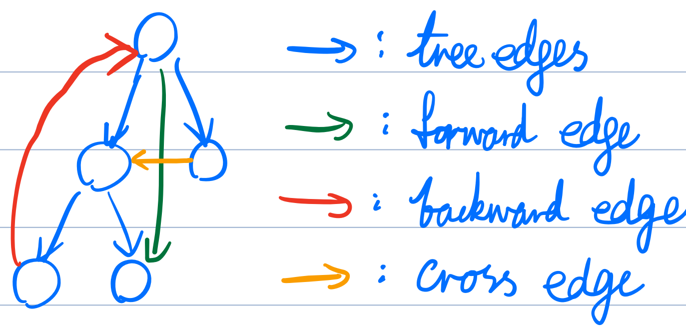
Recap:

Edge Classification

- Tree Edge: Visit new vertex via edge (i.e. vertex was white)
- Forward Edge: node → descendant in tree
- Backward Edge: node → ancestor in tree
- Cross Edge: between two non-ancestor related vertices



How do we detect type of edge (u, v)?

- Tree Edge: v was white when we explored this edge
- Forward Edge: - - - green - - - - - -
- Backward Edge: - - - - yellow - - - -
- Cross Edge - - - green - - - -

Cross Edge vs. Forward Edge?

Add discovery time (disc) and finish time (finish).

- Forward Edge: v was green & $disc(v) > disc(u)$
- Cross Edge: v was green & $disc(v) < disc(u)$ (and also $finish(v) < disc(u)$)

Undirected Graphs:

Thm: In DFS of an undirected graph G, every edge is either a tree edge or a backward edge, i.e. when we first explore $\{u,v\} \in E$ from u, v cannot be green.

Corollary: An undirected graph is acyclic iff there are no backward edges.

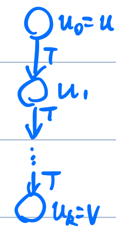
1. Edge Classification (cont'd)

Directed Graphs:

<u>Thm</u> (White-Path Theorem): v is a descendent of u in the DFS forest iff at time $disc(u)$, there exists a path from u to v with only white vertices.

<u>Proof Sketch</u>:

"$\Rightarrow$": Let $u_0 = u, u_1, u_2, \ldots, u_k = v$ be a path in the DFS forest (all edges (u_i, u_{i+1}) are tree edges). Then u_1 is discovered from u_0, u_2 from u_1, etc. Therefore,



$$disc(u_k) > disc(u_{k-1}) > \cdots > disc(u_1) > disc(u_0)$$

So at time $disc(u_0) = disc(u)$, $u_1, u_2, \ldots, u_k$ are all white.

"$\Leftarrow$": At time $disc(u_0) = disc(u)$, $u_1, \ldots, u_k$ are all white.

We have $disc(u_k) > disc(u_{k-1}) > \cdots > disc(u_1) > disc(u_0)$

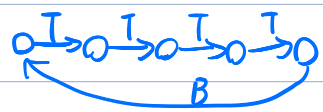
$$finish(u_k) < finish(u_{k-1}) < \ldots < finish(u_1) < finish(u_0)$$

Therefore, $u_k = v$ is discovered during the visit from $u_0 = u$, and there must a path from u to v in the DFS forest.

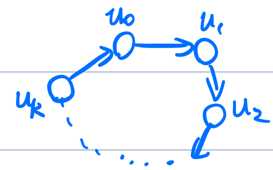
#

Corollary: A <u>directed</u> graph is acyclic iff DFS finds no backward edges.

Proof: We prove a directed graph is cyclic iff DFS finds a backward edge.

"$\Leftarrow$": 

Any backward edge immediately gives a cycle.

"$\Rightarrow$": Assume there is a cycle. 

