- A strongly connected component (SCC) of a directed graph $G(V, E)$ is a maximal induced subgraph $G'(V', E')$ of G such that for every two vertices v_i and v_j in G' there exists paths from v_i to v_j and from v_j to v_j in G'.

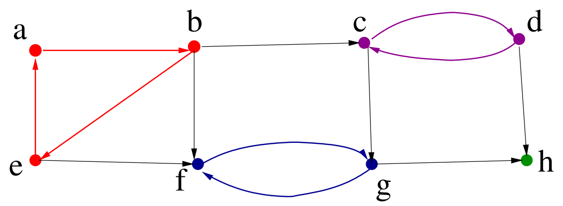

Figure 1: An example depicting each SCC of graph in a distinct color.

- In a DFT of G, if a node $v \in SCC_i$ is the first node discovered among all the nodes of SCC_i then v is said to be the *root of* SCC_i.

- A review of categorizing arcs in a depth-first traversal of G: When an arc (v, w) is explored for the first time, if the color of w is (i) white then (v, w) is a tree edge, (ii) grey then (v, w) is a back edge, (iii) black and $v.d < w.d$ then (v, w) is a forward edge, or (iv) black and $v.d > w.d$ then (v, w) is a cross edge.

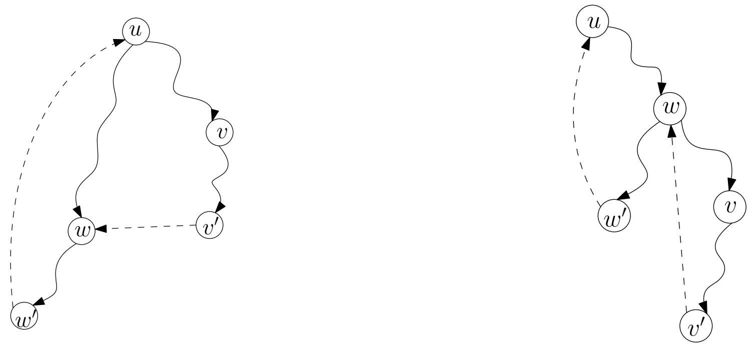

Figure 2: For the cross arc $v'w$ in the left figure, w is black and $v'.d > w.d$.
For the back arc $v'w$ in the right figure, w is grey and $v'.d > w.d$.

- For every node $v \in G$, $v.\ell$ is defined as $\min\{$discovery time of $v \cup \{$discovery time of $w \mid$ there is a cross arc or back arc from a descendent v' of v to w, and the root of the SCC containing w is an ancestor of $v\}\}$.

- Observation: For every node, $v.\ell \leq v.d$.

- v is the root of SCC containing v iff $v.\ell = v.d$

  "$\Rightarrow$ (proof by contradiction):" Suppose v is the root of a SCC of G but $v.\ell < v.d$. The latter implies there exists nodes v', w, and u such that there is a cross arc or back arc from a descendent v' of v to w, and the root u of the SCC containing w is an ancestor of v. Since u is an ancestor of w, $u.d \leq w.d$. And since $w.d < v.d$, we can conclude $u.d < v.d$. Since $u.d < v.d$ and u is an ancestor of v, it must be that u is a proper ancestor of v. In addition, note that v belongs to the SCC containing w. Therefore, v cannot be the root of SCC containing w.

"$\Leftarrow$ (proof by contradiction):" Suppose $v.\ell = v.d$ but v is not the root of the SCC containing v. Then some proper ancestor u of v is the root. To reach u from v in that SCC, in any path from v to u, there exists a back arc to an ancestor of v (analyzed why it is so) or a cross arc from a descendant v' of v to a node w. Note that $w.d < v.d$. In addition, u and w are in the same SCC. Thus $v.\ell \leq w.d < v.d$.

- Hints for devising an algorithm: (i) In a single depth-first traversal of G, compute $v.\ell$ for every v. (ii) Push vertices on to a stack in the order in which $DFT(G)$ discovers them. (iii) Just after associating black color to any node v, if $v.\ell$ is equal to $v.d$, then pop and output vertices from the top of the stack up to v - these together define an SCC in the given graph.

- For any vertex v of G that is not explored yet, invoke the following function.

```
$findscc$(v): (discoverytime and stack are global variables)
  (i) $v.d := discoverytime$
  (ii) $v.\ell := discoverytime$++
  (iii) push $v$ on to stack
  (iv) for each arc $(v, w)$ that is encountered for the first time in the DFT of $G$
      (a) if $(v, w)$ is a tree arc then $findscc(w)$ and set $v.\ell$ to $\min(v.\ell, w.\ell)$
      (b) else if $(v, w)$ is a back/cross arc and $w$ is on stack[1] then set $v.\ell$ to $\min(v.\ell, w.d)$
          (if $w$ is not on stack, then $w$ belongs to an SCC already found; if $w$ is on stack, then the root of SCC containing $w$ is an ancestor
          of $v$)
  (v) if $(v.\ell == v.d)$ then        //output vertices of SCC to which $v$ is the root
      (a) while $w$ on top of stack
      (b)     pop $w$ from stack and put $w$ in the current SCC
      (c)     if $w == v$ then break
```

- Setting finish time stamps and associating colors to nodes are ignored in the above pseudocode as they are same as in DFT.

- Note that there is no need to consider forward arcs vw' in computing $v.\ell$ since tree arcs in (iv)(a) account $w'.\ell$.

- takes $O(|V| + |E|)$ time; unlike algorithm by Kosaraju, only one DFT is required in this algorithm

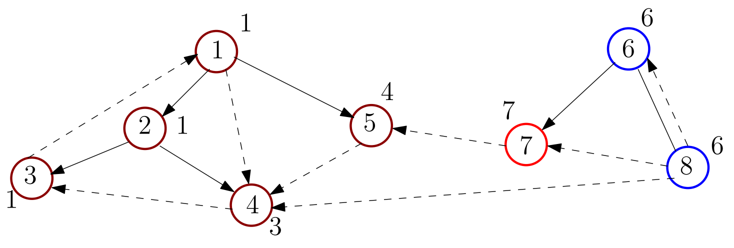

Figure 3: For every node v, both $v.d$ (in the circle) and $v.\ell$ (beside the circle) are shown.

[1]maintain a bit of information with each node to quickly determine whether it is on stack

- Proof of correctness involves proving (i) for every vertex v, $v.\ell$ is computed correctly, (ii) when finish time is associated to root r of any SCC, say SCC_i, the contents of stack from its top to the location where r is stored precisely comprises of those vertices that belong to SCC_i, and (iii) with the induction on the number of SCCs output while considering the SCCs in the order in which they are output, every SCC in the input graph is output by the algorithm.

  In proving (i), while referring to Fig. 2, consider the following:

- if v and w have no common ancestor, then before v is discovered, w must have been removed from stack - thus w has no effect on the computation of $v.\ell$
- otherwise, let u be the lowest common ancestor of v and w in the depth-first tree
    - if u is not in the same SCC as w, then there must be some node r along the tree path from u to w which is the root of SCC containing w; since r finishes before v is discovered, vertices belonging to this SCC must have been removed from the stack by the time arc $v'w$ is traversed
    - if u is in the same SCC as w, w is on the stack by the time $v'w$ is traversed; hence, $v'.\ell$ and $v.\ell$ are computed correctly

References:
Robert Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing, 1(2): 146-160, 1972.