

ECE2810J

Data Structures and Algorithms

Topological Sorting

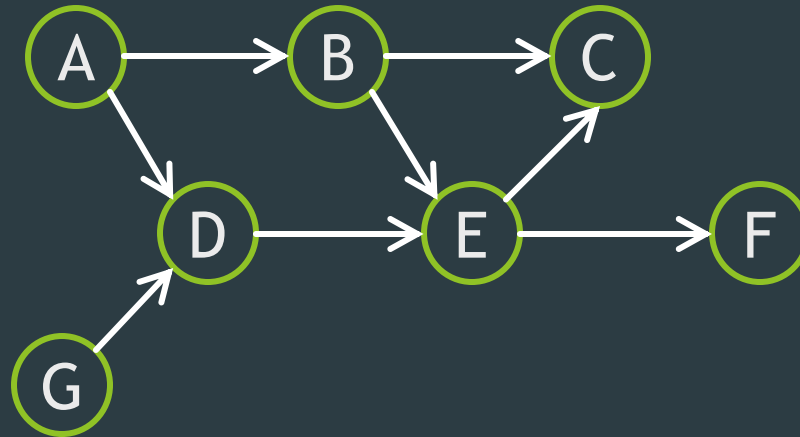
► Learning Objectives:

- Know what a topological sorting is and why it is useful
- Know the topological sorting algorithm and its runtime complexity



Topological Sorting

- ▶ **Topological sorting** : an ordering on nodes of a **directed graph** so that for each edge (v_i, v_j) (means: an edge **from** v_i **to** v_j) in the graph, v_i is before v_j in the ordering.
 - ▶ Also known as **topological ordering**.

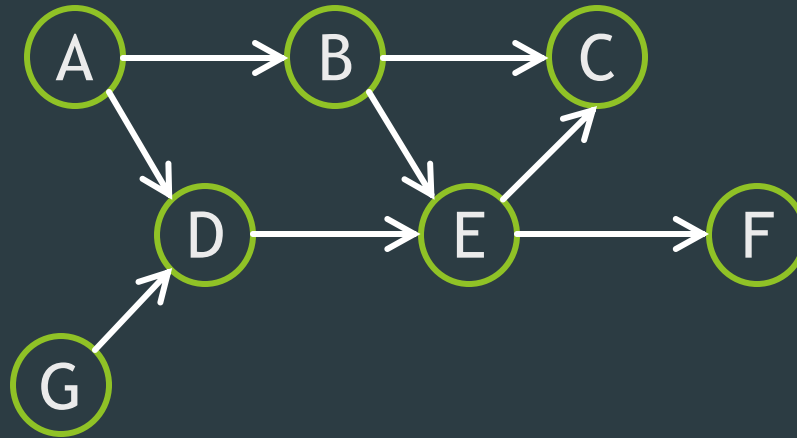


A topological sorting is: A, G, D, B, E, C, F

Which Graph Has Topological Sorting?

- ▶ Is there any “topological sorting” for directed graph **with cycles**?
 - ▶ In other words, can we order the nodes so that for each edge (v_i, v_j) , v_i is before v_j in the ordering?
 - ▶ Answer: **No!** (Why?)
- ▶ How about **directed acyclic graph (DAG)**?
 - ▶ Yes! Guarantee to have a topological ordering.
 - ▶ Why? There is always a **source node** S in a DAG. Put S first. For the graph without S , again, there is a source node. Put it next ...
- ▶ Next, we will focus on topological sorting on **DAG**.

