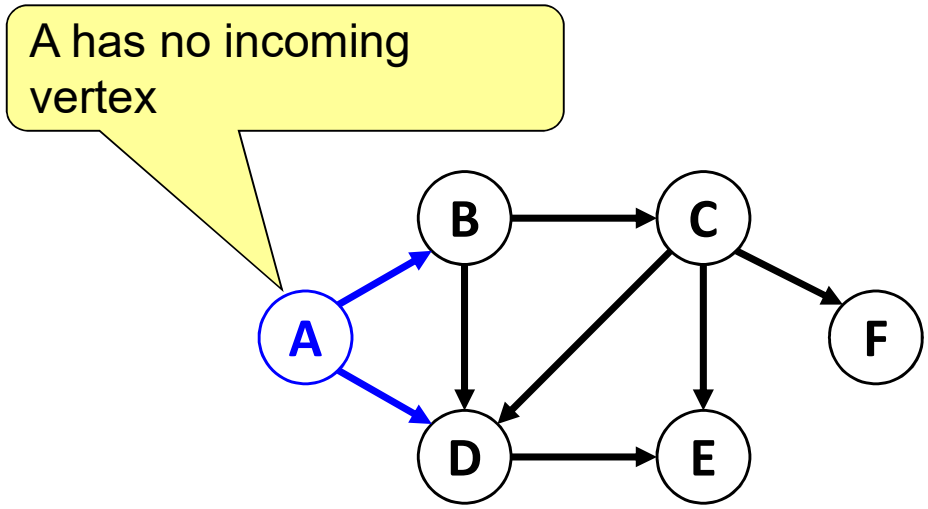


- Question: why?
- Any ordering of B, C and D causes at least an arrow going to the left.

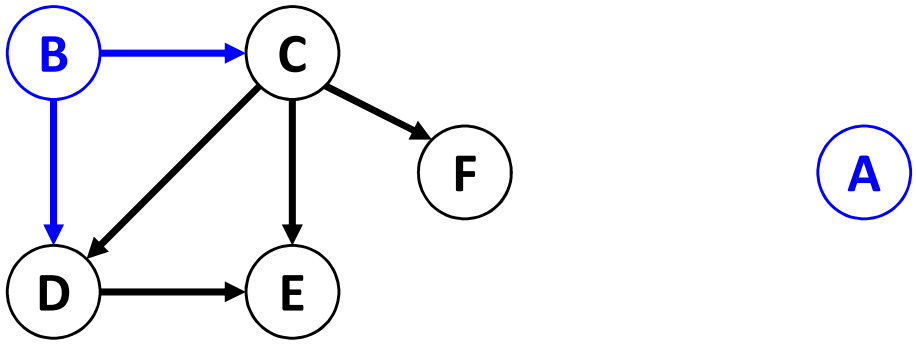
Kahn's Algorithm

- Choose vertices in the same order as the eventual topological sort
- Recursively find vertices without incoming edges

