2019-02-18 Graph Representations, Topological Sort, Bellman-Ford

Sunday, February 17, 2019 4:28 PM

Agenda

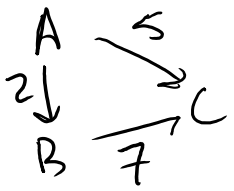
1. PA #2 check-up
2. Task: Converting an edge-list representation into an adjacency matrix.
3. Task: Converting an adjacency matrix into edge-list representation
4. Topological sort using adjacency matrix
5. Topological sort using edge list
6. Bellman-Ford algorithm

Edge List vs. Adjacency Matrix Representations

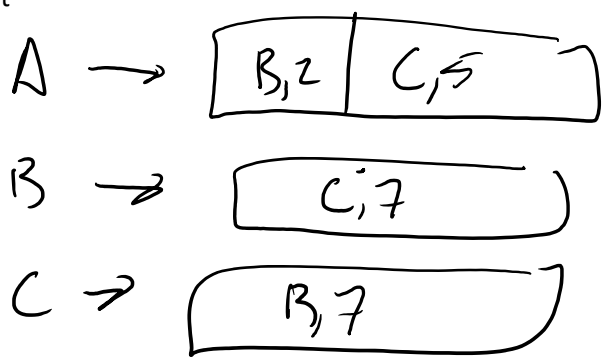
- Edge List representation is Linked-List based graph
- Adjacency Matrix is a vector-based graph

- Adjacency matrix is a 2D array

	A	B	C
A	-1	2	5
B	-1	-1	7
C	-1	7	-1

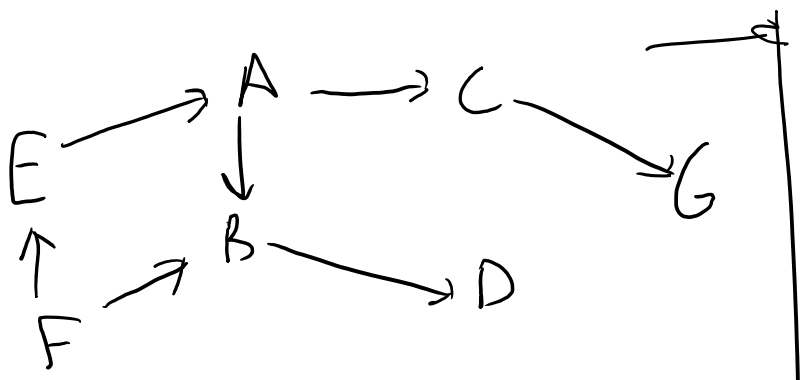


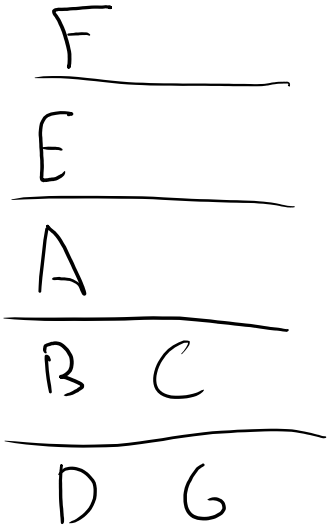
- Edge List



Topological Sort

- Idea: build a dependency graph based on order of operations
  - Think tech tree in video games, course catalog at HSU





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

Adam's original idea

- Start with all vertices having zero incoming edges
  - Practical meaning 0 on the column
- For each of these edges, zero out its row. Check to see if this creates a new zero column. If we have zero column, add to next tier.
- Repeat process until all vertices have been discovered

Ian's idea

- Record incoming edges using a frequency counter (via column order)
- For everything with a zero incoming edge, add to current tier
  - Decrement frequencies of connected edges. This may produce zero counts, which means they can get processed next.