CSE 373
Topological Sort and Graph Traversals

Winter 2015

Rama Gokhale and Megan Hopp

Topological Sort

Idea:

Given a DAG, order all the vertices so that every vertex comes before all of its neighbors

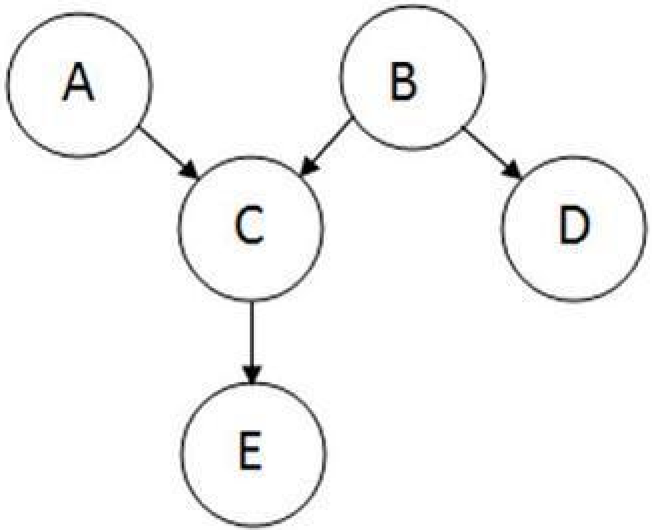
Topological Sort

- Why do we perform topological sorts only on DAGs?
- Is there always a unique answer?
- Do some DAGs have exactly 1 answer? In what case?

Topological Sort

- Why do we perform topological sorts only on DAGs?
  - Cycles mean that there is no correct answer
- Is there always a unique answer?
  - No, in some cases there could be multiple correct answers
- Do some DAGs have exactly 1 answer? In what case?
  - Yes, a list for example

Topological Sort Example

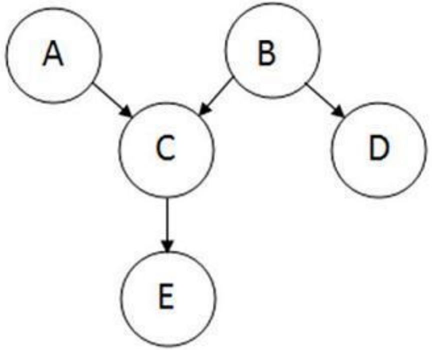


Idea:

- Keep track of the in-degree of each node.
- Use a queue to ensure the proper ordering of nodes (from least to greatest in-degree)
- Every time an in-degree is 0, enqueue it.
- Every time a node is processed, decrement it’s adjacents’ in-degree.

Topological Sort Example

Graph:

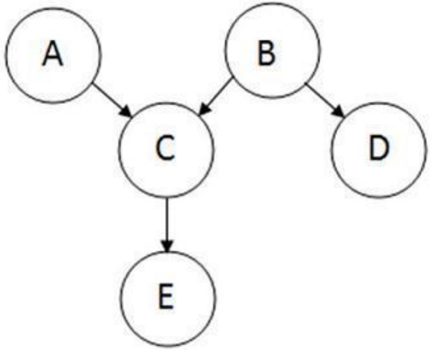


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A, B)

Initialize the in-degree array with each node's in-degree, enqueue all nodes with in-degree of 0

Topological Sort Example

Graph:

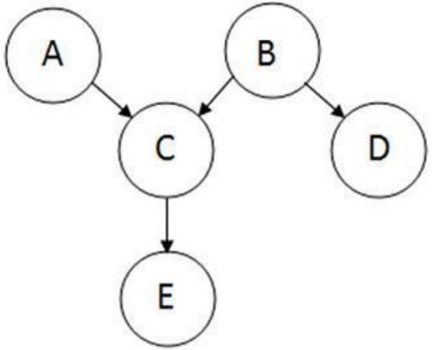


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)

Process A...

Topological Sort Example

Graph:

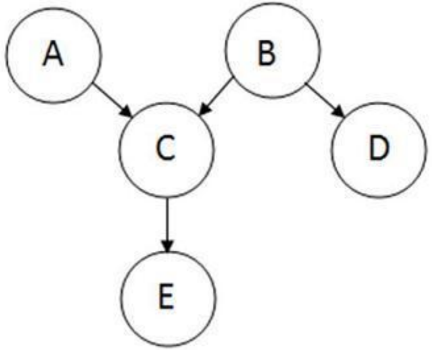


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)
Step 3:	0	0	0	0	1	(C,D)

Process B...