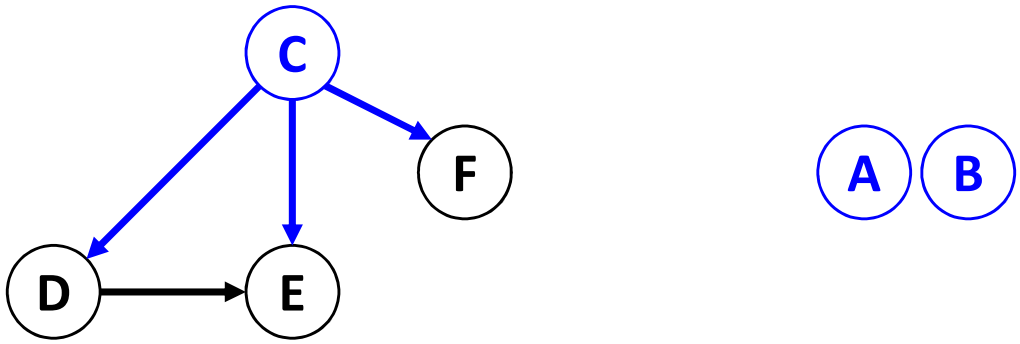
Kahn's Algorithm



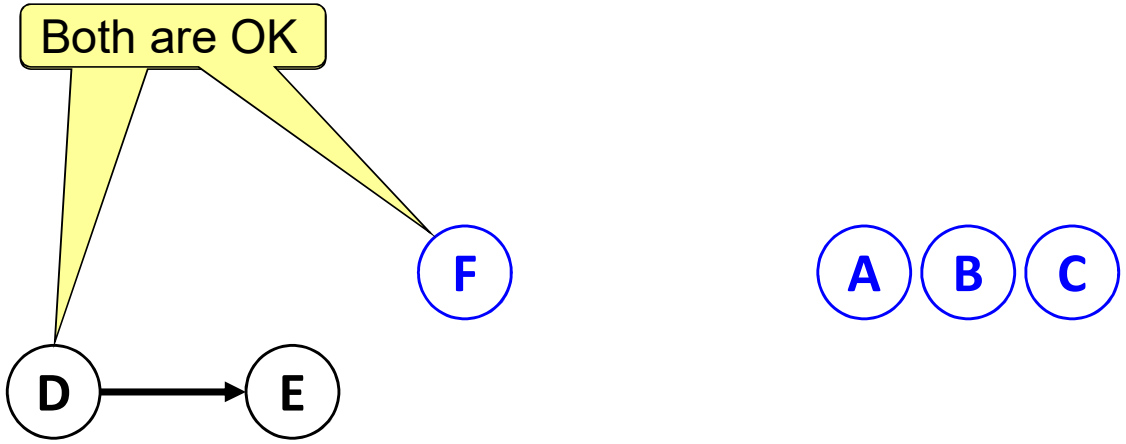
Kahn's Algorithm



Kahn's Algorithm

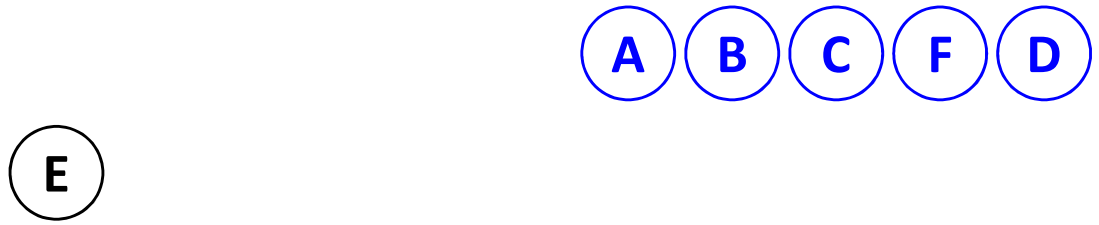


Kahn's Algorithm

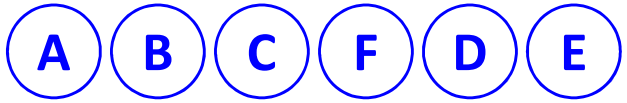


Kahn's Algorithm

Kahn's Algorithm



Kahn's Algorithm



Kahn's Algorithm Details

- How to obtain in-degree information? Go through all edges in each round?

- In what situation it still can have some vertex left?

Kahn's Algorithm Details

- Which graph representation fits this algorithm better, adjacency matrix or adjacency list?

Kahn's Algorithm Details

- Which graph representation fits this algorithm better, adjacency matrix or adjacency list? **Adjacency list!**

	Adjacency matrix	Adjacency list
Is there an edge from v_i to v_j?	M[i][j] == 1? O(1)	Traverse L[i] O(d)
Find all vertices adjacent to v_i.	Traverse row O(n)	Traverse L[i] O(d)
How many edges are there in a graph?	Traverse M O(n^2)	Traverse L O($n + e$)

Exercise 4

- Which of the following algorithm can be used to **efficiently** calculate single source shortest paths in a Directed Acyclic Graph?
- A: Dijkstra
- B: Bellman-Ford
- C: Topological Sort

Q&A