Lecture 6: An Application of DFS : Articulation points

Recall DFS algorithm:

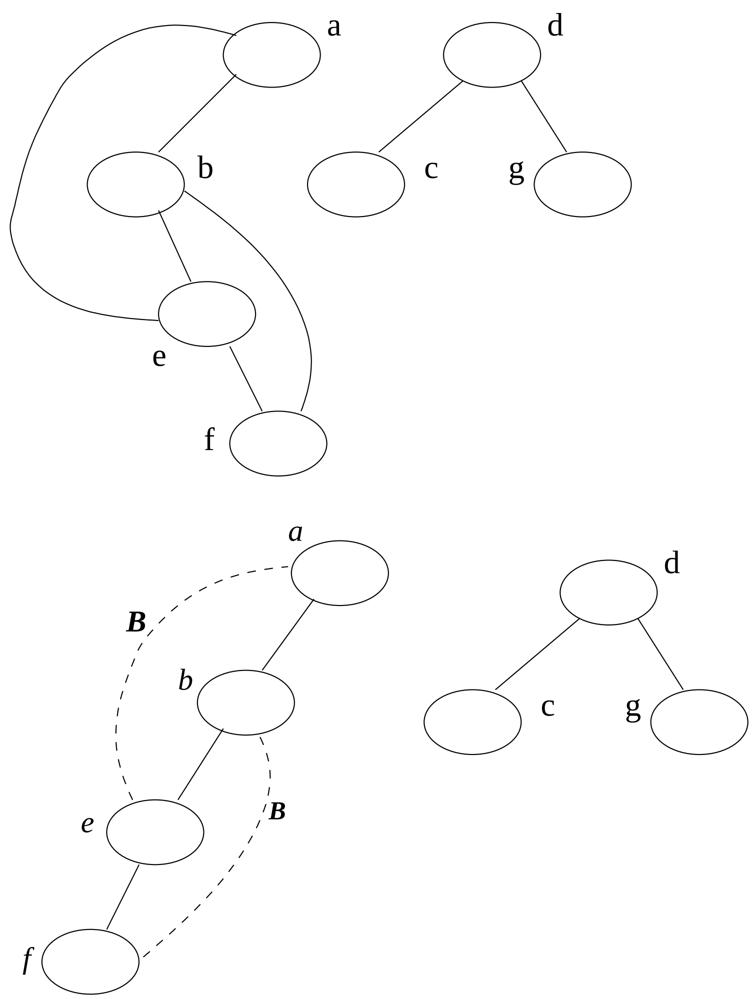
```
DFS{G} {
  for each v in V do // Initialize
      visit[v] = false;

  for each v in V do
      if (visit[v] == false) RDFS(v);
}

RDFS(v) {
  visit[v]=true;

  for each w in Adj(v) do
     if (visit[w] == false) {
       RDFS(w);
     }
  }
}
```

DFS Example



Given a graph $G = (V, E)$, it traverses all vertices of G and

constructs a forest (a collection of rooted trees), together with a set of source vertices (the roots).

Additional Information

- $discover[u]$ – the *discovery time*, a counter indicating when vertex u is discovered.
- $pred[u]$ – the predecessor of u, which discovered u.

```
DFS{G} {
  for each v in V do // Initialize
      visit[v] = false;
      pred[v] = NULL;
  time=0;
  for each v in V do
      if (visit[v] == false) RDFS(v);
}

RDFS(v) {
  visit[v]=true;
  discover[v] = ++time;
  for each w in Adj(v) do
     if (visit[w] == false) {
       pred[w]=v;
       RDFS(w);
       }
  }

}
```

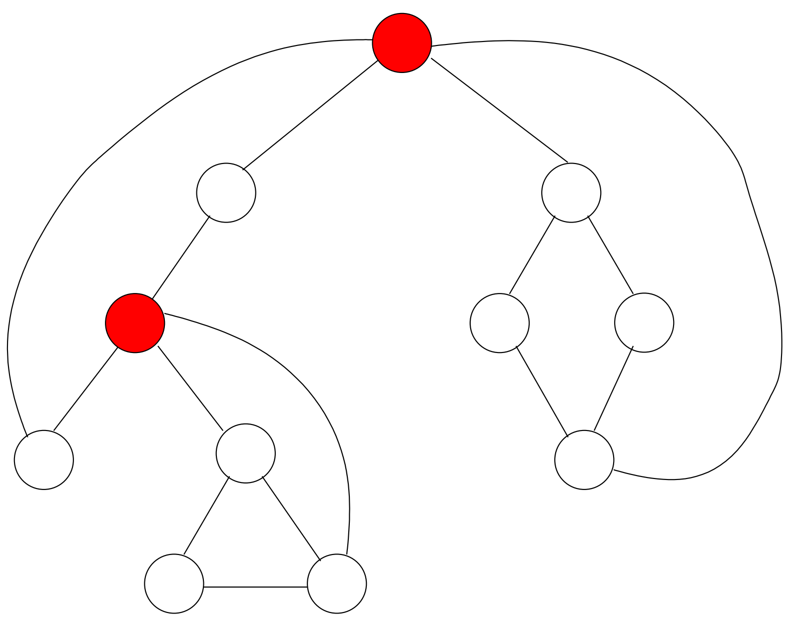
Classification of Edges

Tree edges: which are the edges $\{$*pred[v], v*$\}$ *where DFS calls are made.*

Back edges: *which are the edges* $\{$*u, v*$\}$ *where v is an ancestor of* u *in the tree.*

An Application of DFS : Articulation points

Definition : Let $G = (V, E)$ be a connected undirected graph. An articulation point (or cut vertex) of G is a vertex whose removal disconnects G.



