- DFS and type of edges

**DFS(s)**

- DFS (a)
- DFS (b)
- DFS (c)
- DFS (d)

```
S   a   b   c   c   a   d   d   s
enter enter  enter leave enter leave leave
```

- Pre-order: S a b c d
- Post-order: b c a d s

```
X
a   b   a   b
enter enter leave leave
```

- Type of edges

1. Tree edges: (S, a), (a, b), (a, c), (S, d)
2. Forward edge: (S, c)
   - S is an ancestor for vertex c
3. Backward edge: (b, S)
   - S is an ancestor for vertex b
4. Cross edge: (d, c)
   - d and c do not have ancestor/descendant relationship
(4) cross edge \((d, c)\)

d, c do not have ancestor/descendant relationships on the tree

\[ \begin{array}{cccc}
\cdots & c & \cdots & d
\end{array} \]

Let \( \text{enter leave enter enter leave} \)

- Cycle finding

Proof of correctness:

1. If alg claims to find a cycle, there is indeed a cycle.
2. If alg claims the graph has no cycle, there is no cycle.

Assume towards contradiction that there is a cycle in graph: \( U_1, U_2, \ldots, U_c \)

\[(U_1, U_2), (U_2, U_3), \ldots, (U_{c-1}, U_c), (U_c, U_1)\]

are edges in the graph.

Let \( U_p \) be the first vertex visited by DFS on this cycle. W.l.o.g. assume \( U_p = U_1 \).

Want: when \( U_1 \) is visited, \( U_1 \) is on the stack.

Claim: when every other vertex \( (U_2, U_3, \ldots, U_c) \) is visited, \( U_1 \) is on the stack.

Assume towards contradiction that claim is false. There is a first vertex \( U_{p-1} \) s.t. when \( U_{p-1} \) is visited, \( U_1 \) is not on the stack.

Know: when \( U_{p-1} \) is visited, \( U_1 \) is on the stack when considering the edge \( U_{p-1} \) to \( U_p \) when considering the edge \( U_{p-1} \) to \( U_p \).

Case 1: \( U_p \) has not been visited.

Alg will call DFS-visit(\( U_p \)).

\[ \begin{array}{cccc}
\cdots & U_p & \cdots & U_1
\end{array} \]

Case 2: \( U_1 \) has been visited.
alg will call DFS-visit(Up)

\[
\begin{align*}
\text{case 2: } & \text{ Up has been visited.} \\
\text{Know Up were not visited when } U, \text{ enters stack} \\
\text{Now Up has been visited when } U, \text{ is still on stack}
\end{align*}
\]

when alg looks at edge \((U, v, U_i)\), this is a backward edge
alg can always find a cycle.

- BFS tree

- topological sort
  - algorithm: output the inverse of post-order
  - proof of correctness: assume towards contradiction that
    the algorithm is not correct, then there must be
    an edge \((u, v)\) where \(v\) comes later than \(u\) in
Post order:

consider the DFS Procedure

Case 0 if U is visited before V when the edge (u,v) is considered

Case 1.1 if U is not visited, DFS will visit V, (u,v) is a tree edge, and DFS(u) returns after DFS(v) this contradicts with assumption that U comes after U in post-order.

Case 1.2 if V is already visited, then U must be visited between U is visited and edge (u,v) is considered, so U is on the stack when V is visited, this contradicts with the same assumption.

Case 2 if U is visited before V. Since V is after U in Post-order, the only possible sequence of events is

V enter U enter U leave V leave

therefore when U is visited V is on the stack in this case (u,v) is a backward edge and forms a cycle. This contradicts with the fact that the input graph is acyclic.