VE281

Data Structures and Algorithms

Breadth-Frist Search, Topological Sorting and Shortest Path Problem

Review

- Graph Basics
  - Directed graph, undirected graph, path, cycle, node degree, weighted graph, etc.
- Graph Representation
  - Adjacency matrix
  - Adjacency list
- Graph Search
  - Depth-first search

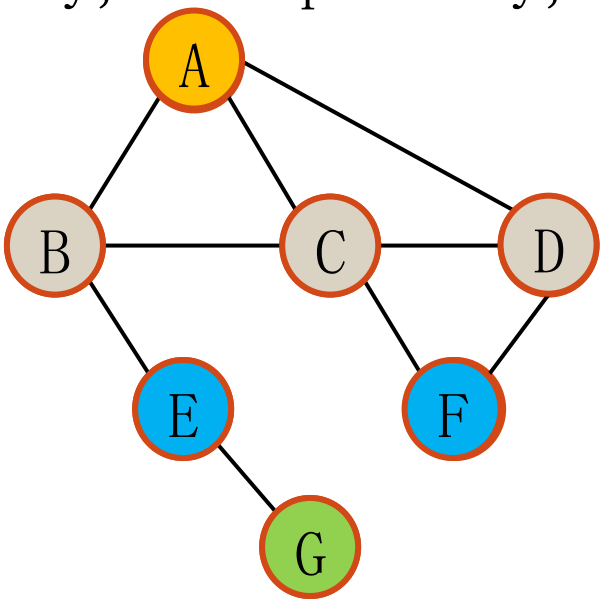
```
DFS(v) {
  visit v;
  mark v as visited;
  for(each node u adjacent to v)
    if(u is not visited) DFS(u);
}
```

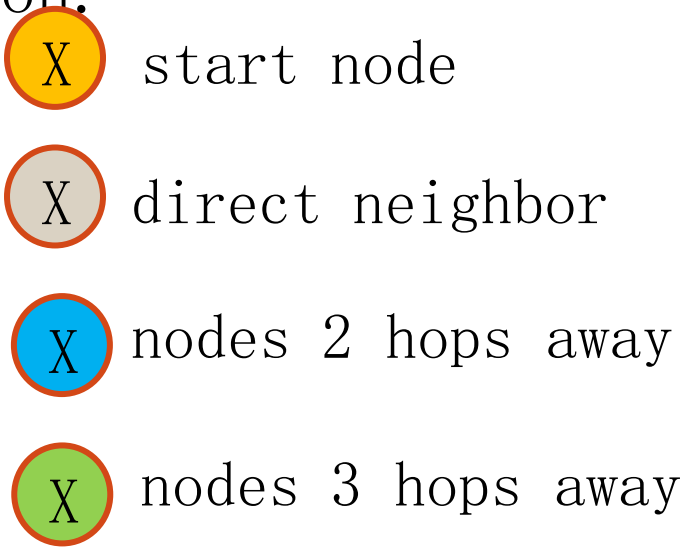
Outline

- Breadth-First Search
- Topological Sorting
- Shortest Path Problem

Breadth-First Search (BFS)

- Given a start node, visit all directly connected neighbors first, then nodes 2 hops away, 3 hops away, and so on.





A→B→C→D→E→F→G

Breadth-First Search (BFS)

Implementation

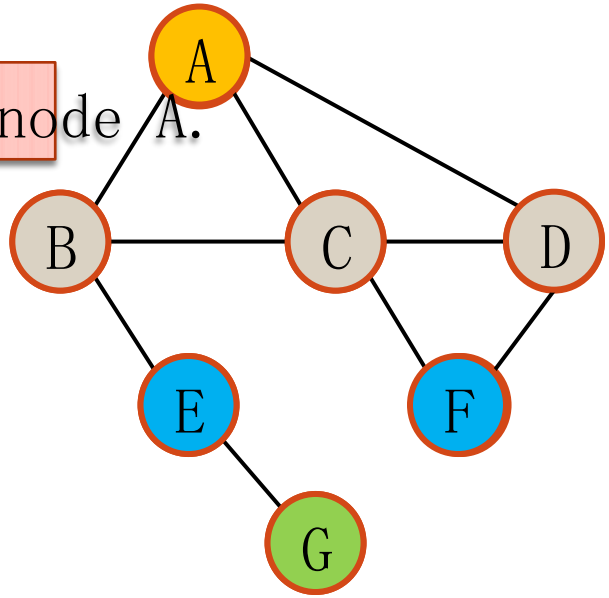
- BFS can be implemented using a queue.

```
BFS(s) {
  queue q; // An empty queue
  visit s and mark s as visited;
  q.enqueue(s);
  while(!q.isEmpty()) {
    v = q.dequeue();
    for(each node u adjacent to v) {
      if(u is not visited) {
        visit u and mark u as visited;
        q.enqueue(u);
      }
    }
  }
}
```

Breadth-First Search (BFS)

Example

Start node is node A.




```
BFS(s) {
  queue q; // An empty queue
  visit s and mark s as visited;
  q.enqueue(s);
  while(!q.isEmpty()) {
    v = q.dequeue();
    for(each node u adjacent to v) {
      if(u is not visited) {
        visit u and mark u as visited;
        q.enqueue(u);
      }
    }
  }
}
```

