CSCI 432 Handout 10: Topological Sort

Name: _____

March 12, 2026

Topological Sort: The Algorithm

Topological sort takes a poset $(P, \leq)$ and returns a total order of all objects that is *compatible* with the partial order of the poset. By compatible, we mean that for all $a \leq b \in P$, we have a appears before b in the total order. For example, consider P to be a set of points in $\mathbb{R}^2$ and let the partial order be the product order; that is, $(a_x, a_y) \leq (b_x, b_y)$ if and only if $a_x \leq b_x$ and $a_y \leq b_y$. Given a set:

$$P = \{(0,0), (\pi,\pi), (2,4), (1,2)\}$$

we can return the following total order:

$$P = [(0,0), (1,2), (2,4), (\pi,\pi)].$$

Note that this is not the only compatible total order.

1. What other total order could have been returned?

   Answer

2. Another order we can impose on $\mathbb{R}^2$ is the lexicographical ordering (sometimes called the dictionary ordering), where $(a_x, a_y) \leq (b_x, b_y)$ if and only if $a_x < a_y$ or ($a_x = a_y$ and $a_y \leq b_y$). What total orders on P exist that are compatible with the lexicographical ordering?

   Answer

The following algorithm takes as input a poset, represented as a directed acyclic graph, and returns a compatible total order. If the poset orders $a \leq b$, then the DAG has an edge from vertex a to vertex b. For this algorithm, vertices have at least one attribute: outgoing, which stores a list of vertices that define the outgoing edges.

Algorithm 1 Topological Sort

Input: $G = (V, E)$, a DAG
Output: L, a list of vertices in G in a total order compatible with the partial order of the DAG.

```
for v ∈ V do
    Add an attribute 'v.count' and initialize it to the indegree of v.
end for
Q ← queue containing the vertices with 'count'=0.
L ← array of length |V|
i ← 1
while Q is not empty do
    v ← Q.POP
    L[i] ← v
    i++
    for u ∈ v.outgoing do
        u.count−−
        if u.count = 0 then
            Add u to Q
        end if
    end for
end while
return L
```

3. In Line 4, why is Q not empty?
   Answer

4. Explain how topological sort is helpful for dynamic programming.
   Answer

5. Walk through a small example.
   Answer

Topological Sort: The Decrementing Function

Recall from HO-07 that a decrementing function is a function $D\colon \mathcal{S} \to \mathbb{N}$, where $\mathcal{S}$ is the state space that such that (when there is no recursion) each time a loop is re-entered, the function is strictly less than the last time it entered that loop.

6. What is the decrementing function for the for loop?

   Answer

7. What is the decrementing function for the while loop?

   Answer

Topological Sort: The Loop Invariant

Next, we investigate partial correctness of the while loop. For this loop, what are the following statements (start with your best guess, then come back and revise if needed):

The loop guard G:

The post-condition Q:

The pre-condition P:

The loop invariant $L = L_i$:

Can you use this loop invariant to prove partial correctness.

8. Initialization: If the preconditions are met and if we enter the loop, then the loop invariant is correct. $(P \wedge G \implies L)$.

   Answer

9. Maintenance: If we entered the loop and the loop invariant held, then when we exit the loop, the loop invariant will still hold: $(G \wedge L_i \implies L_{i+1})$.

   Answer

10. End: If the loop ends and the loop invariant was correct, then the algorithm is correct. $(L \wedge \neg G \implies Q)$.

    Answer