A Topological Sorting Algorithm

UNLV: Analysis of Algorithms   Lawrence L. Larmore

Topological Order

Given a directed graph $G = (V, E)$, a *topological ordering* of G is an ordering "$<$" such that $u < v$ if $(u, v) \in E$. If G is acyclic it has at least one topological order, but if G is cyclic, *i.e.,* has a cycle, it does not have a topological order.

The algorithm written below writes the vertices of G in topological order, provided G is acyclic. The algorithm assumes G is already represented by both in-neighbor lists and out-neighbor lists.

```
Q := EmptyQueue
for all v in V
   NumSource[v] := Indegree(v)
   if (NumSource[v] = 0)
      Insert(Q,v)
   endif
endfor
while not Empty(Q) do
   u := Dequeue(Q)
   Write(u)
   for all w in OutNbrs(u) do
      NumSource[w] := NumSource[w] - 1
      if (NumSource[w] = 0)
         Insert(Q,w)
      endif
   endfor
endwhile
if (not Empty(Q))
   Write(' The graph has a cycle.')
endif
```

Note that I am using a queue. You could use a stack instead, in fact, you could use any priority queue.

The running time of this algorithm is $O(n + m)$. If G is cyclic, the algorithm will halt with the queue not empty.