Given a connected graph G, how to find all articulation points?

Articulation points: Easy solution

The easiest solution is to remove a vertex (and its corresponding edges) one by one from G and test whether the resulting graph is still connected or not (say by DFS). The running time is $O(V * (V + E))$.

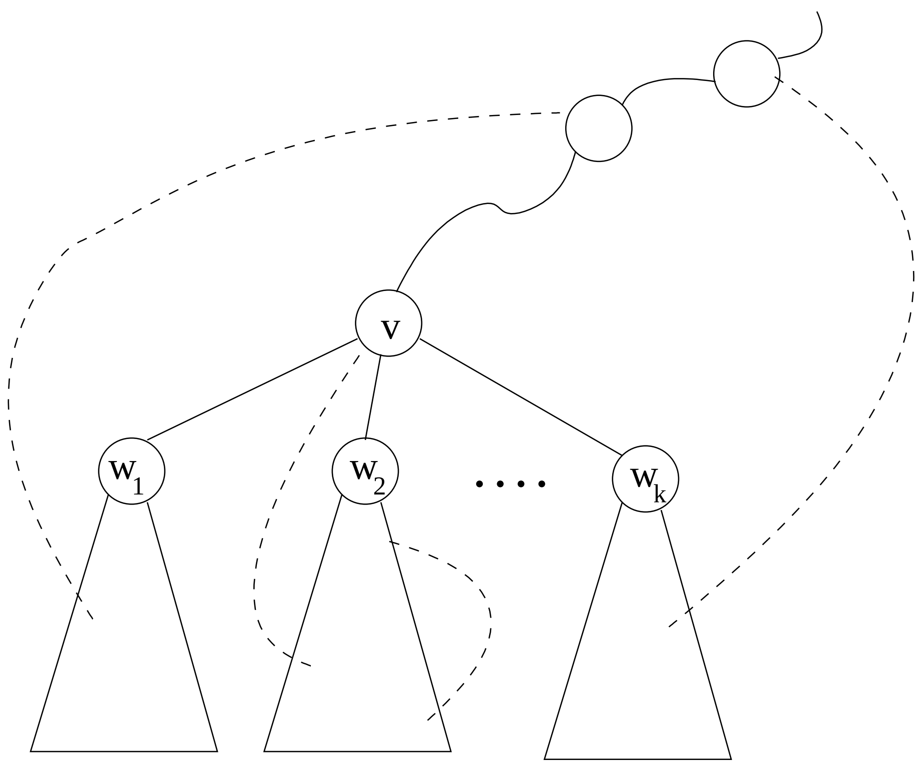
Articulation points: Observations

1. The root of the DFS tree is an articulation if it has two or more children.

2. Any other internal vertex v in the DFS tree, if it has a subtree rooted at a child of v that does NOT have an edge which climbs ’higher’ than v, then v is an articulation point.

Articulation points: How to climb up

Observe that for an undirected graph, it can only has tree edges or back edges. A subtree can only climb to the upper part of the tree by a back edge, and a vertex can only climb up to its ancestor.



Articulation points: Tackle observation 2

We make use of the *discovery time* in the DFS tree to define 'low' and 'high'. Observe that if we follow a path from an ancestor (high) to a descendant (low), the *discovery time* is in *increasing order*.

If there is a subtree rooted at a children of v which does not have a back edge connecting to a SMALLER discovery time than $discover[v]$, then v is an articulation point.

How do we know a subtree has a back edge climbing to an upper part of the tree ?

Define $Low[v]$ be the smallest value of a subtree rooted at v to which it can climb up by a back edge.

$Low[v] = min\{discover[v], discover[w]$: (u, w) is a back edge for some descendant u of $v\}$

```
RDFS_Compute_Low(v) {
  visit[v]=true;
  Low[v]=discover[v] = ++time;

  for each w in Adj(v) do
     if (visit[w] == false) {
       pred[w]=v;
       RDFS_Compute_Low(w);

       // When RDFS_Compute_Low(w) returns,
       // Low[w] stores the
       // lowest value it can climb up
       // for a subtree rooted at w.

       Low[v] = min(Low[v], Low[w]);

     } else if (w != pred[v]) {
             // {v, w} is a back edge
            Low[v] = min(Low[v], discover[w]);
     }
}
```

Articulation points

Articulation points are now determined as follows:

1. The root of the DFS tree is an articulation point if it has two or more children.

2. Any other internal vertex v in the DFS tree is an articulation point if v has a child w such that $Low[w] \geq discover[v]$.

```
 ArticulationPoints{
   for each v in V do
       if (pred[v] == NULL) {  //v is a root
           if (|Adj(v)|>1)
              articulation_point(v)=true;
       } else{
           for each w in Adj(v) do {
               if (Low[w] >= discover[v])
                  articulation_point(v)=true;
           }
      }
}
```

Running time = ?