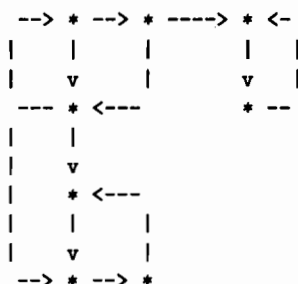


Data Structures and Algorithms 2001

Answer notes:

A directed graph $G(V,E)$ is a set V of vertices and a set E of directed edges connecting selected pairs of vertices from V . A strongly connected component is a maximal subset of V for which paths exists connecting any of its vertices to any other.



```

DFS(v)
{ mark(v)
  setdiscoverytime(v,t++)
  for each w in successors(v) do
    if unmarked(w) do { DFS(w); addtreeedge(v,w) }
  setfinishingtime(v, t++)
}
t := 1
while w is an unmarked vertex do DFS(w)

setdiscoverytime(v,t) // set the discovery time
setfinishingtime(v,t) // set the finishing time
addtreeedge(v,w)      // add edge (v,w) to the depth first
                      // spanning tree
                      // eg: tsuccs(v) := mk2(tsuccs(v), w)
                      // and/or
                      // parent(w) := v

```

	discovery time	finishing time
v	d1	f1
phi(v)	d2	f2

Forefather property: there is a path from $\phi(v)$ to v

Proof:

```

true if v = phi(v)
so consider v != phi(v)
we know d1 < f1 and d2 < f2
and f1 <= f2 (1)
possible orderings (di/fi dj/fj must nest or be disjoint)
d1 < f1 < d2 < f2      no -- there must be a path from v to phi(v)
d2 < f2 < d1 < f1      no -- (1) fails

```

d1<d2<f2<d2 no -- (1) fails
d2<d1<f1<f2 yes -- so there is a path from phi(v) to v

so v and phi(v) are in the same strongly connected component

Algorithm

- 1 do DFS on graph to assign finishing numbers
- 2 find the vertex v with largest finishing time that is not yet in a strongly connected component. This is a forefather. Find all vertices that can be reached from v in the graph with all its edges reversed.
- 3 repeat 2 until all found

The cost is $O(n)$ since it just used DFS twice.