Queue: A B C D E F G

Visit Order: A B C D E F G

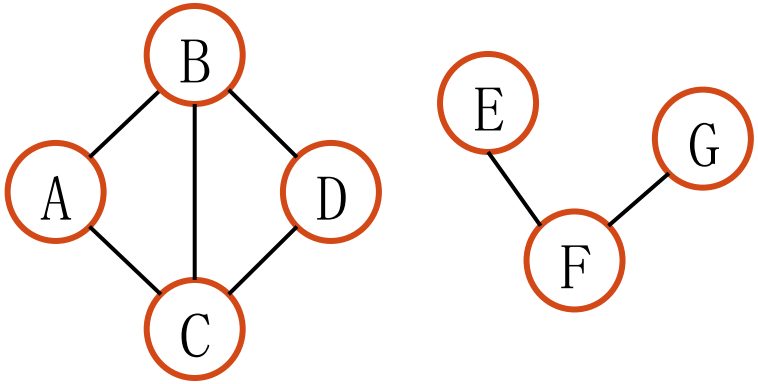
Breadth-First Search (BFS)

Time Complexity

- Same complexity as DFS:
  - Each node is visited exactly once.
  - Adjacency list (or row) of each node is scanned once.
  - For adjacency matrix representation: $O(|V|^2)$.
  - For adjacency list representation: $O(|V| + |E|)$.

Traverse All the Nodes in a Graph

- The graph may not be connected. How can we traverse all the nodes in the graph?



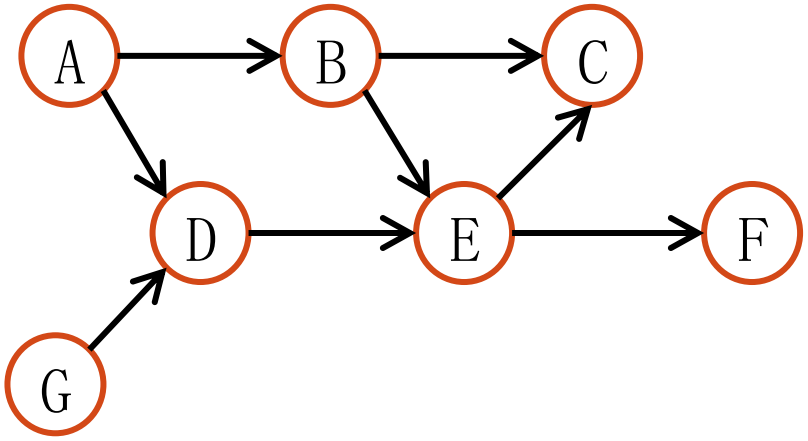

```
for(each node v in the graph)
  if(v is not visited)
    DFS(v);
```

Outline

- Breadth-First Search
- Topological Sorting
- Shortest Path Problem

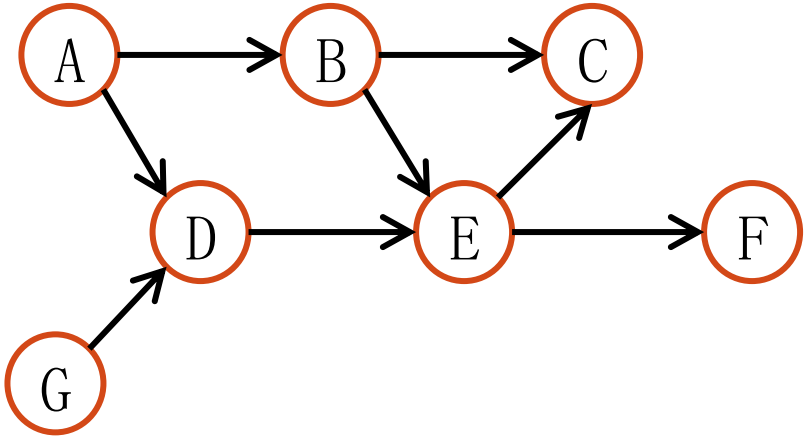
Topological Sorting

- **Topological sorting**: a sorting on directed acyclic graph.
- Given a directed acyclic graph G, order the nodes such that for each edge $(v_i, v_j) \in E$, v_i is before v_j in the ordering.