Topological Sorting

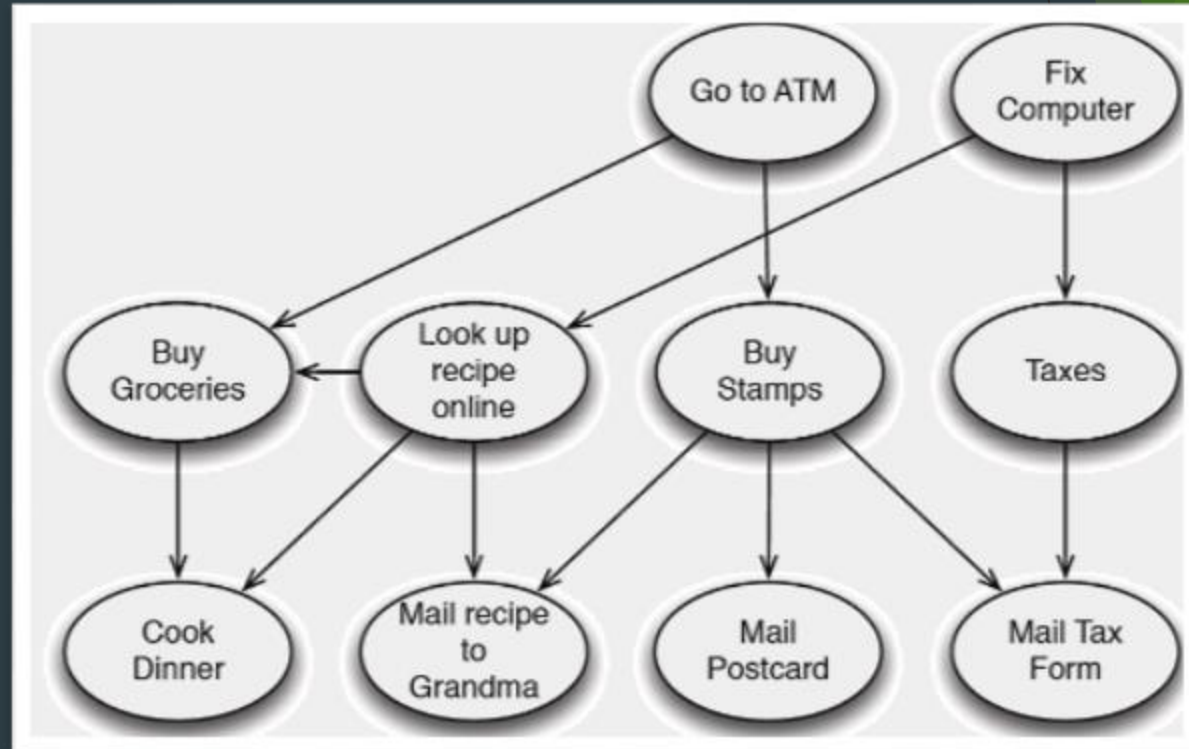


- ▶ Topological sorting is not necessarily **unique**:
 - ▶ A, G, D, B, E, C, F and A, B, G, D, E, F, C are both topological sorting.
- ▶ Are the following orderings topological sorting?
 - ▶ A, B, E, G, D, C, F
 - ▶ A, G, B, D, E, F, C

