General Algorithms on a DAG

Here DAG = Directed Acyclic Graph

At the highest level, a DAG algorithm typically looks like the following

Given graph $G=(V,E)$ and start node $s$, to calculate a property “prop” for each node

1) for each $v$ in $V$, initialize $v.prop$
2) initialize $s.prop$
3) determine topological order of $G$ (may already be known)

4)
for each $u$ in $V$, taken in topologic order
   for each $v$ such that $(u,v)$ is an edge
       adjust $v.prop$ based on $u.prop$

5) (optional) for a specified target node $t$, return $t.prop$

More specifically for homework 2

In this case the topological order is 1,2,3,…,N. Start node is node 1 and target node is node N. Here we will outline how to compute the longest path, which will be stored in an array $LP[1..N]$. The weight of edge $(u,v)$ is stored as $W[u,v]$.

initialize
$LP[1]=0$
for $i = 2$ to $N$
   $LP[i] = -\infty$

loop
for $i = 1$ to $N-1$
   for $j=i+1$ to $N$
      if $(i,j)$ is an edge
         then $LP[j] = \max[ LP[j], LP[i]+W[i,j] ]$
         (update other properties here, if any)

return
print “longest path is” $LP[N]$. 