

Biconnected Components Algorithm

To output the biconnected components of an undirected graph G , we modify the algorithm for finding articulation vertices so that whenever it encounters a new edge in G , it pushes it on a *stack*. Also after we finish the recursive search from a child v of vertex u , we check if u is an articulation vertex for v . If so, we output all edges on the stack upto and including (u, v) . Notice that when we return to the root we need to output a biconnected component even if the root is not a cut-vertex. (For example, the graph may not have *any* cut-vertices.)

```
proc BiconnectedComponents;
    count  $\leftarrow$  0;
    Initialize(stack);
    for each  $u \in V[G]$  do visited[u]  $\leftarrow$  false;
    for each  $u \in V[G]$  do parent[u]  $\leftarrow$  nil;
    for each  $u \in V[G]$  do
        if not visited[u] then DFS-Visit[u];
    end proc;

proc DFS-Visit(u);
    visited[u]  $\leftarrow$  true;
    count  $\leftarrow$  count+1;
    d[u]  $\leftarrow$  count;
    low[u]  $\leftarrow$  d[u];
    for each  $v \in \text{Adj}[u]$  do
        if not visited[v] then
            Push(stack, (u,v));
            parent[v]  $\leftarrow$  u;
            DFS-Visit(v);
            if (low[v]  $\geq$  d[u]) then OutputComp(u,v);
            low[u]  $\leftarrow$  min(low[u],low[v]);
        endif
        else if (not(parent[u]=v) and (d[v] < d[u]) then
            (* (u,v) is a back edge from u to its ancestor v *)
            Push(stack, (u,v));
            low[u]  $\leftarrow$  min(low[u],d[v]);
        endif;
    end for;
end proc;

proc OutputComp(u,v);
    Print(" New Biconnected Component Found");
    repeat
        Pop(stack, e);
        Print(e);
    until (e = (u,v));
end proc;
```