Topological Sorting Applications

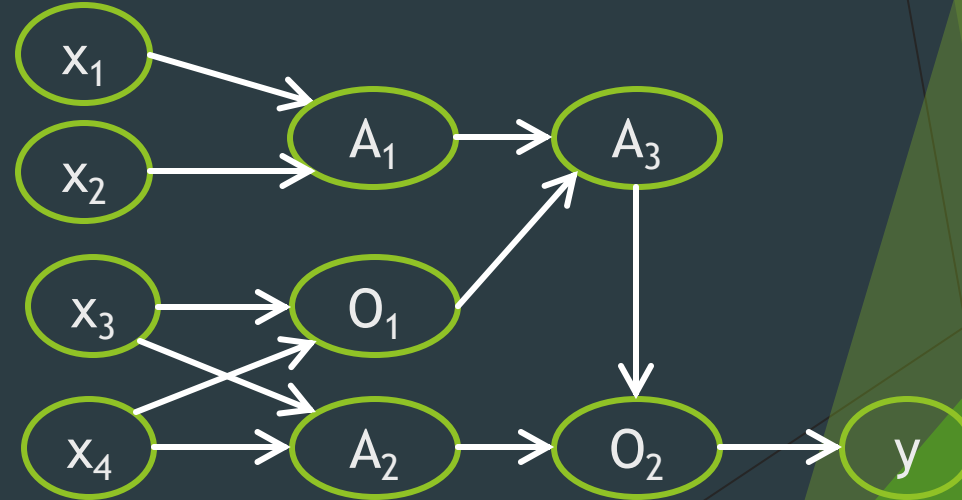
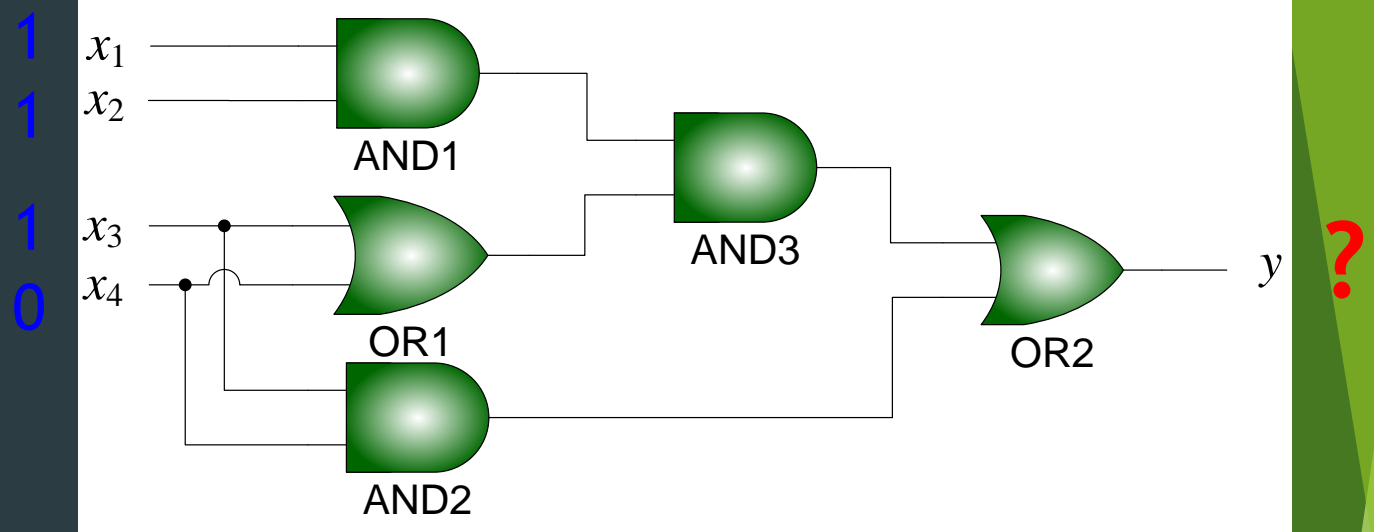
- Scheduling tasks when some tasks depend on other tasks being completed.

Serialization
Parallel → Serial
Sort out complex
dependencies



Topological Sorting Applications

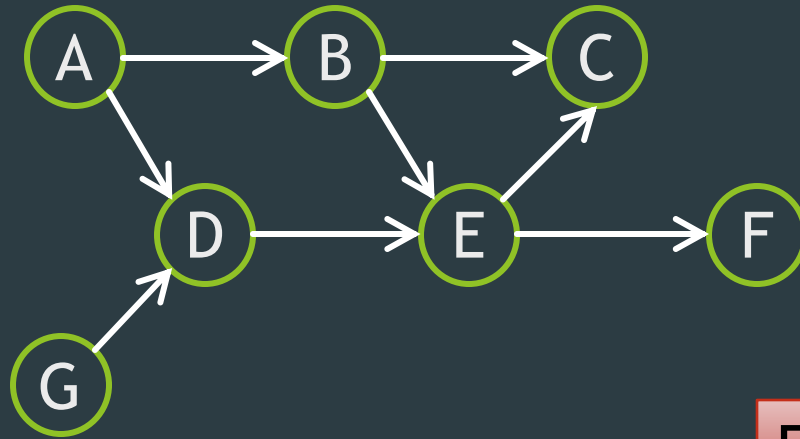
- Evaluating a combination logic circuit given a set of inputs.



Topological Sorting: Algorithm

- ▶ Based on a **queue**.
- ▶ Algorithm:
 1. Compute the in-degrees of all nodes. (**in-degree**: number of **incoming** edges of a node.)
 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, **enqueue** it into the queue.

Topological Sorting Algorithm Example



Queue

Enqueue A and G

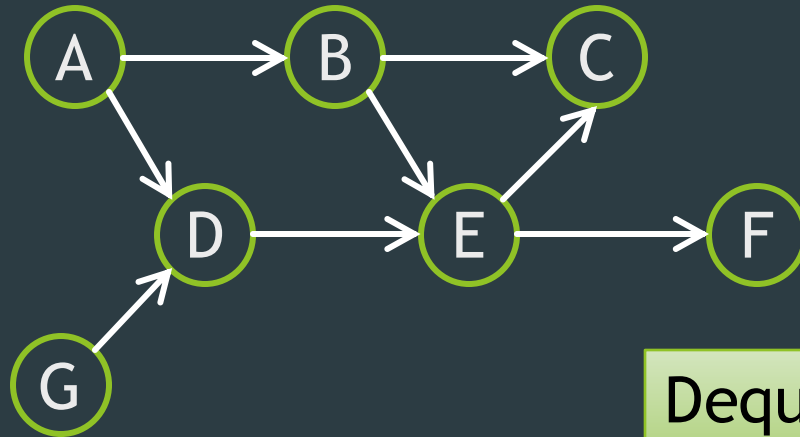
In-degrees

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 decrement in-degrees of A's neighbors.

