Thus G has a top order \( \iff \) G is a DAG

Lemma 1: G has top order \( \iff \) G is a DAG

Proof by contradiction: Suppose G has top order \( V_i \).

Suppose G is not a DAG, there is a directed cycle.

Let \( i_i \) be the lowest index in the cycle.

Cycle \( \iff \) there is edge \( (V_{i_k}, V_{i_i}) \).

Since \( i_i \) is lowest, \( i_k \geq i_i \).

This contradicts to the fact that G is a top order.

Lemma 2a: If G is a DAG, it has a source node.

(source node = node with no incoming edge)

Proof by construction:

Algorithm:

Start at any node \( V \).

While there is an incoming edge \( (u, V) \) from \( V \),

\( V \leftarrow u \).

Output \( V \).

Termination:

No node is repeated in the loop because no directed cycle.

Hence, it takes \( O(n) \) time.

Conclusion: \( V \) is source because of the loop condition.
Correctness: \( V \) is source because of the loop condition.

Lemma 2b: \( G \) is a DAG \( \Rightarrow \) it has topo order

Algorithm:
1. Let \( L \) be empty list
2. While \( G \neq \emptyset \)
   - Find a source \( u \) in \( G \)
   - Put \( u \) in the end of \( L \)
   - Delete \( u \) from \( G \)
3. Output \( L \).

Time: \( O(n^2) = \frac{O(n)}{\text{iter}} \times O(n) \)

Improve this to \( O(m+n) \).

Correctness: We can find a source \( u \) in \( G \) by Lemma 2a, and by \( G \) is DAG.

Let \( V_1, V_2, \ldots, V_n \) be the order algo find.

For any edge \((V_i, V_j)\)
- \( i < j \) because the algo do job \( V_j \) only if all incoming edge to \( V_j \) is removed/done.

Proof of theorem follows from Lem 1 and Lem 2b.