A topological sorting is: A, G, D, B, E, C, F

Topological Sorting

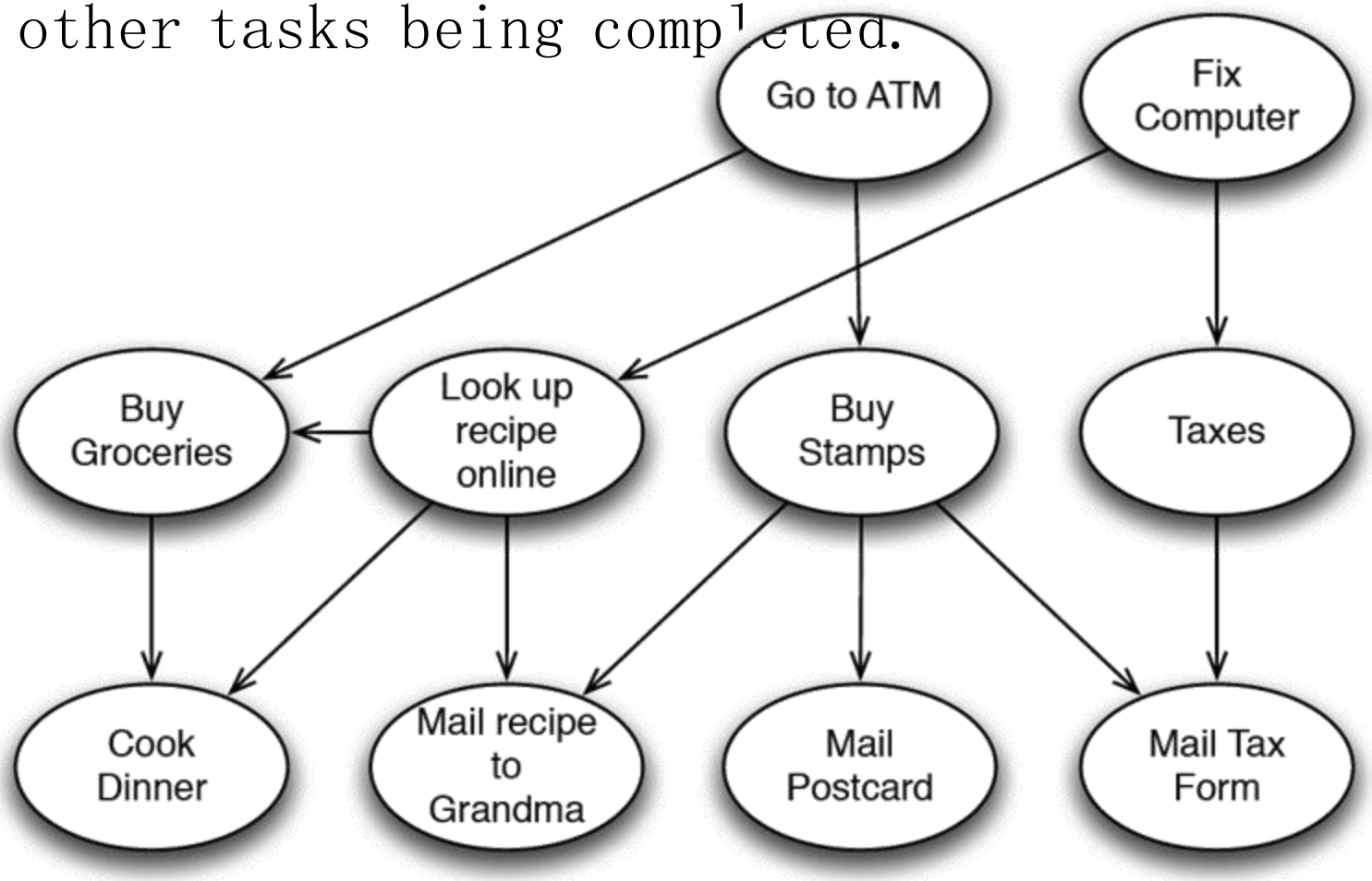


- Topological sorting is not necessarily unique:
  - A, G, D, B, E, C, F and A, B, G, D, E, F, C are both topological sorting.
- Are the following orderings topological sorting?
  - A, B, E, G, D, C, F
  - A, G, B, D, E, F, C

Topological Sorting

Applications

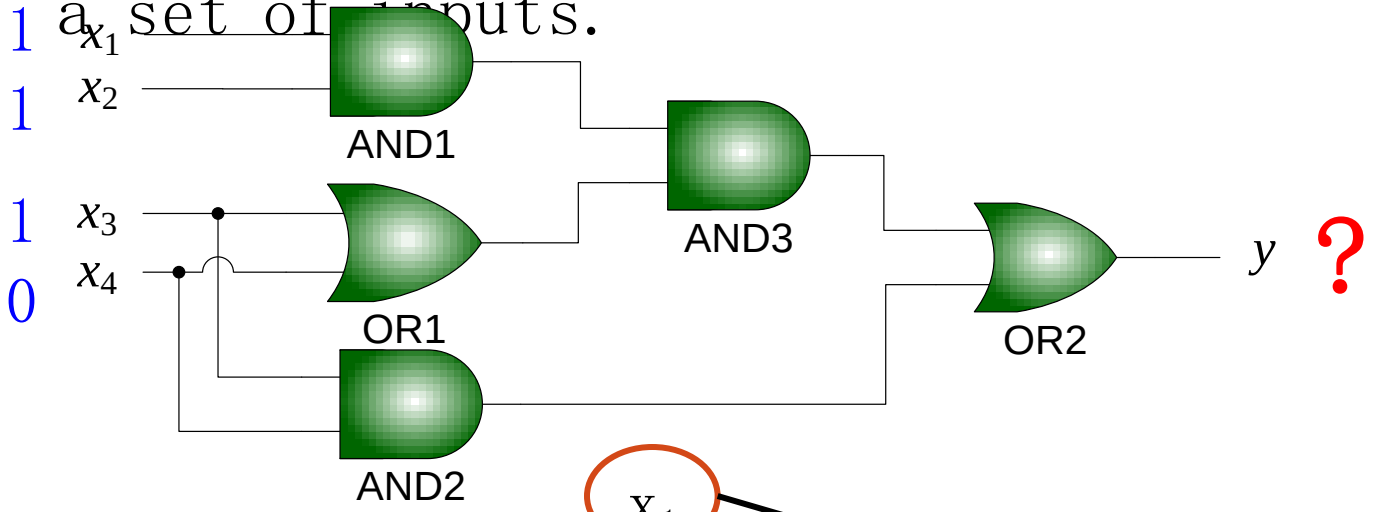
- Scheduling tasks when some tasks depend on other tasks being completed.

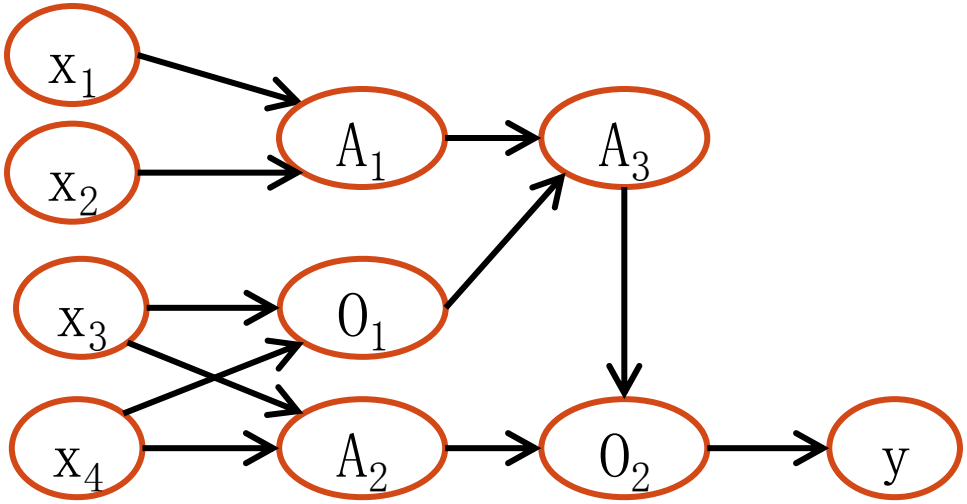


Topological Sorting

Applications

- Evaluating a combination logic circuit given a set of inputs.





