CS345 Assignment 4

Chinmay Goyal(180206) and Supreeth Baliga(180801)

1 Path Numbering in a DAG

Solution:
Here, we are assuming that a pair of vertices cannot have more than one edge and there are no self loops. We propose the following algorithm:

Algorithm:

We first apply topological sort on the graph G. Let us say that the topological sequence that we get is T(every DAG has a valid topological sequencing as proved in class). Now, since t is **the** exit vertex, it will be the last element of T and since s is **the** root vertex, it will be the first element of T (by property of topological sort). Let us say for each node in the graph say u, we are storing the value *num-ways(u)* which denotes the number of paths from s to u. For each node u in G except s, we set num-ways(u)=0 and set num-ways(s)=1. Now, we traverse T in the forward order i.e. starting from the first element which is s.

We do the following for each node in the traversal:

Say, in this traversal, we are currently at node v. Consider all the incoming edges of v. Say these edges are $(u_1, v), (u_2, v), ..., (u_k, v)$. Now, all the points $u_1, u_2, ..., u_k$ will have their num-ways already calculated since we are doing a bottom-up approach in T. So, we do the following. Maintain a variable *curr* which is initially 0.

```
for i=1 to k, we do the following:
    weight(u_i, v) = curr
    curr = curr + num-ways(u_i)
    num-ways(v) += num-ways(u_i)
```

After this, we go to next point in T and do the above procedure again and so on.

Let us write the formal pseudocode for this algorithm (say PathNumbering).

Pseudocode:

```
PathNumbering(G) {
    T <- TopologicalSort(G);
    num-ways(s) = 1;
    for v in traversal of T {
        curr = 0;
        for each edge (u,v)  {//each incoming edge
            weight(u,v) = curr;
            curr = curr + num-ways(u);
            num-ways(v) = num-ways(v) + num-ways(u);
        }
    }
}
```

Proof of Correctness:
We will use Induction to prove the correctness of our algorithm.

Claim: Let the node under consideration be v and the number of paths to reach this vertex from s be num_v, then the pathid of each path from s to v will be unique and will be from 0 to $num_v - 1$.

Base Case: When $v = s$, then number of paths to reach s from s is 1 that is by not moving and we assign value 0 to that path. Here we can note that a path is a sequential set of vertices where all the vertices are unique, so in the base case we are considering the path which consists of only one vertex that is s, hence the pathid 0 for this path satisfies our claim. We can also look at in the way that, if we have a graph consisting of only one node where the source and the target vertex are the same, then this provides our base case and also upholds our claim. Hence our base case is satisfied.

General Case: Let the node under consideration be v. Let this node have incoming edges from nodes $u_1, u_2, ...u_k$. The number of ways to reach a vertex from s to $u_i = num_i$ and the number of ways to reach vertex v from s be num_v. By induction hypothesis we know that the paths from s to $\{u_i\}_{i=1}^{k}$ have unique pathids from 0 to $num_i - 1$ for each respective vertex. Let w_i be the weight assigned to the edge (u_i, v). Let $w_1 = 0$, then this way we have paths from s to v via u_1, that have unique pathids from 0 to $num_1 - 1$. Similarly if we assign $w_2 = num_1$, then we have paths from s to v via u_2 with unique pathids from num_1 to $num_1 + num_2 - 1$. Generalizing it, we assign weight $w_l = \sum_{i=1}^{l-1} num_i$, through this method we will have paths from s to v via u_l with unique pathids in the range $\sum_{i=1}^{l-1} num_i$ to $\sum_{i=1}^{l} num_i - 1$. Hence doing this for all the incoming edges into v, we will get paths from s to v with unique pathids in the range 0 to $\sum_{i=1}^{k} num_k - 1 = num_v - 1$. Hence Proved for the general case.
This proves our induction hypothesis which henceforth proves the correctness of our algorithm.

Time Complexity:

Let us say the graph G has n vertices and m edges. Topological Sort takes O(m+n) time. We traverse the topologically sorted list T is O(n) time. In the traversal of topologically sorted list, for each vertex (say v), we are processing indegree(v) number of edges and for each edge, we take O(1) time. Thus, for each vertex, we take O(indegree(v)) time to process it. Thus, total time complexity is given by:

$$Time(PathNumbering(G(V,E))) = O(m+n) + O(n) + \sum_{v \in V} O(indegree(v))$$

Now, $\sum_{v \in V} O(indegree(v)) = O(\text{Number of edges in G}) = O(m)$.

Thus,

$$Time(PathNumbering(G)) = O(m+n)$$