Topological Sort: aka. Linearization

- Directed Acyclic Graph (DAG): A directed graph with no cycle.
- Used to model processes (without a feedback loop).
- Partial Order: A binary relation \( \leq \) over a set \( P \) is a partial order if it is
  - reflexive: \( a \leq a \).
  - antisymmetric: \( a \leq b \land b \leq a \Rightarrow a = b \).
  - transitive: \( a \leq b \land b \leq c \Rightarrow a \leq c \).

  \( P \) is called a partially ordered set or poset.

- Total Order: \( \forall a, b, c \in P \) the following properties hold:
  - reflexive: \( a \leq a \).
  - totality: \( a \leq b \lor b \leq a \).
  - transitivity: \( a \leq b \land b \leq c \Rightarrow a \leq c \).

- Topological sort linearizes a partial ordering, i.e. it gives a total ordering.

- Examples of a DAG:

```
Prof A --- Prof G
|         |
|         |
Prof C --- Prof D --- Prof H
|         |
|         |
Prof E --- Prof I --- Prof H
|         |
|         |
Prof J  --- Prof K

```

```
Under shorts  --- Socks
|               |
|               |
Pants   --- Shirt
|               |
|               |
belt --- Skirt
|               |
|               |
Jacket

```

"\( \rightarrow \)" is advisor of

"\( \rightarrow \)" is worn before
**TOPOLOGICAL-SORT(G)**

1. Call **DFS(G)** to compute $v.f$ values i.e. the finish times $v.f \in V$.
2. Output the vertices in the decreasing order of finishing times.

**How?**

**Answer:**

(a) As each vertex is finished, insert it into the front of a linked list.

(b) Return the linked list of vertices.

**Time:** $O(V + E)$.

**Example:**

```
Prof A --> 1/14 --> Prof B --> 15/16 --> Prof H --> 18/19

Prof C --> 2/9 --> Prof D --> 10/13 --> Prof I --> 11/12

Prof E --> 3/8

Prof F --> 4/7

Prof I --> 5/6

Prof K
```

```
G -> H -> B -> A -> D -> I
     | -> C -> E -> J -> K
```
Strongly Connected Components (SCC):

- A strongly connected component of a directed graph $G = (V,E)$ is a maximal set of vertices $C \subseteq V$ such that for $u, v \in C$ both $u \to v$ and $v \to u$.

Example:

How to compute SCC?

- Do a DFS on $G$.
- Compute the transpose $G^T$ (i.e., all edges reversed) of $G$. Do a DFS on $G^T$, but in the main loop, consider the vertices in order of decreasing $v.f$ values as computed in the first DFS.
- Output the vertices of the DFS forest for $G^T$ as separate SCC nodes.

Time: $O(V + E)$.

Theorem: $G^{scc}$ is a DAG.