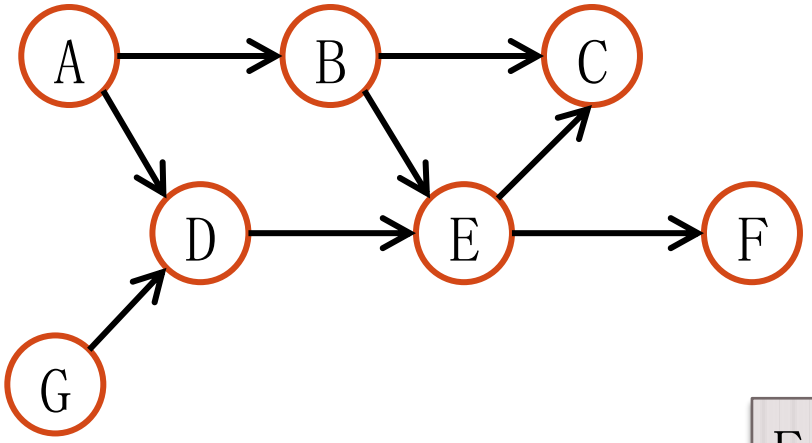
Topological Sorting

Algorithm

1. Compute the in-degrees of all nodes.
2. Enqueue all in-degree 0 nodes into a queue.
3. While queue is not empty
    1. Dequeue a node v from the queue and visit it.
    2. Decrement in-degrees of node v's neighbors.
    3. If any neighbor's in-degree becomes 0, place it in the queue.

Topological Sorting Algorithm

Example



Queue

Enqueue A and G

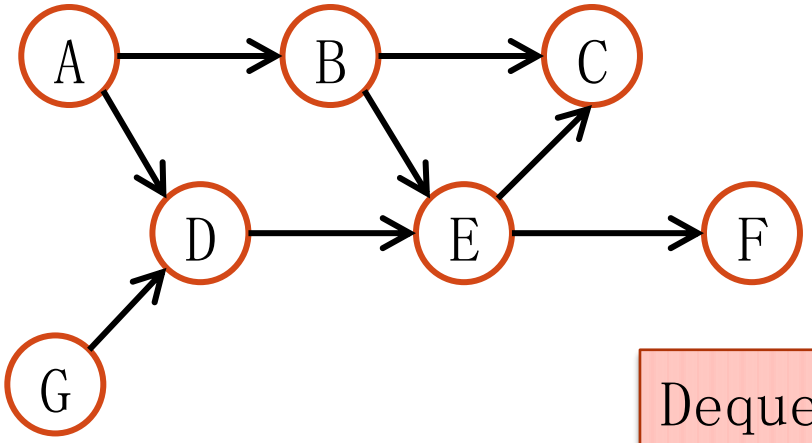
In-degrees

A	B	C	D	E	F	G
0	1	2	2	2	1	0

Order

Topological Sorting Algorithm

Example



Queue

A
G

Dequeue A, visit A, and decremen
in-degrees of A's neighbors.

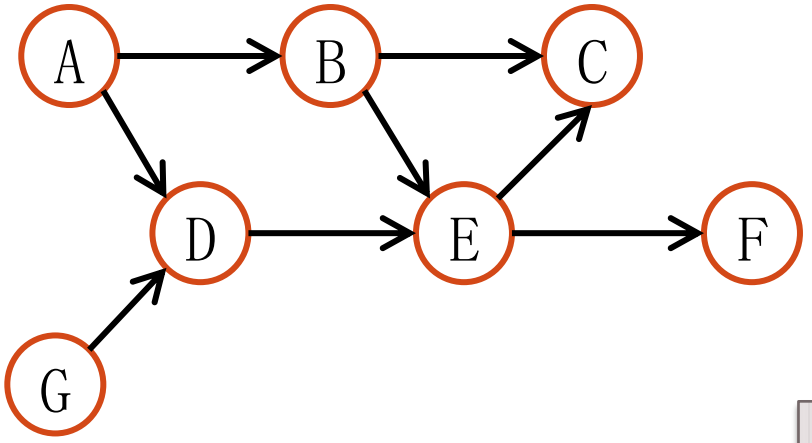
In-degrees

A	B	C	D	E	F	G
0	1	2	2	2	1	0

Order

Topological Sorting Algorithm

Example



Queue

G

Enqueue B

In-degrees

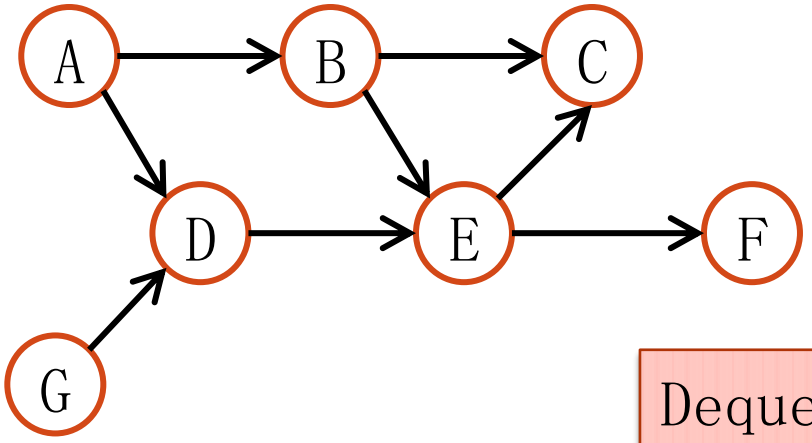
A	B	C	D	E	F	G
0	~~1~~ 0	2	~~2~~ 1	2	1	0

Order

A						

Topological Sorting Algorithm

Example



Queue

G
B

Dequeue G, visit G, and decremen
in-degrees of G's neighbors.

