Topological Sort: Decrementing Functions and Loop Invariants

CSCI 432

October 25, 2023 Day 2: 27 Oct 2023

Name:
Who did you work with today?

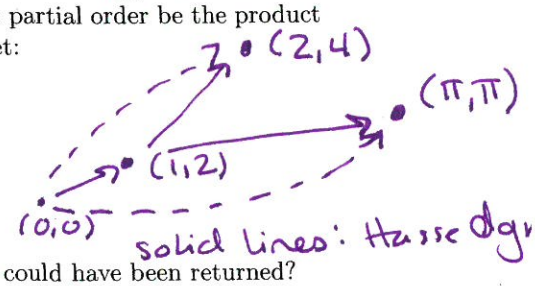
Topological Sort: The Algorithm

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:

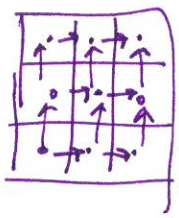
$$[(0,0), (1,2), (2,4), (\pi,\pi)].$$



Note that this is not the only compatible total order. What other total order could have been returned?

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.



A Dec. fcn is a fcn

$D: S \longrightarrow \mathbb{N}$

S: state space; $\mathbb{N}$: or any well-ordered set

Topological Sort: The Decrementing Function

What is the decrementing function for the for loop?

Answer

To prove it is a dec. fcn

1. D(state) must be $\in \mathbb{N}$
2. must get smaller each time through the loop!

What is the decrementing function for the while loop?

Answer

Topological Sort: The Loop Invariant

Here, 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: "Q is not empty" (aka - the statement that keeps us in the loop)

The post-condition Q: What was the loop supposed to do?

The pre-condition P:

The loop invariant $L = L_i$:

Can you use this loop invariant to prove partial correctness (Remember, there are three parts to partial correctness: Initialization, Maintenance, and End).

Answer

1. INITIALIZATION $P \Rightarrow L$ "the first time we attempt entering the loop, LI is true"
2. MAINT. $L_i \wedge G \Rightarrow L_{i+1}$
3. END $L \wedge \neg G \Rightarrow Q$

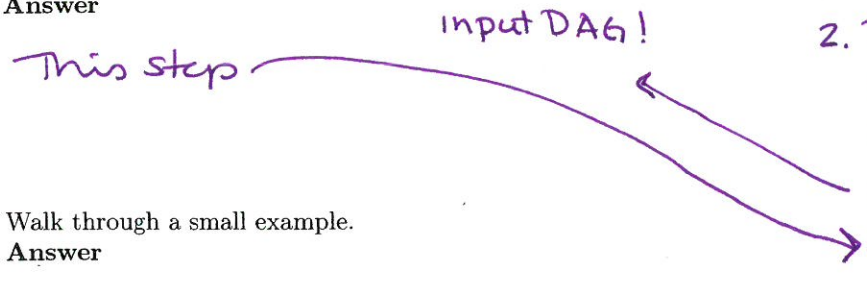
Algorithm 1 Topological Sort

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.

```
1: for v ∈ V do
2:    Add an attribute 'v.count' and initialize it to the indegree of v.
3: end for
4: Q ← the sent of vertices with 'count'=0.
5: L ← array of length |V|
6: i ← 1
7: while Q is not empty do
8:    v ← Q.POP
9:    L[i] ← v
10:   i++
11:   for u ∈ v.outgoing do
12:      u.count−−
13:      if u.count = 0 then
14:         Add u to Q
15:      end if
16:   end for
17: end while
18: return L
```

In Line 4, why is Q not empty?
Answer

Explain how topological sort is helpful for dynamic programming.
Answer



Walk through a small example.
Answer

Steps for Dynamic Problems
1. Formulate Prob Recursively
2. Build-up ~~subsp~~ solutions
   a) identify sub problems
   b) data structure
   c) identify dependecies
   d) good eval order
   e) analyze