CSCI 323 Lecture 8 Note Wednesday, 03/02/2022

Announcements:

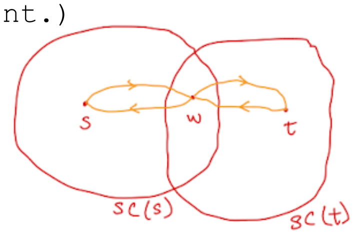
- The Lecture Recordings will be available on the following YouTube Playlists Link: https://youtube.com/playlist?list=PLZaTmV9UMKlgYpo2cAiMaEWxqyvbiXDFd

Graph

References:
Algorithm Design - Chapter 3

Directed Graph

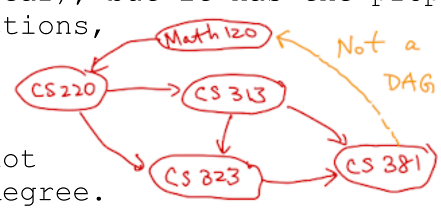
- Definition: u and v are *mutually reachable*, MR, if u has a path to v and v has a path to u.
- Lemma: If u and v are MR, and v and w are MR, then u and w are MR.
- Definition: A *strong component* of a vertex s, $SC(s)$, is a set of vertices u such that u and s are MR.
    - o How do you compute it?
        - ▪ $SC(s) = P_{from}(s) \cap P_{to}(s)$
- Theorem: For any two nodes s and t in a directed graph, either $SC(s) = SC(t)$ or $SC(s) \cap SC(t) = \emptyset$. (They are either equal or disjoint.)
    - o Proof by contradiction: Assume that $SC(s) \neq SC(t)$ and $w \in SC(s) \cap SC(t)$.
        - ▪ s and t are MR.
        - ▪ $t \in SC(s) \Leftrightarrow SC(s) = SC(t)$



DAG – Directed Acyclic Graph

- A directed graph with no directed cycles.
    - o What is an undirected graph without cycles? Tree
    - o DAG – directed version of tree. It might not look like a tree (cycle might appear), but it has the properties of tree (with the directions, there is no cycle).

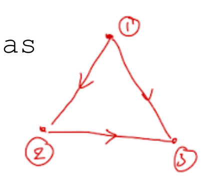
    - o For example, CS degree course must be a DAG, otherwise, student cannot start or complete the degree.



    - o DAG encode dependencies or precedence relations.

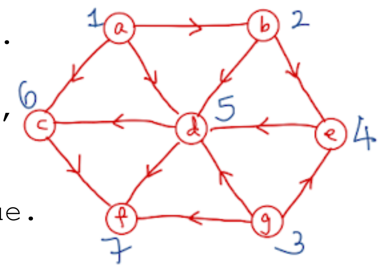
Topological Ordering:

- Topological ordering of G is an ordering of its nodes as $v_1, v_2, \ldots, v_n$ such that for every edge from v_i to v_j, (v_i, v_j), then $i < j$.

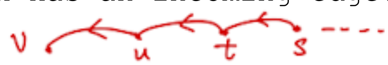


- Question: Given a DAG G,
  a) does a topological ordering always exist? → Yes!
  b) if so, how do we find it?
    - o We will give an algorithm that always outputs a topological ordering (this also implies that one always exists).
- Topological Ordering Algorithm
    - o Question: Which vertex to label 1?
      Answer: A vertex v with no incoming edges (in-degree).
        - ▪ Question: Does such a vertex always exist?
        - ▪ Theorem: In any DAG, there always exists a vertex v with no incoming edges. (Once this is proven, the algorithm is easy.)

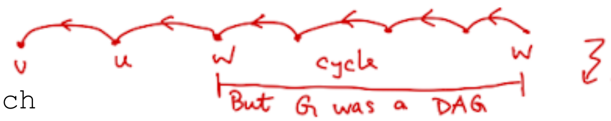
- Algorithm
  1) Find a v in G with no incoming edges. Label it 1.
  2) Delete v and all its outgoing edges from G.
  3) Recurse (find a u in $G-v$ with no incoming edges, label it 2 and so on)



    - Removing a vertex from a DAG is still a DAG.
    - The result of this algorithm might not be unique.
    - If implemented naively, it will run $O(n^2)$ time.
      - n iterations, search for no incoming edges, vertex $O(n)$.
      - but if implemented better (page 103, active nodes) make it runs in $O(m+n)$ time.
- Theorem: In any DAG, there always exists a vertex v with no incoming edges.
    - Proof by contradiction:
      - Assume every vertex has an incoming edge.
      - Pick any vertex v.



      - but G has only n vertices, so this backtracking must repeat a vertex at some point.
      - Then there must be a (directed) cycle, which contradict that G is a DAG, which cannot have directed cycles.



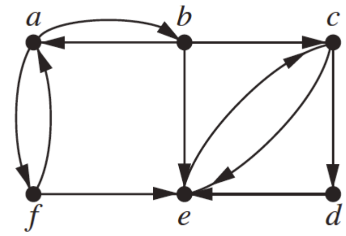
What to expect or prepare for the next class:
- Greedy Algorithm
- Interval Scheduling

Reading Assignment

Algorithm Design: 3.5 – 3.6

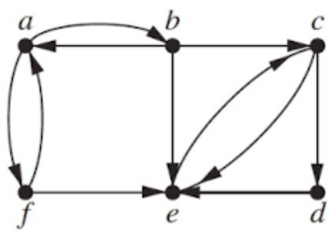
Suggested Problems
- Algorithm Design – Chapter 3 # 1, 2, 6

Project on Graph [8 %]

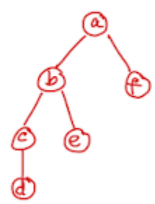


- Write a program that will return all the strong components in the directly graph.
    1) input the graph
        - Adjacency List or Adjacency Matrix
    2) Implement both DFS and BFS to traverse the graph.
        - Use one to find $P_{from}(v)$ and the other one for $P_{to}(v)$
    3) Use $P_{from}(v)$ and $P_{to}(v)$ to find the strong component of v.
    4) Return all the strong component in the directed graph.

- ❑ Submit your work via email, subject "CS323 Graph".
- ❑ What's the run time of your algorithm with input size n?
- ❑ Schedule a time to explain your work.
- ❑ Due Sunday, March 13, 2022.



BFS (a)



DFS (a)

$P_{from}(a) = \{a, b, f, c, e, d\}$

a and b are MR.

a and f are MR.

→ b and f are MR

BFS (a, G^{rev})

DFS (a, G^{rev})

$P_{to}(a) = \{a, b, f\}$

$\{\{a, b, f,\}, \{c, d, e\}\}$

$SC(a) = P_{from}(a) \cap P_{to}(a) = \{a, b, f\}$