In-degrees

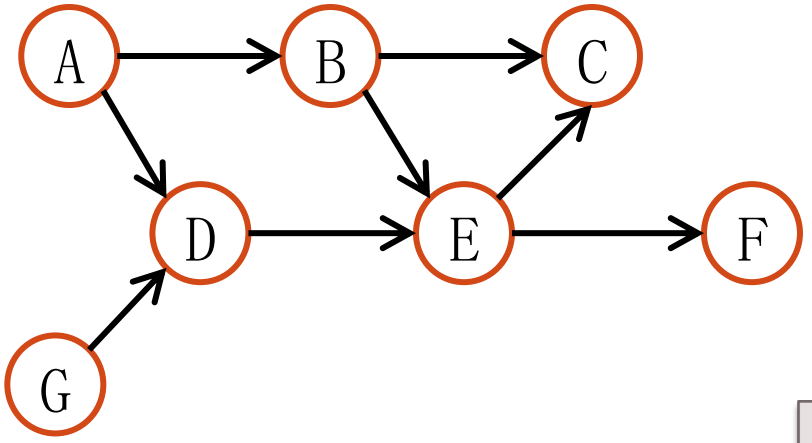
A	B	C	D	E	F	G
0	0	2	1	2	1	0

Order

A						

Topological Sorting Algorithm

Example



Queue

B

Enqueue D

In-degrees

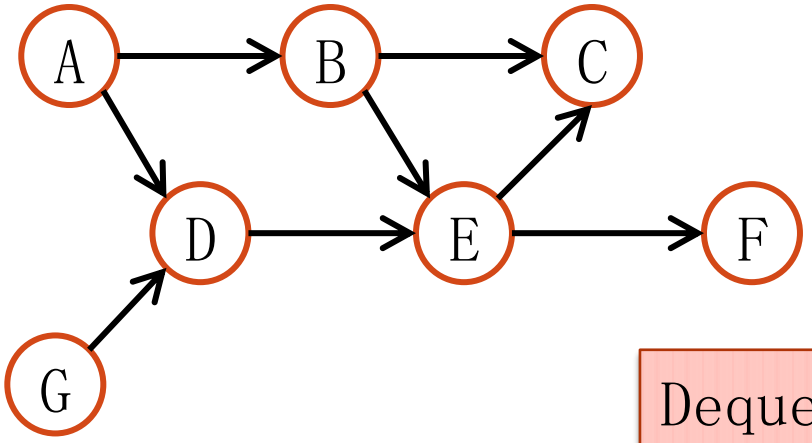
A	B	C	D	E	F	G
0	0	2	1 0	2	1	0

Order

A	G					

Topological Sorting Algorithm

Example



Queue

B
D

Dequeue B, visit B, and decremen
in-degrees of B's neighbors.

In-degrees

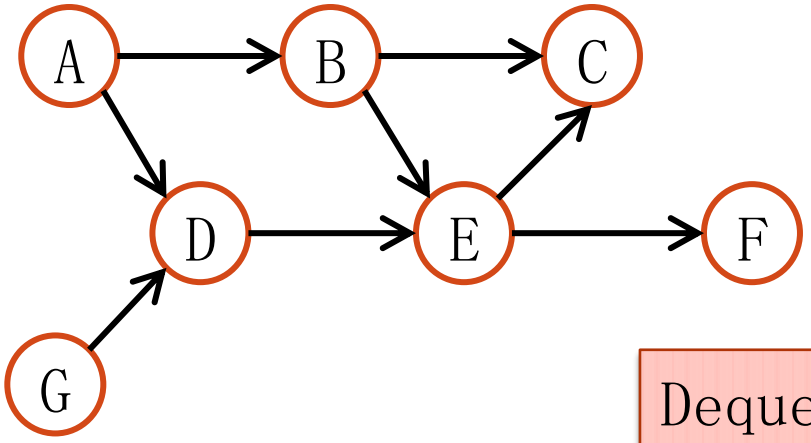
A	B	C	D	E	F	G
0	0	2	0	2	1	0

Order

A	G					

Topological Sorting Algorithm

Example



Queue

D

Dequeue D, visit D, and decremen
in-degrees of D's neighbors.

