CMPSC 465 Fall 2024 | Data Structures & Algorithms Mingfu Shao and Debarati Das

Recitation 3

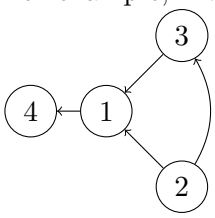
Sep 23, 2024

1. (pts.) We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such that vertex i is reachable from vertex j. (As a convention, we assume that i is reachable from i.) Provide an algorithm to compute all the values $m(1)$,..., $m(n)$ and show that these can be computed in $O(|V| + |E|)$ time.

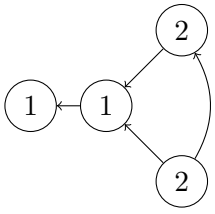
   Solution

   All vertices within the same SCC should ultimately receive the same label, along with all the vertices within any other SCC that our current SCC can reach. Thus, for all unvisited nodes in the graph, we can run DFS starting from the numerically first unvisited node j, and for all the nodes i that are reachable from j, we assign $m(i) = j$.

   For example, if we start with the graph:



   When we run DFS on this graph, we'll need two 'starts'. Once starting at 1 (smallest valued vertex) and then again starting at 2 (second smallest value we hadn't seen already) after we hit a dead end. Our two runs produce the following labels:



   Looking back at our original graph, we can see that this does indeed correspond to the right answer!

   We can see that this algorithm is very similar to our algorithm for finding SCCs. Rather than starting from a vertex in a sink SCC, however, we start from the vertex with the smallest value. So rather than finding SCCs one at a time, we'll first find all the vertices in the SCCs that can reach 1, then all the vertices in SCCs that can reach the next smallest available number, and so on.

Algorithm

```
Function DFS-Clusters(G):
    while there are unvisited nodes in G do
        Run DFS on G starting from the numerically-first unvisited node j
        for i visited by this DFS do
            m(i) := j
        end
    end
```

To see that this algorithm is correct, note that if a vertex i is assigned a value then that value is the smallest of the nodes that can reach it in G, and every node is assigned a value because the loop does not terminate until this happens.

The running time is $O(|V| + |E|)$ since we process every vertex and edge exactly once in the DFS.

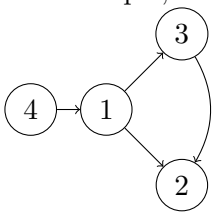
2. (pts.)

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such that vertex j is reachable from vertex i. (As a convention, we assume that i is reachable from i.) Provide an algorithm to compute all the values $m(1)$,..., $m(n)$ and show that these can be computed in $O(|V| + |E|)$ time.

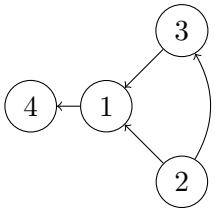
Solution

The approach is the same one as the previous problem, except this time we reverse all the edges of the graph first and then run the DFS-Clusters() algorithm on G^R. Here, if we reverse the direction of every edge in the graph, the set of vertices that can reach some node k become precisely the set of vertices that k can reach.

For example, if we start with the graph:



We can see that upon reversing it we'd get the graph:



Observe that the set of vertices reachable by j in G^R is the set of vertices which can reach j in G.

Algorithm

Let G^R be the graph G with its edge directions reversed. The algorithm is as follows.

Function DFS-Clusters-Reverse(G):
    $G^R \leftarrow \mathit{Reverse\text{-}Edges}(G)$
    DFS-Clusters(G^R)

The running time is $O(|V| + |E|)$ since computing G^R can be done in linear time, and we process every vertex and edge exactly once in the DFS.