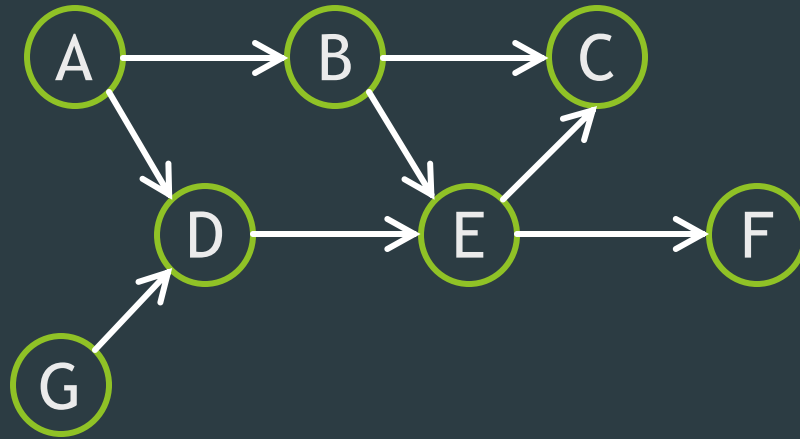
In-degrees

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

Order

--	--	--	--	--	--	--

Topological Sorting Algorithm Example



Queue

G

Enqueue B

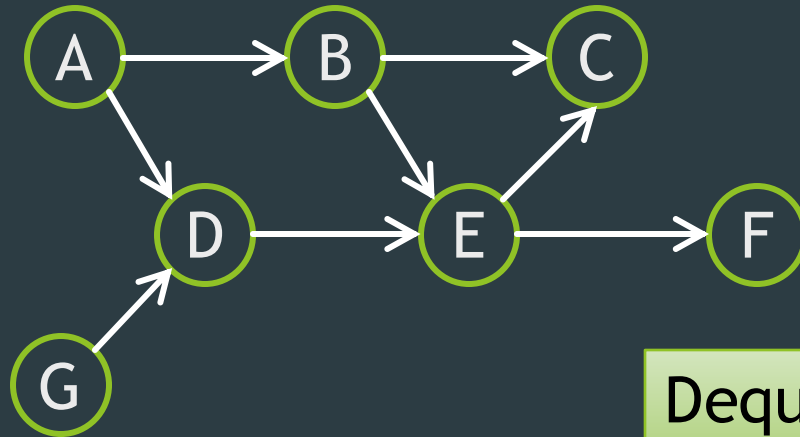
In-degrees

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 decrement in-degrees of G's neighbors.