In-degrees

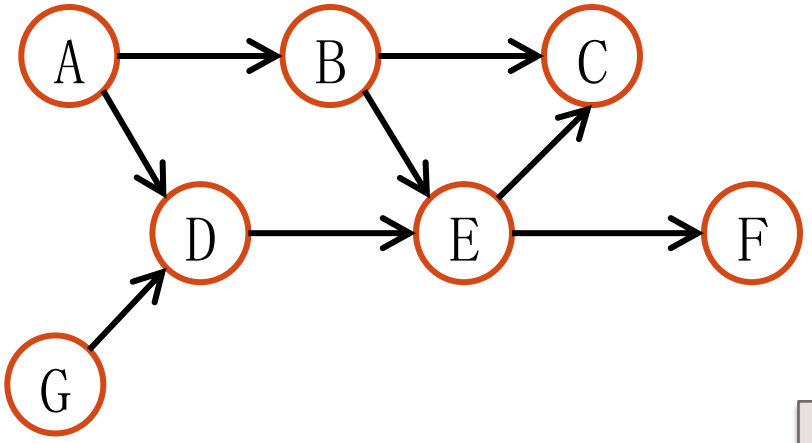
A	B	C	D	E	F	G
0	0	~~2~~ 1	0	~~2~~ 1	1	0

Order

A	G	B				

Topological Sorting Algorithm

Example



Queue

Enqueue E

In-degrees

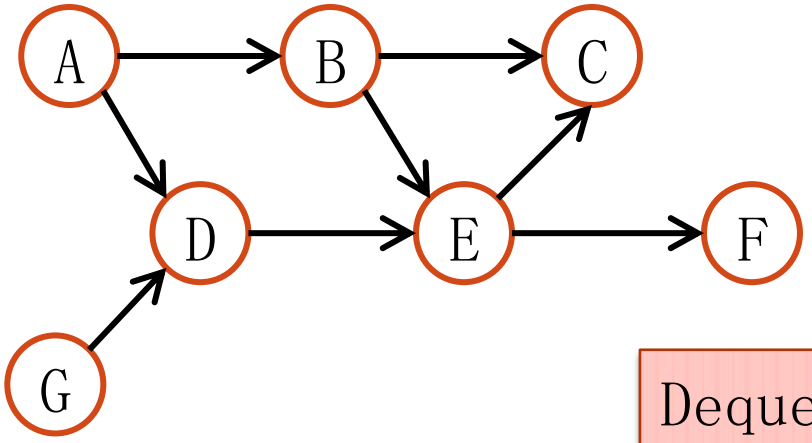
A	B	C	D	E	F	G
0	0	1	0	~~1~~ 0	1	0

Order

A	G	B	D			

Topological Sorting Algorithm

Example



Queue

E

Dequeue E, visit E, and decremen
in-degrees of E's neighbors.

In-degrees

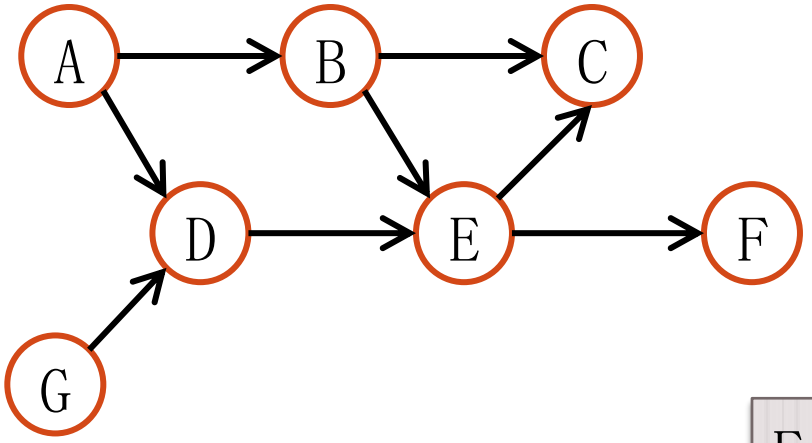
A	B	C	D	E	F	G
0	0	1	0	0	1	0

Order

A	G	B	D			

Topological Sorting Algorithm

Example



Queue

Enqueue C and F

In-degrees

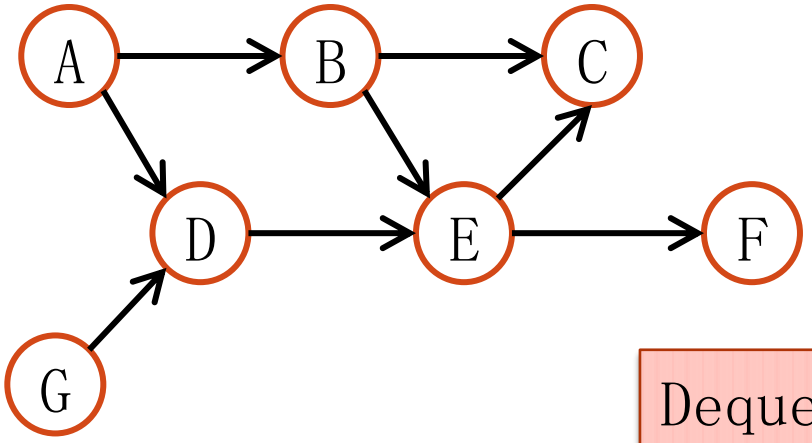
A	B	C	D	E	F	G
0	0	1 0	0	0	1 0	0

Order

A	G	B	D	E		

Topological Sorting Algorithm

Example



Queue

C
F

Dequeue C, visit C, and decremen
in-degrees of C's neighbors.

In-degrees

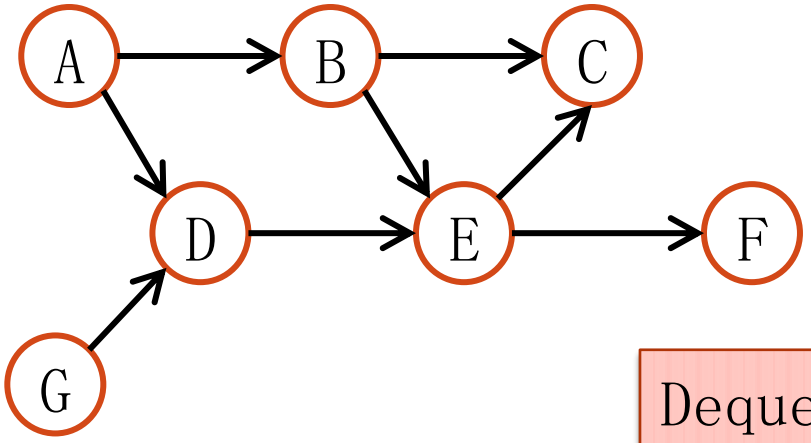
A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E		

Topological Sorting Algorithm

Example



Queue

F

Dequeue F, visit F, and decremen
in-degrees of F’ s neighbors.

