Algorithmn HW7

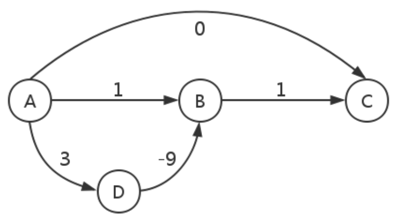
5140379032 JIN YI FAN

Problem 8.16

Just change the RELAX part to store the best predecessor of the vertex, and later we can find the shortest path iterately.

procedure RELAX(u, v)
    if $d[v] > d[u] + w(u, v)$ **then**
        $d[v] \leftarrow d[u] + w(u, v)$
        $pred[v] \leftarrow u$
    end if
end procedure

Problem 8.19

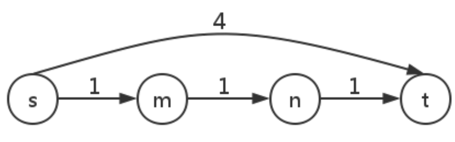


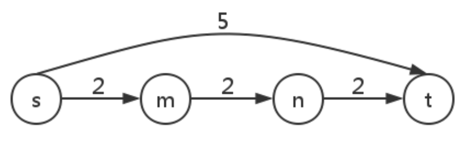
Considering this case, Dijkstra works like this:
(1) Starting from A, set $d(A) = 0$ and $d(others) = +\infty$;
(2) A out, set $d(B) = 1$, $d(C) = 0$ and $d(D) = 3$;
(3) C out, with no successor edge;
(4) B out, with no change $(1 + 1 > 0)$;
(5) D out, updating $d(B) = -6$, then terminate.

Obviously, $d(C)$ should be -5 instead of 0 in this case, which shows Dijkstra does not work in this case.

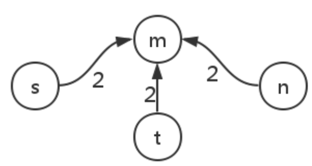
True or False

(a) **False**, orignally it's $s \rightarrow m \rightarrow n \rightarrow t$ but after modification it will be $s \rightarrow t$





(b)**False**, in the DAG on the right, Dijkstra starting at m definitely won't pick vertexes as any of the topological orders.



(c)**False**, because in that case, the short path would be either **(1)**modified to other path-that won't change-or **(2)**remain the same path-that will increase the short path by exactly x

(d)**True**, because Bellman-Ford will search every possible edge for every vertex regardless of the order, so it will update later if there is some negative value.

Paths in DAG

Require: a DAG $G = (V, S)$
Ensure: the number of paths in G

procedure CountPath(G)
  topologically sort $G.V$
  $p[s] \leftarrow 1$, $p[\text{other } v \in G.V] \leftarrow 0$ ▷ $p[]$ stores the number of paths
  for each vertex i in topological sorted sequence **do**
    for each vertex j connected with i **do**
      $p[j] \leftarrow p[j] + p[i]$ ▷ The last element will be the total number of paths
    end for
  end for
end procedure

We can have the relationship as follows:
$initial(u) = u$ is the last element in topo order? $1 : 1 + \sum_{(u,v)\in E}$
$path(u) = initial(u) + \sum_{(u,v)\in E} path(v)$
Since $path(u)$ and $initial(u)$ can be computed in linear time, the algorithm takes $O(V + E)$ time

Shortest path tree

Require: directed graph $G = (V, E)$, a tree $T = (V, E')$, $s \in V$ and $E' \in E$
Ensure: whether T is the shortest-path tree of G starting with s

$Q \leftarrow [s]$
while Q is not empty **do** ▷ travel the tree
  $u = Q.pop()$ ▷ Q is a priority queue
  for each edge $(u, v) \in E'$ **do**
    $d[v] = d[u] + w(u, v)$ ▷ $d[]$ keeps the shortest paths
    $Q.push(v)$
    remove (u, v) from E'
  end for
end while
for each edge $(u, v) \in E$ **do** ▷ check in the graph whether it is shortest
  $temp \leftarrow d[v]$
  Relax(u, v) ▷ compare the value before and after Relax
  if $temp \neq d[v]$ **then**
    return $False$
  end if
end for
return $True$