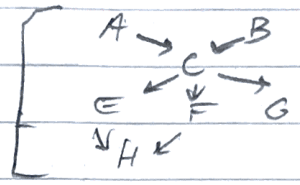
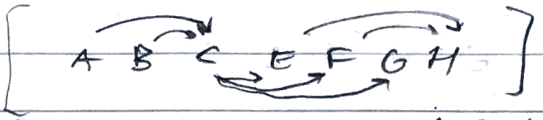
CSC263 Friday Tutorial

- Topological Sort
  - given non-linear ordered graph of nodes:
  - find linear order of nodes:





notice constant: arrows left to right, means ordering meets dependancies needs

- topological sort order not unique
- not all graphs have topological sort order

- DAG - directed acyclic graph
  - topological sort requires DAG
  - cycle in topological sort results in backwards angle

Ex - SEE: phone picture of complicated course dependancy graph

TOPOLOGICAL SORT(G)
call DFS(G)
as each vertex is finished
insert it into the front of a linked list
return the linked list

- Intuitively: last thing finished means a lot of things depend on it
- imagine: pulling the dependancy list out

Ex - Do DFS, get finishing times, make reverse order list (PIC #2)

- topological sort order depends on node explore order
- there may be multiple orderings that are acceptable

Theorem
- TOPOLOGICAL-SORT(G) produces a topological sort order of DAG $G=(V,E)$

Claim
- $[\forall (u,v) \in V,\ f[u] > f[v]]$

Proof
- DFS classifies (u,v) as either (i) tree edge, (ii) forward edge, or (iii) cross edge, (not backwards edge, no cycle).
  - (i, ii) v descendant of u, $\therefore f[u] > f[v]$
  - (iii) either $(d(v) < f(v) < d(u) < f(u))$ or $(d(u) < f(u) < d(v) < f(v))$
    We know "directed" edge $u \to v$, means v finished first.
    Then we know that $f[u] > f[v]$
- Given claim, via topological sort also, v will occur before v in the list.
- Then we can guarantee that list is topologically sorted.

TODO
- SEE CLRS Topological sort section!