In-degrees

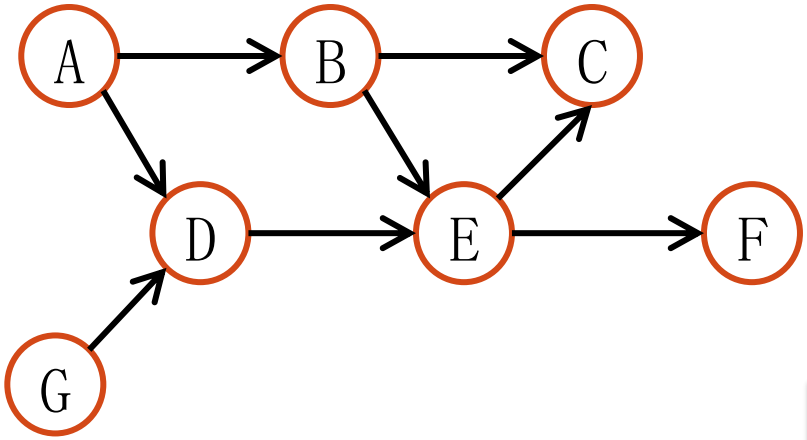
A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E	C	

Topological Sorting Algorithm

Example



Queue

Queue is now empty. Done!

In-degrees

A	B	C	D	E	F	G
0	0	0	0	0	0	0

Order

A	G	B	D	E	C	F

Topological Sorting Algorithm

Time Complexity

Assume adjacency list representation

1. Compute the in-degrees of all nodes. $O(|V| + |E|)$ in total
2. Enqueue all in-degree 0 nodes into a queue.
3. While queue is not empty
   1. Dequeue a node v from the queue and visit it. $O(|V|)$ in total
   2. Decrement in-degrees of node v's neighbors. $O(|E|)$ in total
   3. If any neighbor's in-degree becomes 0, place it in the queue. $O(|V|)$ in total

Total running time is $O(|V| + |E|)$.

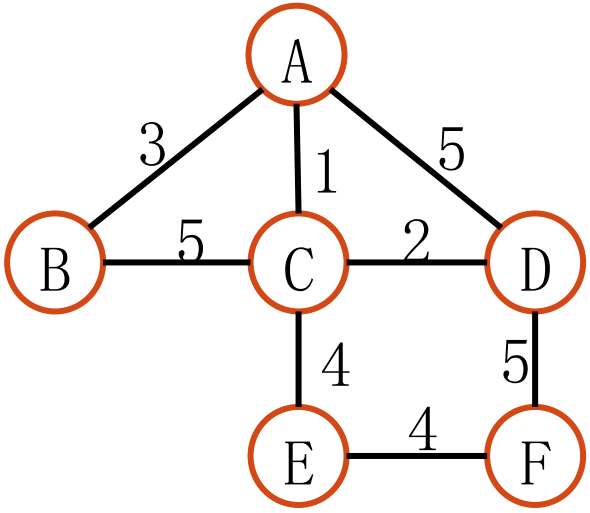
Outline

- Breadth-First Search
- Topological Sorting
- Shortest Path Problem

Shortest Path Problem

Introduction

- Given a weighted graph $G = (V, E)$, **path length** is defined as the sum of weights of edges on the path.
  - E.g., length of the path B, C, E, F is 13.
- **Shortest path problem**: given a weighted graph $G = (V, E)$ and two node $s, d \in V$, find the shortest path from s to d.



What is the shortest path from B to F?

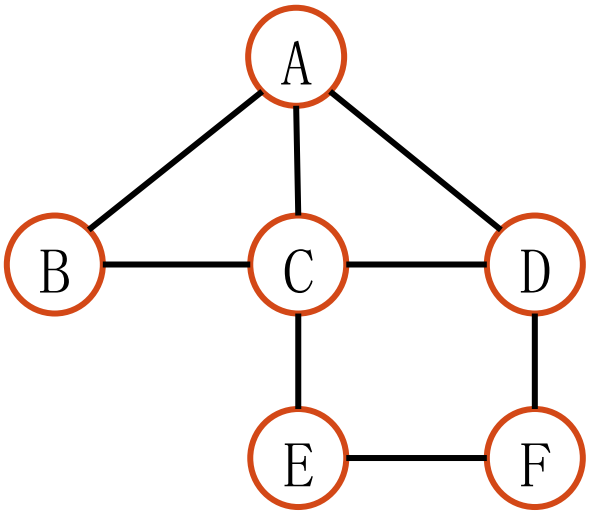
Shortest Path Problem

- The starting node on the path is the **source** node and the ending node is the **destination** node.
- The previous problem is a **single source single destination** problem.
- What we will solve is a **single source all destinations** problem: Given $G = (V, E)$ and a node $s \in V$, find the shortest path from s to **every other** node in G.
  - Single source single destination problem can be solved by solving the single source all destinations problem.
  - However, single source single destination problem is not much easier than the single source all destinations problem.

Shortest Path Problem

A Simple Version: Unweighted Graphs

- For an unweighted graph, path length is defined as the number of edges on the path.
- How do you obtain the shortest path between a pair of nodes?