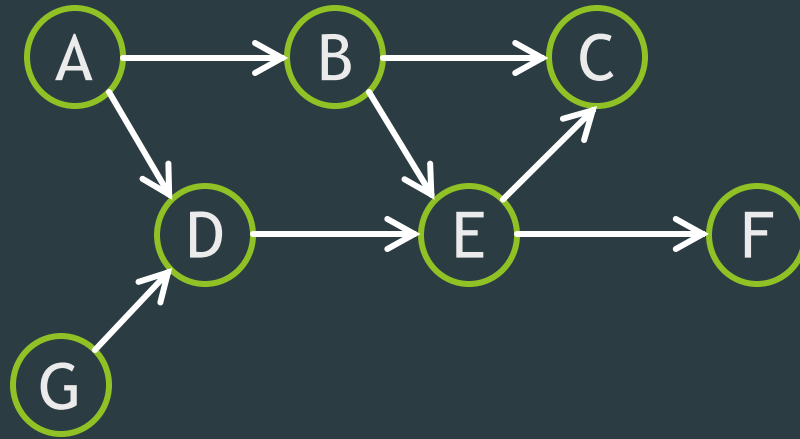
In-degrees

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

Order

A						
---	--	--	--	--	--	--

Topological Sorting Algorithm Example



Queue

B

Enqueue D

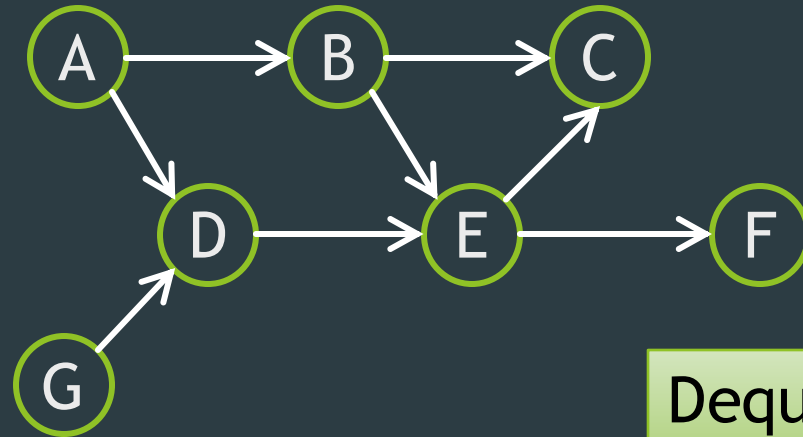
In-degrees

A	B	C	D	E	F	G
0	0	2	4 0	2	1	0

Order

A	G					
---	---	--	--	--	--	--

Topological Sorting Algorithm Example



Queue

B
D

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

In-degrees

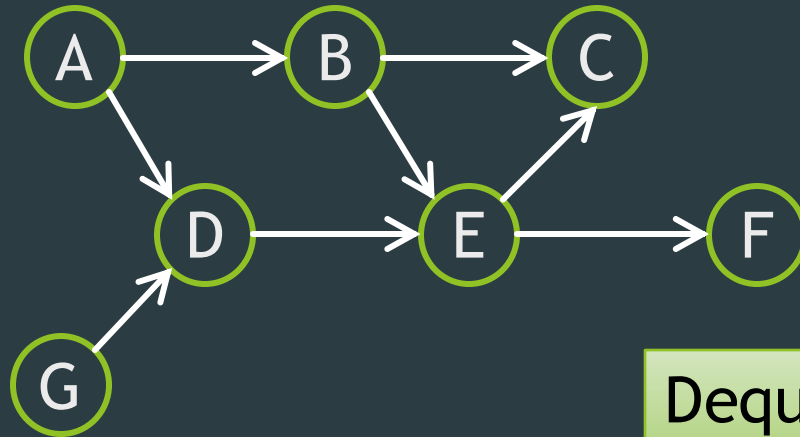
A	B	C	D	E	F	G
0	0	2	0	2	1	0

Order

A	G					
---	---	--	--	--	--	--

Topological Sorting Algorithm

Example



Queue

D

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

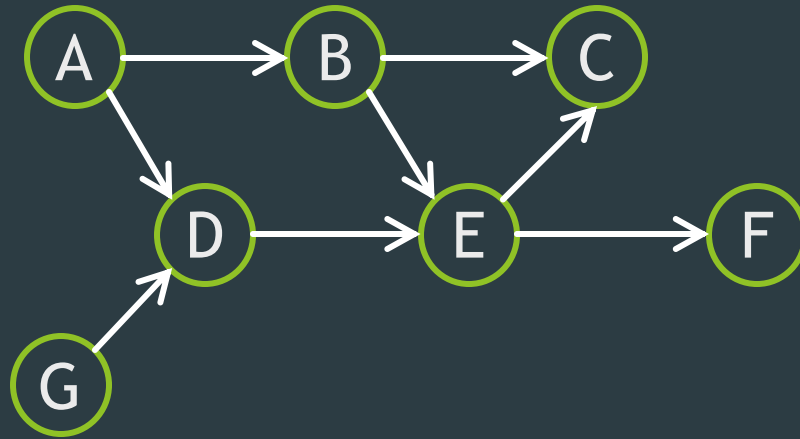
In-degrees

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

