Topological Sort:

**Input:** DAG (Directed Acyclic Graph) $G$

**Output:** Ordering of the vertices of $G$ in a row such that the edges of $G$ go from a vertex on the left to a vertex on the right.

![Diagram of a DAG](attachment:image.png)
Alexander's idea: Look for a vertex with no incoming edges (source).

Lemma: In a DAG there is always a source.

$P$: maximal path in $G$

$v'$ cannot exist as otherwise $P$ is not maximal.
i. vertex u is a **sink**

   vertex w/ no outgoing edges.

Similarly, vertex u is a **source**.

**Algorithm: TS(G)**

1. u ← source in G          \(O(n)\)
2. \(G' \leftarrow G - u\)     \(O(1)\)
3. \(L' \leftarrow TS(G')\)     \(T(n-1)\)
4. Output u followed by \(L'\). \(O(1)\)

\[T(n) = T(n-1) + O(n)\]

\[= O(n^2)\]

---

An efficient implementation:

1. Find all sources in \(G\). Call this set \(S\).
1. $O(1) \ u \leftarrow \text{arbitrary vertex} \ m \in S$

2. $O(1) \ \text{for } u$

3. $O(1) \ \text{for each neighbor} \ v \in N(u) \ \text{in } V \ \text{do}$

4. $O(d_{y}(w)) \ \text{indegree}(v) \leftarrow \text{indegree}(v) - 1$

   if $\text{indegree}(v) = 0$ then

   add $v$ to $S$.

5. go to (1).

Running time: $O(n + m) + O(n) + O(\sum d_{y}(w))$

$\overset{\text{over the course of the}}{\overset{\text{first cycle.}}{\text{O}(m)}}$

$= O(n + m)$. 
DFS-based alg.

1. Do DFS(A)

2. Sort vertices in order f HJ

Running time: $O(n(m+n)) + O(n \log n)$
\[ = O(m + n \lg n). \]

Do both steps together. Every time a vertex finishes, we insert it at the beginning of the linked list.

\[
\therefore O(nm) \text{ alg.}
\]

Thus: Our alg works.

Proof: Let \( e = (u, v) \) be any edge in \( G \).

\[ u \xrightarrow{e} v \]

Want to prove that our alg outputs \( u \) and then \( v \).

Can we: \( d[u] < d[v] \)

Want to show that \( f[u] > f[v] \).
Because e exists and because of the theorem, we have learned, life is good.

\[ f(u) > f(v) \]

At \( d(u) \), we have \( u \to v \) in \( G \) (\( edn(\text{uv}) \)).

\[ \vdash u \text{ is a descendant of } v \text{ in the DFS forest.} \]

By the Parenthese theorem \( f(u) > f(v) \).

Case II: \( d(v) < d(u) \) \[ u \xrightarrow{e} v \]

By WPT, there must be a path from \( v \to u \), i.e., a cycle, a contradiction.

At \( d(v) \), no WP from \( v \to u \);

\[ \vdash \text{this would be a cycle.} \]
u is not a descendant of v in the DFS forest.

By Parent-hence thin:

```
  du  vv  du
  ^    ^
    fd
```

**Shortest Paths**

**I/P:** Directed graph $G = (V, E)$

*Weights on edges (tree)*

$\forall \in V$

**Output:** Shortest Path Tree rooted at $\in$.

That is, we want shortest paths from $\in$ to every other vertex in $G$. 


Ans: (Dijkstra's algorithm).

1. for each u ∈ V do
\( d[u] \leq \infty \)
\( d[e] \leq 0 \).

\( S \leftarrow \emptyset \)

// assume all vertices are reachable from \( S \).

while \( S \neq V \) do

\( u \leftarrow \text{vertex in } V \setminus S \text{ with smallest } d[u] \).

\( S \leftarrow S \cup \{u\} \)

for each \( v \in N(u) \) s.t. \( v \in V \setminus S \) do

\[ d[v] + \text{wwu} < d[v] \]

if \( \text{wwu} < d[v] \) then

\[ d[v] \leftarrow \text{wwu} \]

\[ \Pi[v] \leftarrow u \]

// add (u,v) to the tree.