Let u_0 be the first vertex discovered in the cycle. So at disc(u_0), $u_1, u_2, \dots, u_k$ are all white. By white-path thm, u_k is descendent of u_0.
Then the edge (u_k, u_0) is a backward edge. #

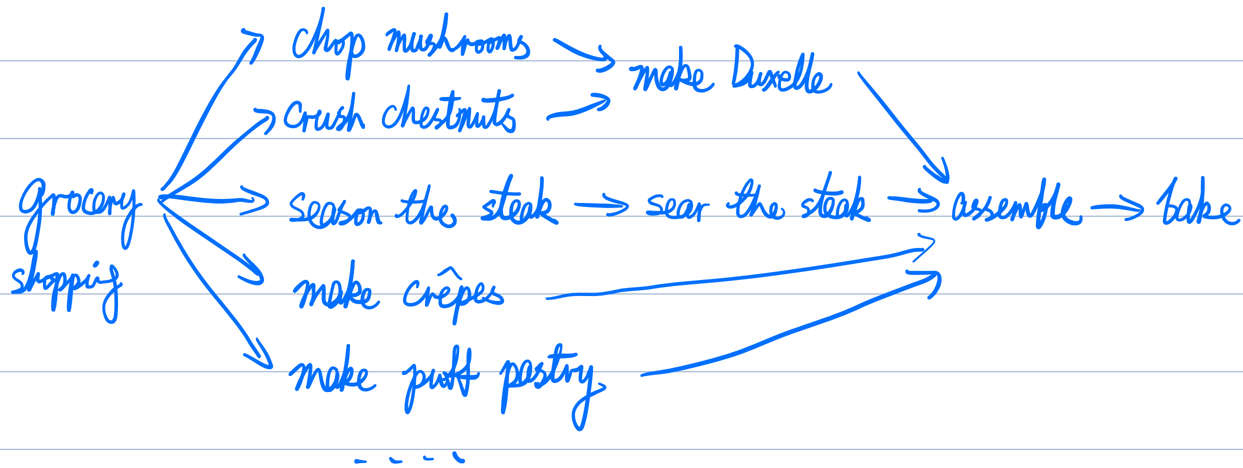
$\boxed{\text{Cycle detection? Run DFS and look for a backward edge!}}$

2. Topological Sort

Imagine scheduling a list of tasks, and some tasks must be completed before others.

We can represent such dependencies as a graph!
The name comes from the graph being a "topology".

e.g. to cook Beef Wellington



Problematic if we have a cycle... So we have a DAG.

<u>Algorithm:</u>

Input: DAG $G=(V,E)$

1. Run DFS(G)
2. Output vertices in descending order of <u>finish</u> time

<u>Runtime:</u> $\Theta(|V|+|E|)$. Don't need sorting for step 2. We just add vertices to the <u>front</u> of a linked list when they finish.

<u>Correctness:</u> The alg. above outputs a topological Sort, i.e. a node is output only after all its parents/dependencies are output.

Proof: We show that for all $(u,v) \in E$, $finish(u) > finish(v)$.

$$\underset{u}{\bigcirc} \longrightarrow \underset{v}{\bigcirc}$$

Note when we explore (u,v), v cannot be yellow (no backward edges in a DAG). So v can only be white / green.

- If green, $finish(v)$ already set, but we're still exploring u, so $finish(u) > finish(v)$.
- If white, we will recursively visit v. When recursively call returns, we set $finish(v)$, but not $finish(u)$. So $finish(u) > finish(v)$.

#

$$\boxed{\text{If you see DAG, think topological sort!}}$$

Alternative way to do topo. sort:

A <u>source</u> in a DAG is a vertex w/o incoming edges.

Each DAG must have a source! (why?)

Algorithm:

While there are still vertices left:

Output arbitrary source s,

and remove s and edges out of s from the graph

Correctness: ✓. When we output a vertex v, it is a source, meaning all its parents have been output & removed.

Runtime: Maintain in-degree for each vertex, and a list of sources. $\Theta(|V|+|E|)$