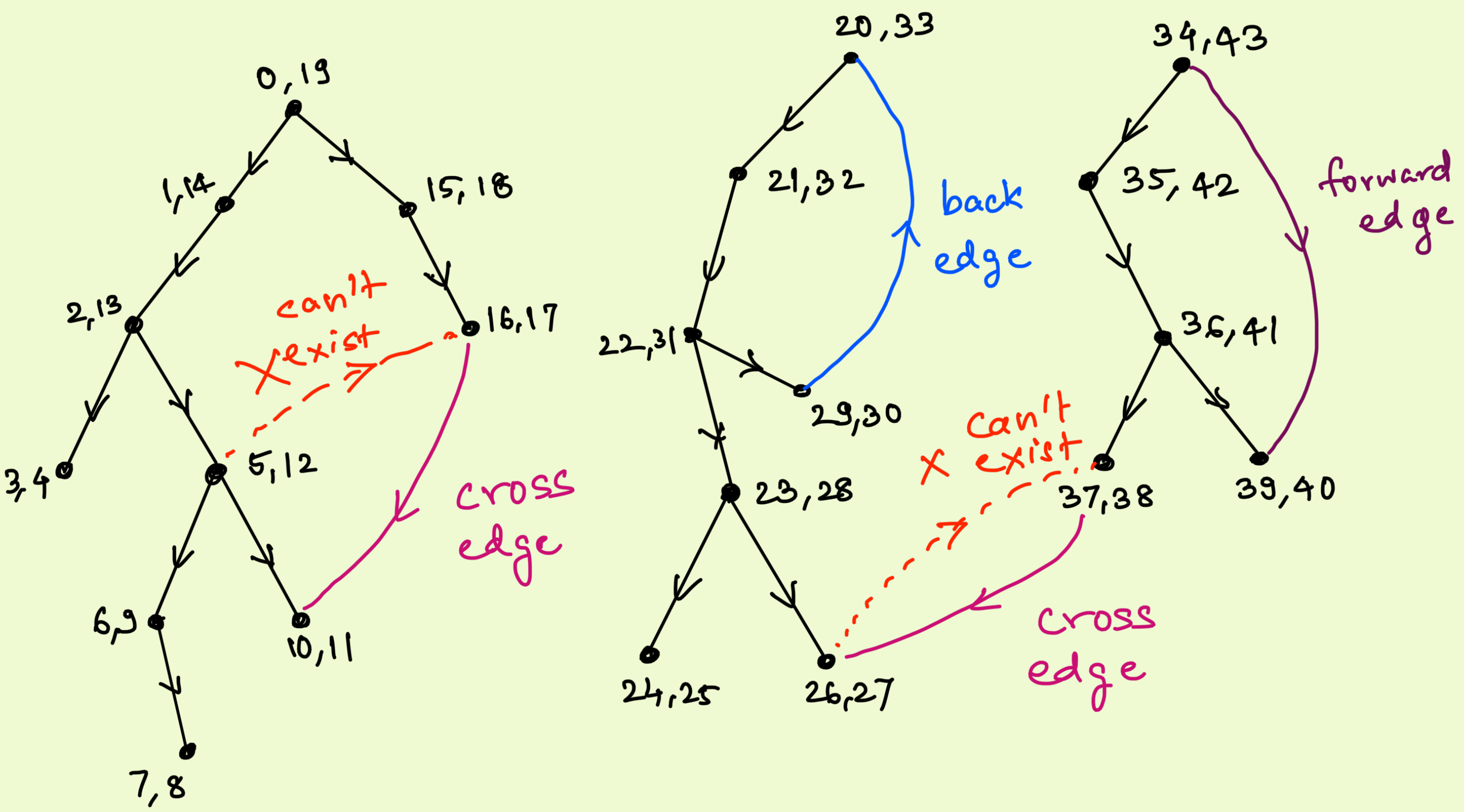
COL 106 Lecture 39

Topic : Testing Acyclicity
Topological Sort

Recap: Global DFT of directed graphs



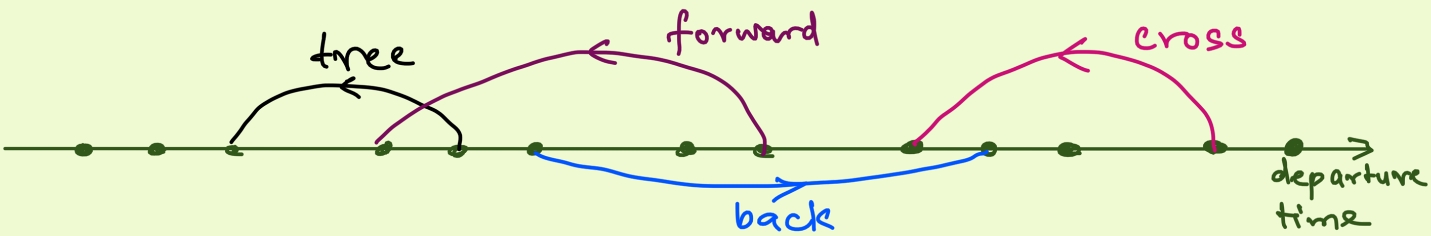
Claim: Consider a DFT of a directed graph and the resulting arrival and departure times of vertices. Suppose (u,v) is an edge.

1. If (u,v) is a tree edge
2. If (u,v) is a forward edge

then $arr[u] < arr[v] < dep[v] < dep[u]$.