Topological Sort Example

Graph:

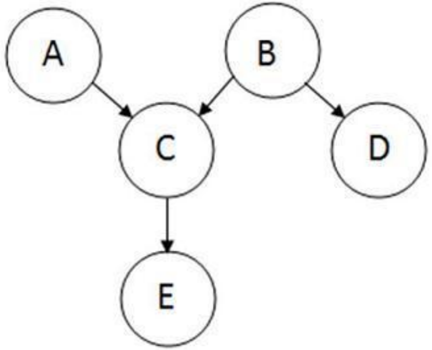


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)
Step 3:	0	0	0	0	1	(C,D)
Step 4:	0	0	0	0	0	(D,E)

Process C...

Topological Sort Example

Graph:

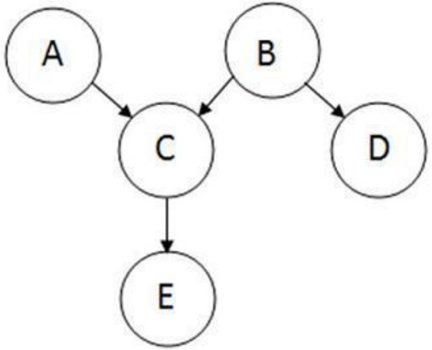


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)
Step 3:	0	0	0	0	1	(C,D)
Step 4:	0	0	0	0	0	(D,E)
Step 5:	0	0	0	0	0	(E)

Process D...

Topological Sort Example

Graph:

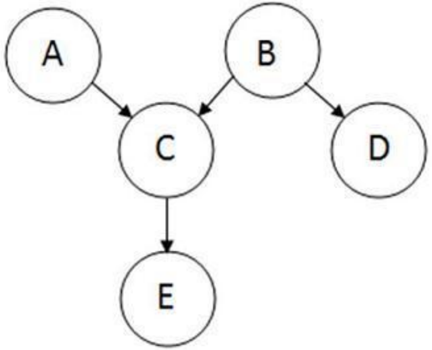


	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)
Step 3:	0	0	0	0	1	(C,D)
Step 4:	0	0	0	0	0	(D,E)
Step 5:	0	0	0	0	0	(E)
Step 6:	0	0	0	0	0	()

Process E...

Topological Sort Example

Graph:



	A	B	C	D	E	Queue contents:
Step 1:	0	0	2	1	1	(A,B)
Step 2:	0	0	1	1	1	(B)
Step 3:	0	0	0	0	1	(C,D)
Step 4:	0	0	0	0	0	(D,E)
Step 5:	0	0	0	0	0	(E)
Step 6:	0	0	0	0	0	()

Final Ordering: **A,B,C,D,E**

Running Time

- Initialization: O(|V|+|E|) (assuming adjacency list)
- Sum of all enqueues and dequeues: O(|V|)
- Sum of all decrements: O(|E|) (assuming adjacency list)

So total is **<u>O(|E| + |V|)</u>** - much better for sparse graphs

Graph Traversals

Depth-First Search:

- Recursively explore one part before going back to the other parts not yet explored
- Typically use a *stack* to keep track of which nodes to process next (non-recursive)

Breadth-First Search:

- explore areas closer to the start node first
- Typically use a *queue* to keep track of which nodes to process next

Graph Traversals

For reference:

Pseudo-code is available for DFS and BFS in the lecture slides posted on the course website.

CSE 373 HW 5

Winter 2015

Main Idea:

Comparing literary works of Shakespeare vs. Bacon to analyze word frequencies and squared error.

Using two types of HashTables to keep track of word frequencies:

- Separate Chaining Implementation
- Quadratic Probing Implementation

HashTable Implementations

Responsible for:

- constructors for each
- insert(key) -- inserting a word into the HashTable (String ‘key’ parameter), if already present in the table, just increment it’s count
- findCount(key) -- finding the word count for a given word (String ‘key’ parameter)
- getNextKey() -- used to iterate through your hashtable to retrieve the next key, should allow you to access every key in the table on subsequent calls

Homework 5 Tips

Keep in mind that you only ever care about a word AND it’s frequency. If you just have one or the other, it is useless for the analysis.

For quadratic probing, a prime table size will help reduce collisions.

Not required to make your own hash function, but you get extra credit.

Questions?