Problem Set 4

Noella James

03/04/2017

collaborators: none

Problem 4-1: Longest Path

Solution

We are given a DAG G. First, perform a topological sort on G. The topological sort orders the nodes from 1 to n where 1 is leftmost node and n is rightmost node.
If vertex u does not have any incident edges then $path-length(u) = 0$. If vertex u has multiple incident edges from vertexes $v_1, v_2, \ldots, v_m$ the the maximum $path - length(u) = 1 + max(path - length(v_1)\,, path - length(v_2)\,, \ldots\,, path - length(v_m))$ The maximum path length for a graph G is thus $max(path - length(v_i)\,, path - length(v_2)\ \ldots\ , path - length(v_n))$

Algorithm

The algorithm for finding the longest path of a DAG is based on memoization. It prints the path from the last vertex to start of the path and returns the length of the longest path

Algorithm 1 LONGEST PATH

```
procedure LONGEST-PATH(G = (V, E))                ▷ G is a DAG
    TOPOLOGICAL-SORT(G)
    Array M[1 . . . n]
    Initialize M[i] = 0 for each i = 1 , 2 , . . . , n
    for i = 1 . . . n do
        if n_i has an incoming edge then
            nopt_max ← 0
            for all immediate predecessors p_i of n_i do
                if M[p_i] > nopt_max then
                    nopt_max ← M[p_i]
                end if
            end for
            M[i] ← nopt_max + 1
        end if
    end for
    j ← index of M with maximum value
    PRINT-PATH(G, M, j)
    return max(M[i]) for i in 1 . . . n
end procedure
```

Algorithm 2 PRINT PATH

```
procedure PRINT-PATH(G = (V, E), M, j)        ▷ G is a DAG, M is the
longest paths for each vertex, j is a vertex
    if  M[j] = 0 then
        return
    else
        PRINT(j)
        Find vertex i from where an edge is incident on j such that its path
length = M[j] − 1
        PRINT-PATH(G, M, i)
        return
    end if
end procedure
```

Complexity

Let $n = |V|$ and $m = |E|$ for DAG $G = (V, E)$
An optimal topological sort has the complexity of $O(n + m)$.
The initialization of array M has the complexity $O(n)$.
The outer for loop will execute $O(n)$ times, yet the overall execution of the innermost loop is at most $O(m)$.
Therefore, the overall complexity of the loops is at most $O(n + m)$.
Therefore, the overall complexity of the algorithm is $O(n + m)$.

Correctness

Lemma 0.1. *The algorithm $LONGEST-PATH(j)$ correctly computes longest path for each vertex $j = 1, 2, \ldots, n$.*

Proof. By definition, $M(1) = 0$ as index 1 hosts the node with no incident edges. Now, take some $j > 0$, and suppose by way of induction that $LONGEST - PATH(i)$ correctly computes $M(i)$ for all $i < j$. For the induction step, for vertex j, we identify a node k for which there is an edge from k to j such that k has the maximum path length among all the nodes that have a edge incident to j. Based on ordering of topological sort $k < j$ and thus holds the maximum path lengths from one of the nodes without any incident edge to k. Thus, the path length to j would be 1 more than the path length to k.Thus, the path length to j is the longest path. This completes the proof. $\square$