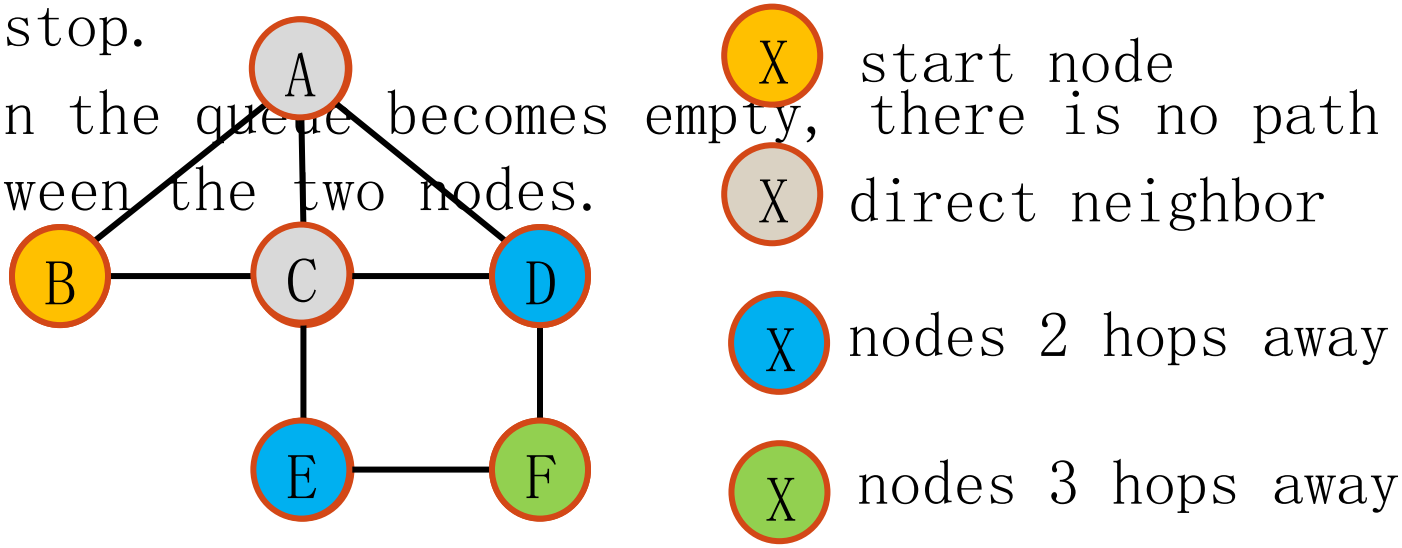


What is the shortest path from B to F?

Using breadth-first search!

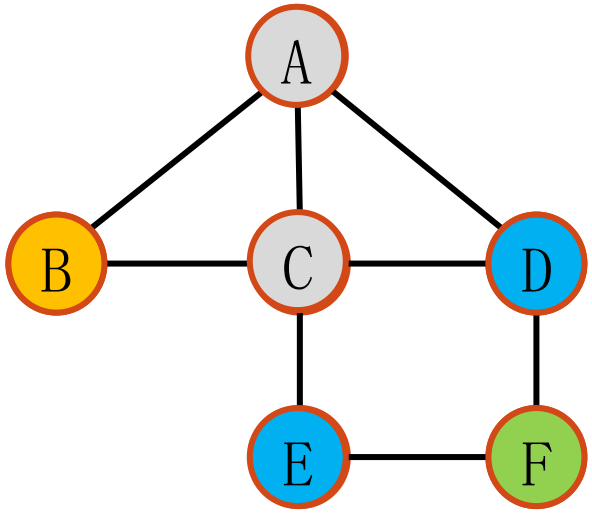
Shortest Path for Unweighted Graphs

- Recall breadth-first search (BFS): Given a start node, visit all directly connected neighbors first, then nodes 2 hops away, 3 hops away, and so on.
  - This is exactly what we want!
  - When the node visited is the destination node, we stop.
  - When the queue becomes empty, there is no path between the two nodes.



Shortest Path for Unweighted Graphs

- Additional bookkeeping
  - Store the distance.
  - Store the predecessor on the shortest path, i.e., the previous node on the path.



X start node

X direct neigh

X nodes 2 hops away

X nodes 3 hops away

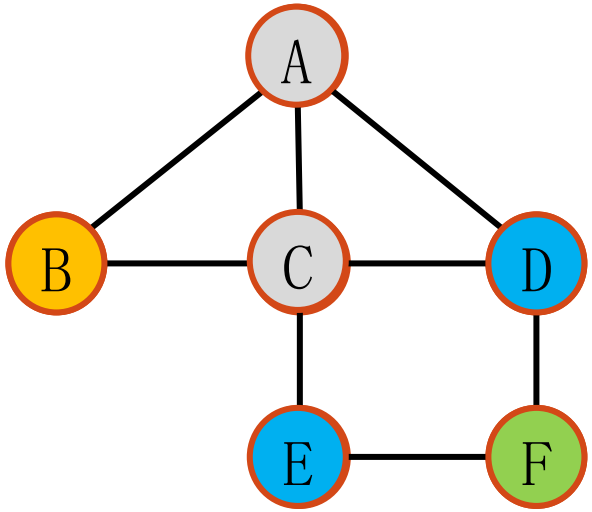
	A	B	C	D	E	F
dist	1	0	1	2	2	3
pred	B	-	B	A	C	D

Shortest Path for Unweighted Graphs

- We can obtain the shortest path by backtracking.

  B→A→D→F

  - E.g., shortest from B to F



X start node

X nodes 2 hops away

X direct neigh

X nodes 3 hops away

	A	B	C	D	E	F
dist	1	0	1	2	2	3
prev	B	–	B	A	C	D