Class Test

INSTRUCTIONS:

Please have sheets ready to answer the questions.

The slides are marked with time limits to encourage you to complete the set of questions within the end time stamp of the slide

Each slide has a new set of questions. The slide changes to the next slide(next set of questions) at the finish time of each slide.

Feel free to copy down the questions if you are not certain of completing it before the end time stamp of the slide. Else writing only the answers will be sufficient.

All questions are compulsory. Required definitions will be introduced as the test proceeds.

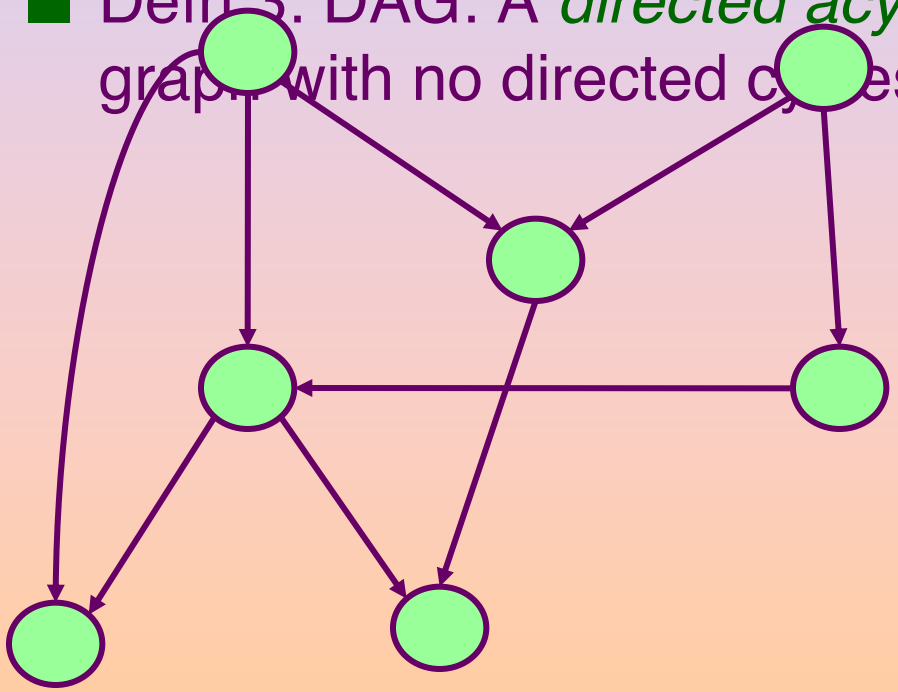
- **Cass Test -Topological Sort**
- The quiz slide contains the solution. Congratulations to each of you as you
  have taught yourself the topic "Topological sort" in the test today.

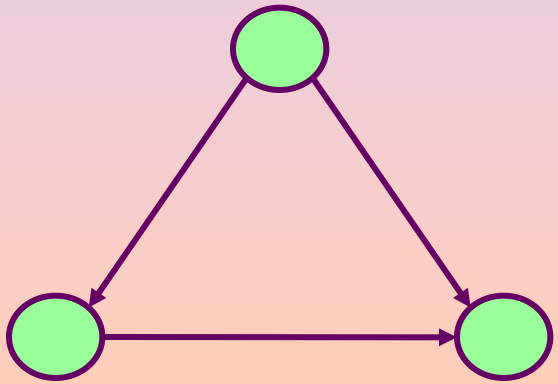
  The quiz leads you to understand why the finishing time stamps help you
  do a topological sort.
  Make sure you also read the additional notes at the end of the test slides.

  Note: Only question 1 and 2 of today's quiz will be graded. The rest were to help
  you understand topological sort which is part of your Mid2 syllabus. The tutorials this week will handle any questions/issues related to the quiz(all five questions).

Definitions

- Defn 1: Indegree : The number of incoming edges into a node
- Defn 2: Outdegree: The number of outgoing edges from a node
- Defn 3: DAG: A *directed acyclic graph(DAG)* is a directed graph with no directed cycles:





End time stamp: 10:30am

1. Given a graph G, the edges selected when doing a DFS traversal forms which of the below;
   (a) connected subgraph of G (b) disconnected subgraph of G (c) tree (d) forest
   Ans (d) forest

   Can not be connected/disconnected subgraph as a connected/disconnected subgraph can contain cycles.

Can not be a tree as a DFS can lead to several trees if there are white nodes left after the DFS at the starting node.
DFS gives us a set of trees (one or more).