Order

A	G	B				
---	---	---	--	--	--	--

Topological Sorting Algorithm Example



Queue

Enqueue E

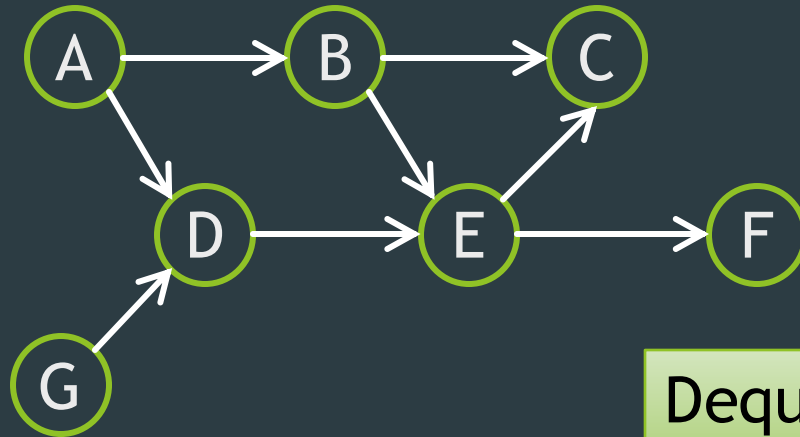
In-degrees

A	B	C	D	E	F	G
0	0	1	0	4 0	1	0

Order

A	G	B	D			
---	---	---	---	--	--	--

Topological Sorting Algorithm Example



Queue

E

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

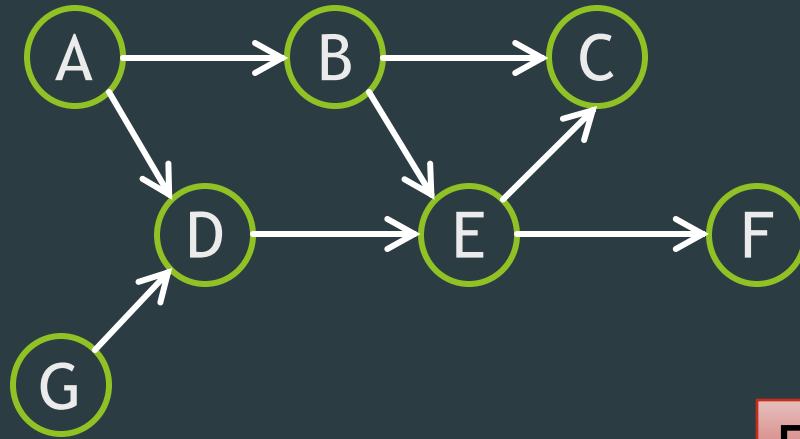
In-degrees

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

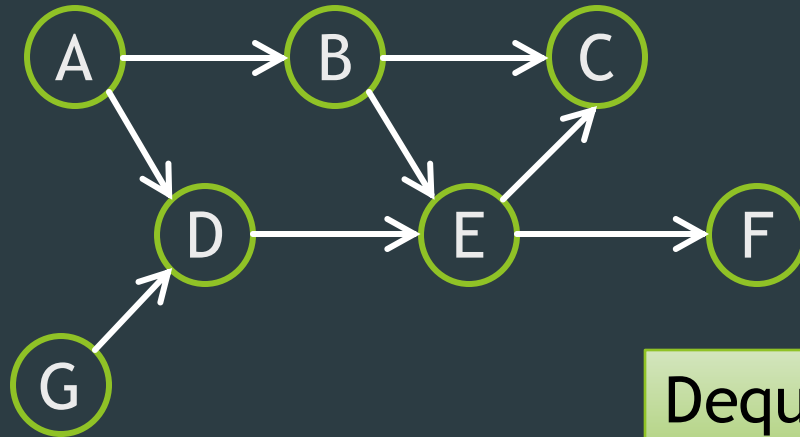
In-degrees

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 decrement in-degrees of C's neighbors.

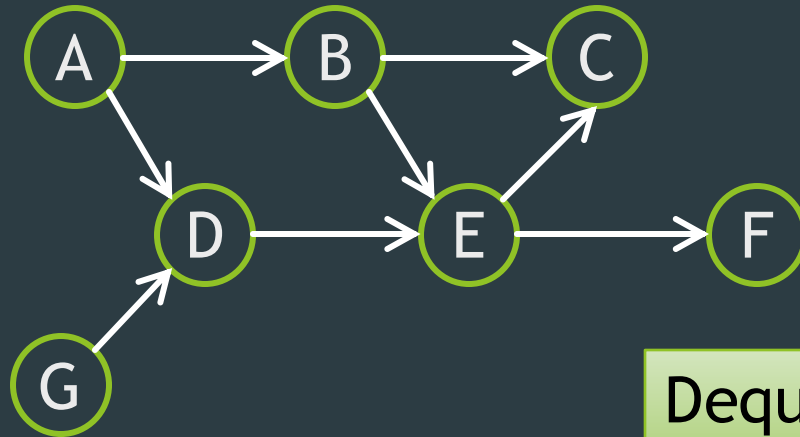
In-degrees

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 decrement in-degrees of F's neighbors.