3. If (u,v) is a back edge, then $arr[v] < arr[u] < dep[u] < dep[v]$.

4. If (u,v) is a cross edge, then $arr[v] < dep[v] < arr[u] < dep[u]$.



Testing Acyclicity of Directed Graphs

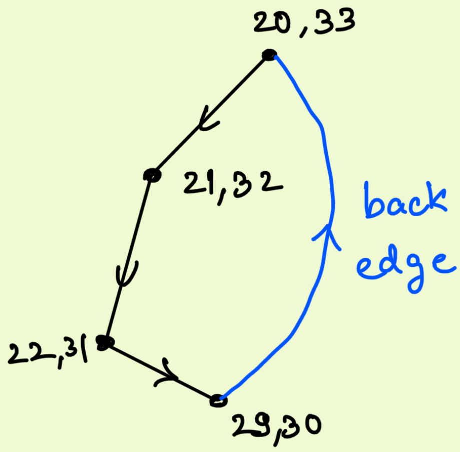
isAcyclic (G) :

1. Perform global DFT of G.
2. If G has a back edge:

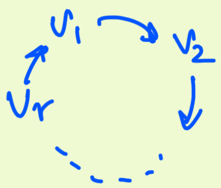
        return FALSE → Back edge + some tree edges form a directed cycle

    Else:

        return TRUE



If no back edge, but if

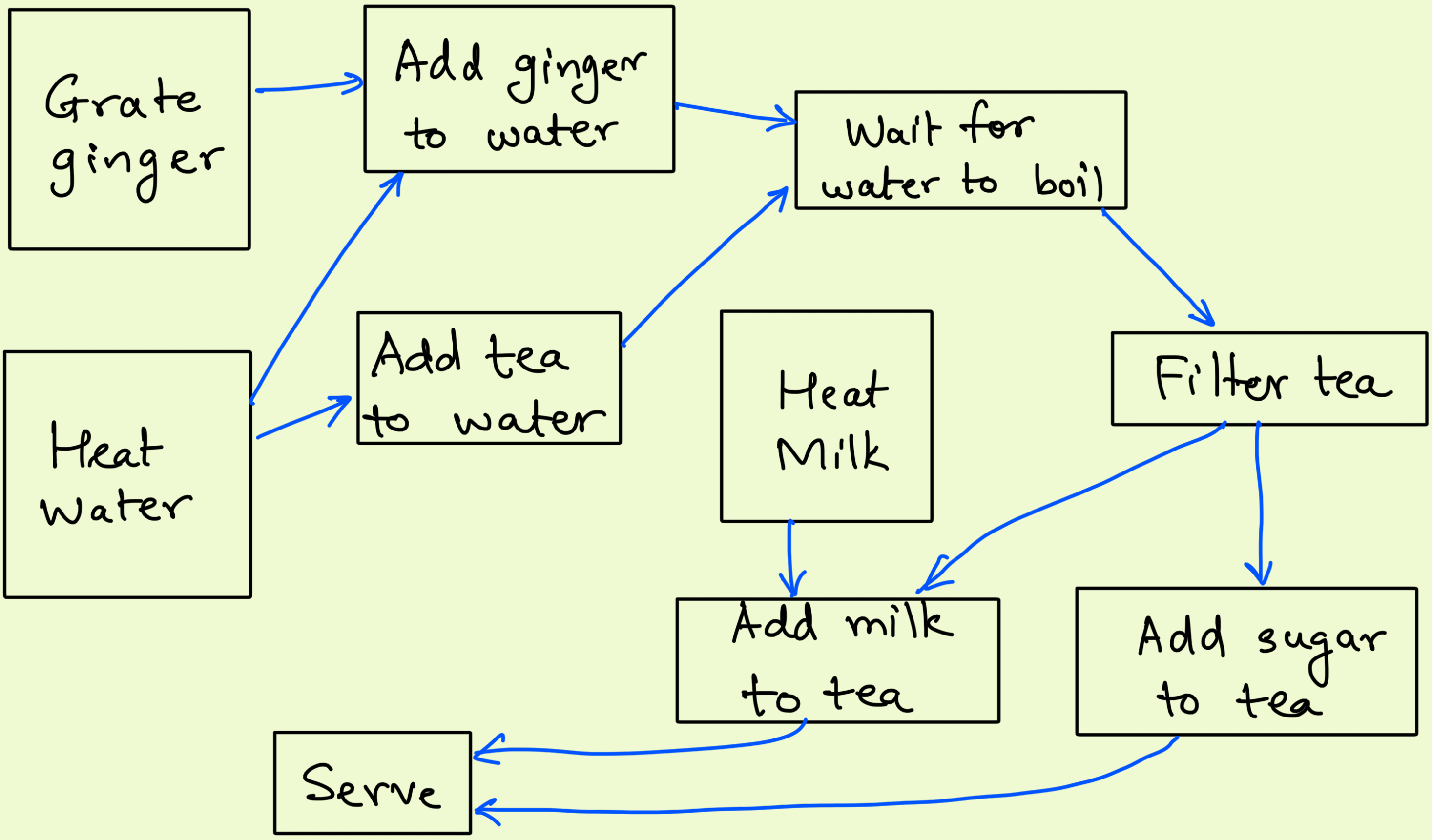


is a directed cycle, then

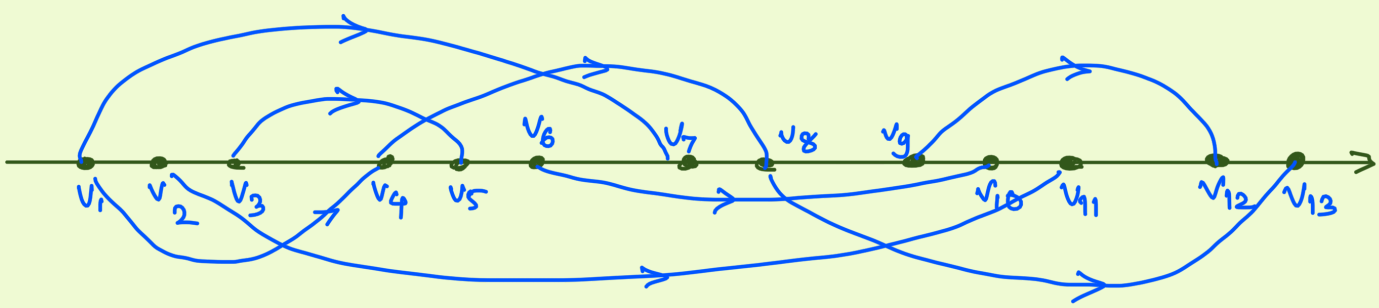
$dep[v_1] > dep[v_2] > \ldots > dep[v_r] > dep[v_1]$.

This is a contradiction.

Topological Sort : Motivation



Definition : A topological sort of a DAG $G=(V,E)$ is an ordering $v_1, v_2, \ldots, v_n$ of the vertex set V such that each edge is of the form (v_i, v_j) for some $i<j$.



Observation: In a DAG, the descending order of vertices by departure time (assigned by some DFT) is a topological sort.

find Top Sort ( )
~~DFT ()~~
visited [v] ← FALSE
~~arr [v] ← -1, dep [v] ← -1~~ ∀ vertices v
topSort ← [ ]
For each vertex v:
If ~~arr [v] = -1~~ not visited [v] then
~~DFT_rec (v)~~
find Top Sort_rec (v)
return reverse (topSort).

find Top Sort_rec (v)
~~DFT_rec (v):~~
~~arr [v] ← count~~ visited [v] ← TRUE
~~count ← count +1.~~
For each out-neighbor u of v:
If ~~arr [u] = -1:~~ not visited [u] .
~~prev [u] ← v~~
~~DFT_rec (u)~~ find Top Sort_rec (u)
~~dep [v] ← count~~
~~count ← count + 1~~
Append v to topSort

Running time : $O(n+m)$ n: # vertices , m: # edges