2. True/False:

A . In a graph, The node with the smallest start time will also have the largest finish time? FALSE. Take the case of a graph where after the DFS at some node you are still left with white nodes. In such a case, the node with largest finish time lies in a different tree than the node with smallest start time.

B. In a graph, the node with the largest finish time will always have indegree 0 (no incoming edges)? FALSE. Take the case where a graph has a directed cycle of three nodes a, b, c. The node with largest finish time is the node you started with but every node in this graph has indegree 1.

C. In a DAG(directed acyclic graph), the node with the largest finish time will always have indegree 0 (no incoming edges)? TRUE. Proving by contradiction. Suppose not, then node with largest finish time has indegree>0. Let this node be x. This implies that x has an incoming edge from some node y. Since x has the largest finishing time, y is black (finished) before x became black at the end.

Thus, f(y) < f(x)..... 1

However, when y was grey just before it turned black, y did not find find x as black as otherwise y would have explored a dfs from its neighbor x. So x was white or grey.

-If x is white then f(x) < f(y) contraditing 1.

- If x is grey it means x is ancestor of y in the dfs tree and now there is an edge from y to x as well that forms a cycle. This contradicts that its a acyclic graph.

D. In A DAG, there is at least one node with indegree 0 and at least one node with outdegree 0. TRUE.

In 2C we have proved that in a DAG, the node with largest finish time with have indegree 0. Since on doing a DFS you will have a node with largest finish time, you can conclude that in a DAG you always have a node with indegree 0.

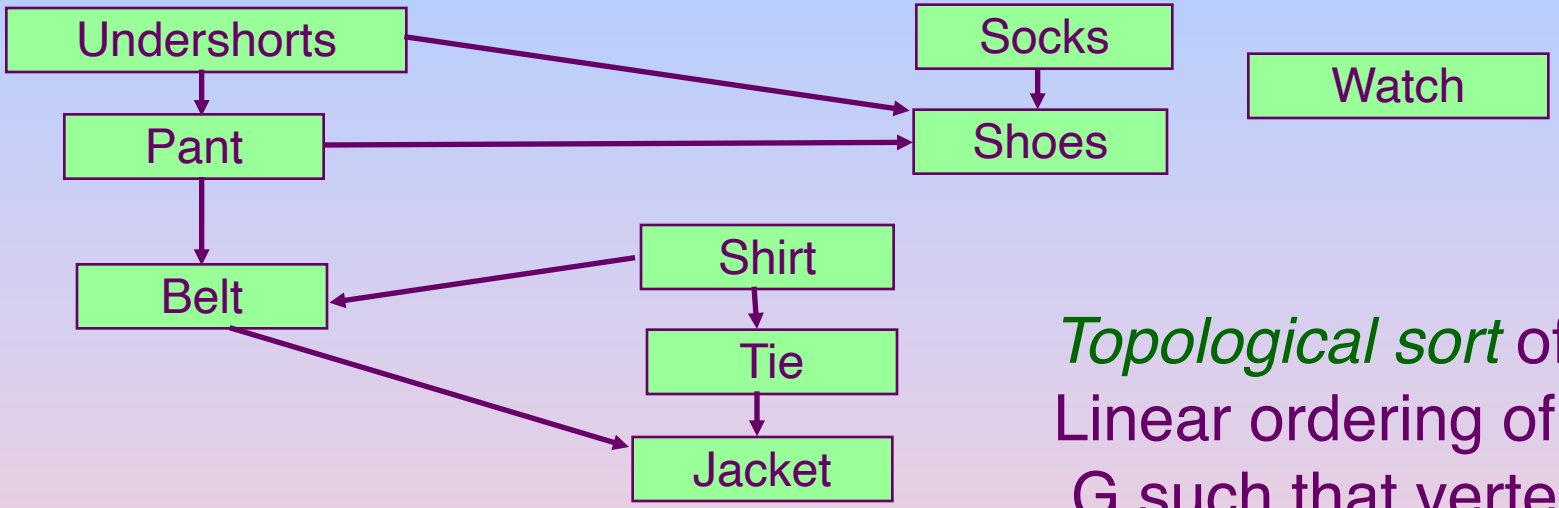
Now lets prove that there is at least one node with outdegree 0. To see this, choose any node. Select any outgoing edge and follow it to another node. Repeating this process, we will never repeat a node, because the graph has no cycles, so after V-1 iterations (where IVI is the number of nodes in the graph), this process must terminate in a node with no outdegree.

3. Prove that :
If there is a path from a node u to another node v in a DAG, then f(u) > f(v).
Note that there might not be a path from u to v in the DFS forest but v might be reachable from some other node x in the DFS forest.
f(u), f(v) indicate the finish time of node u and v respectively.