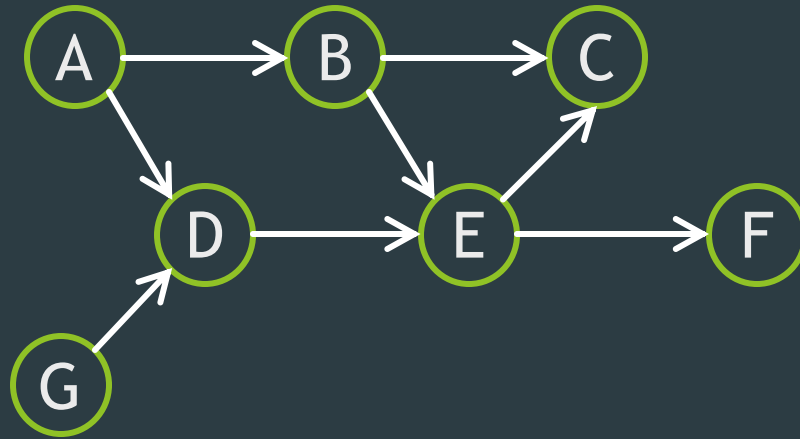
In-degrees

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 Time Complexity

Assume adjacency list representation

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.

$O(|V| + |E|)$ in total

$O(|V|)$ in total

$O(|V|)$ in total

$O(|E|)$ in total

$O(|V|)$ in total

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

Exercise 1

LeetCode: Problem 207. Course Schedule

Search

The screenshot shows the LeetCode search interface. At the top, there are topic filters: Array (1502), String (644), Hash Table (522), Dynamic Programming (471), Math (463), Sorting (349), Greedy, and an Expand button. Below these are buttons for All Topics, Algorithms, Database, pandas, JavaScript, and Shell. A search bar contains the text '997', which is highlighted by a red box and a red arrow pointing to it from the word 'Search' above. To the right of the search bar are icons for settings, a 'Pick One' button, and a 'Reset' button. Below the search bar is a table with the following columns: Status, Title, Solution, Acceptance, Difficulty, and Frequency. The table contains two rows of results.

Status	Title	Solution	Acceptance	Difficulty	Frequency
	2391. Minimum Amount of Time to Collect G...		85.6%	Medium	
	997. Find the Town Judge		49.2%	Easy	