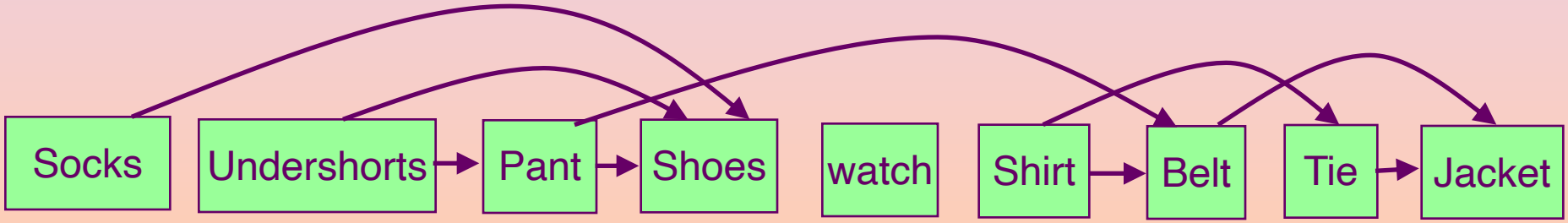
There are two cases
1. S(u) > S(v) 2. S(u) < S(v)
Case 1. If s(u) >s(v) then node v was seen through the DFS started on some node x and v was visited before node u was visited. Being a DAG there can not be a path from v to u as well(acyclic). This implies that u can be visited only after v finishes. f(v) < s(u) <f(u) .

Case 2: If s(u) < s(v). Since there is a path from u to v, the node v is white when the DFS traversal reaches v via u. Thus f(u) > f(v).

Topological Sort



Topological sort of a DAG:
Linear ordering of all vertice
G such that vertex *u* comes
vertex *v* if edge (*u*, *v*) ∈ G



- Topological sort was defined as “Linear ordering of all vertices in graph G such that vertex *u* comes before vertex *v* if edge $(u, v) \in G$”

4. True/False : (10mins)

A. Topological sorting definition above is same as ordering all the vertices in graph G such that vertex *u* comes before vertex *v* if there is a path from u to v in G.

TRUE. Let the path from u to v be u->a_1->a_2->…a_k->v. Proof by contradiction. Suppose not, then u comes after v though there is a path from u to v. However, a_1 has to occur after u in the sort by the definition given and a_2 has to occur after a_1. That is, any node a_i has to occur after a_{i-1} in the sorted order since there is a edge from a_{i-1} to a_i. Thus a_k occurs after u in the sort. Node v can occur only after a_k and hence only after u in the sort. Contradiction!

B. There can exist a DAG for which there are several topological sorted outputs.

TRUE. Give another topological sorted form of the example in the previous slide

5. Propose an algorithm that can produce a topological sorted sequence of the nodes in a DAG. (20mins)
(Hint:Use the statement in question 3 which said "If there is a path from a node u to another node v in a DAG, then f(u) > f(v)".)

Using question 3 you have shown that f(u) >f(v) is there is a path from u to v. In a topological sorted sequence you would want to output u before v is there is a path from u to v. We hence run a DFS and find the finish times of all the nodes. The nodes are then arranged in decreasing order of finish times.

```
Topological-Sort()
{
    1. Call DFS to compute finish time f[v] for each vertex
    2. As each vertex is finished, insert it onto the front of a linked
       list
    3. Return the linked list of vertices
}
```

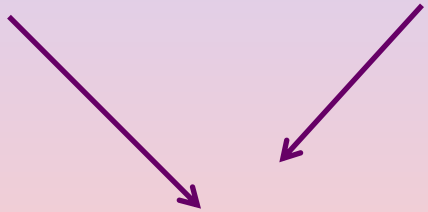
- Time: O(V+E)

Additional Slides

- Why not use sorted starting times to find the topological sort?

Counterexample!

Consider a triangle A 1/6 B 4/5



C 2/3

Kahn's algorithms

- There could be many other alternate ways to find the topological sorted form of a DAG. Another popular alternative uses a BFS like appraoch called the Kahns algorithm. The idea is to first output the nodes with indegree 0 in any order. Remove them and their outgoing edges. We now have a new set of nodes with indegree 0 which are the children of the earlier nodes. Output them in any order and remove them. This goes on.

Kahns Algorithm

- You maintain an AdjList along with the current indegree of each node.
- Keep a queue which is the set of nodes with indegree 0.
- Each time you pick a node whose indegree is 0, dequeue it, reduce indegree of all nodes adjancent to it. While reducing the indegree if any node indegree turns 0 add it to the queue.