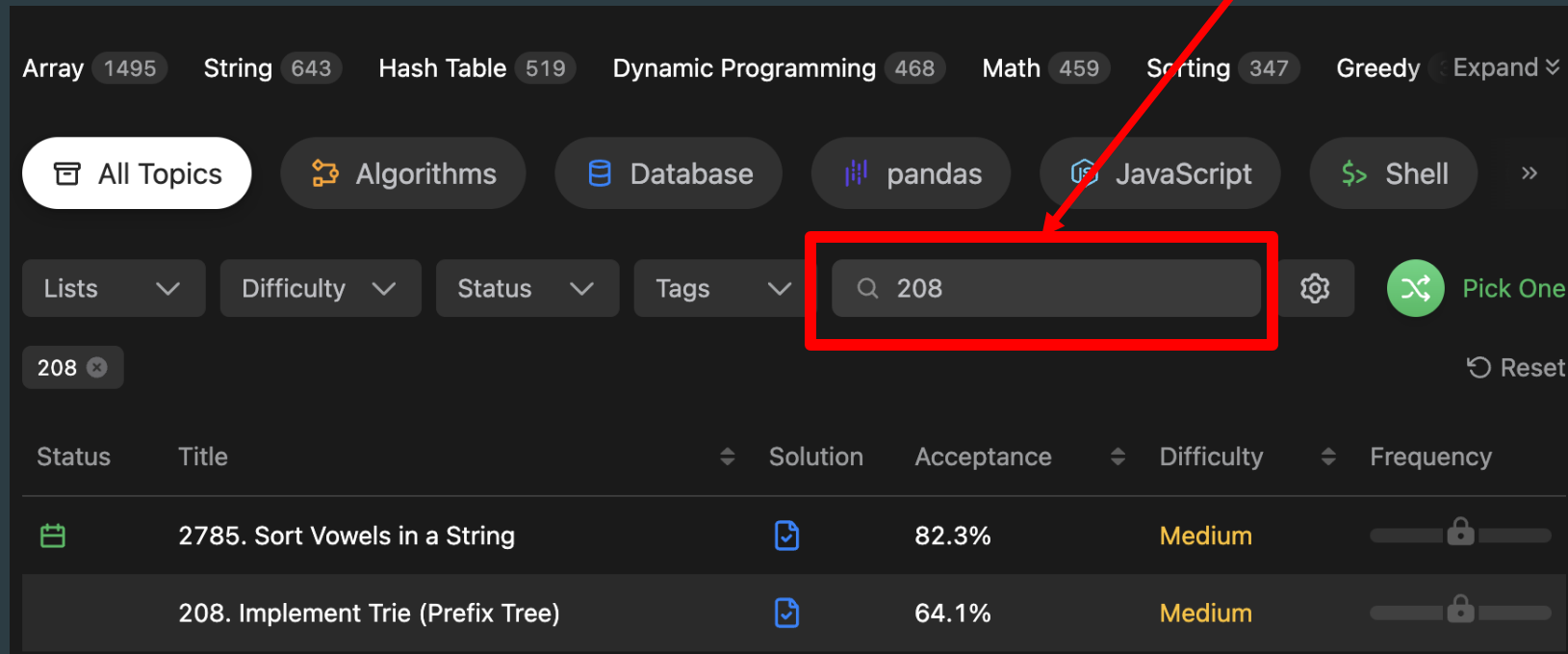
50 / page

22

Exercise 2

Problem 14. Longest Common Prefix

Search



The screenshot shows a search interface for coding problems. At the top, there are tags for various topics: Array (1495), String (643), Hash Table (519), Dynamic Programming (468), Math (459), Sorting (347), and Greedy (Expand). Below these are buttons for 'All Topics', 'Algorithms', 'Database', 'pandas', 'JavaScript', and 'Shell'. A search bar is highlighted with a red box and contains the text '208'. To the right of the search bar are buttons for 'Pick One' and 'Reset'. Below the search bar, a table lists search results. The first result is '2785. Sort Vowels in a String' with a difficulty of 'Medium' and an acceptance rate of '82.3%'. The second result is '208. Implement Trie (Prefix Tree)' with a difficulty of 'Medium' and an acceptance rate of '64.1%'. The second result is highlighted with a dark background.

Status	Title	Solution	Acceptance	Difficulty	Frequency
	2785. Sort Vowels in a String		82.3%	Medium	
	208. Implement Trie (Prefix Tree)		64.1%	Medium	

Exercise 3

Problem 720. Longest Word in Dictionary

Search

Array 1495 String 643 Hash Table 519 Dynamic Programming 468 Math 459 Sorting 347 Greedy Expand

All Topics Algorithms Database pandas JavaScript Shell >>

Lists Difficulty Status Tags 208 Pick One Reset

Status	Title	Solution	Acceptance	Difficulty	Frequency
	2785. Sort Vowels in a String		82.3%	Medium	
	208. Implement Trie (Prefix Tree)		64.1%	Medium	

Exercise 4

Problem 692. Top K Frequent Words

Search

The screenshot shows a coding platform interface with a dark theme. At the top, there are topic filters: Array (1495), String (643), Hash Table (519), Dynamic Programming (468), Math (459), Sorting (347), Greedy, and an Expand button. Below these are category buttons: All Topics, Algorithms, Database, pandas, JavaScript, and Shell. A search bar is highlighted with a red box and contains the text '208'. To the right of the search bar are buttons for 'Pick One' and 'Reset'. Below the search bar, a table displays search results. The table has columns for Status, Title, Solution, Acceptance, Difficulty, and Frequency. Two results are visible: '2785. Sort Vowels in a String' and '208. Implement Trie (Prefix Tree)'. The second result is highlighted with a dark background.

Status	Title	Solution	Acceptance	Difficulty	Frequency
	2785. Sort Vowels in a String		82.3%	Medium	
	208. Implement Trie (Prefix Tree)		64.1%	Medium	