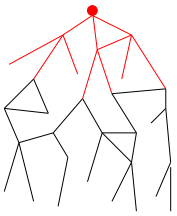
17—Depth First Search and Biconnectivity

May 20, 2002

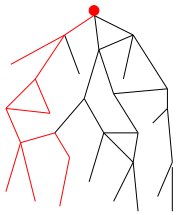
Deep and Wide

Breadth-First Search



Explore *across* before *down*

Depth-First Search



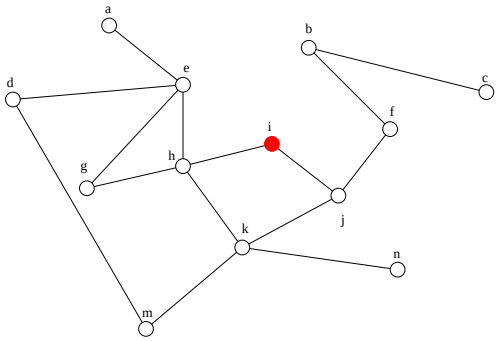
Explore *down* before *across*

Depth-First Search

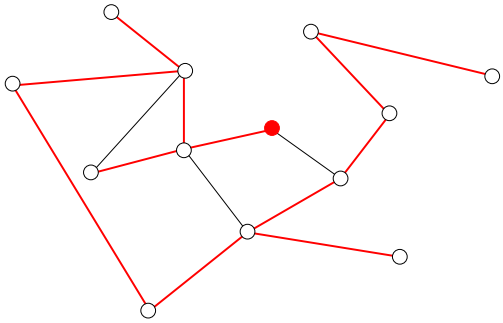
```
NumberDFS(Graph G, Vertex *root)
{
  for each (v in G) {
    Encountered(v) = false;
    Number(v) = -1;
  }
  int num = 1;
  RecursiveDFS(root, &num);
}
```

```
RecursiveDFS(Vertex *v, int *pn)
{
  Number(v) = (*pn)++;
  for each (w in v->Neighbors())
    if (!Encountered(w))
      RecursiveDFS(w, pn);
}
```

NumberDFS in Action

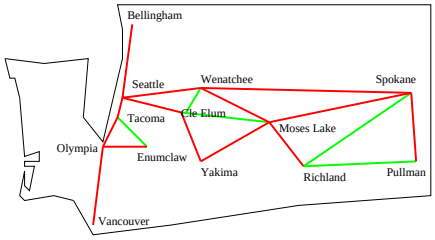


The DFS Tree



Biconnectivity

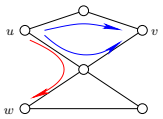
Network Fault-Tolerance



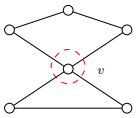
If we lose Wenatchee, can Seattle still talk to Spokane?

Biconnectivity

Two Equivalent Definitions



Two *vertex-disjoint* paths between any two distinct vertices

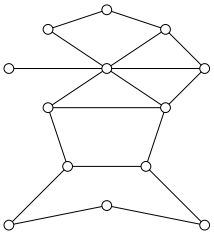


Connected and no *cutverticies*

- v a cutvertex if $(G - v)$ is disconnected

Testing Biconnectivity

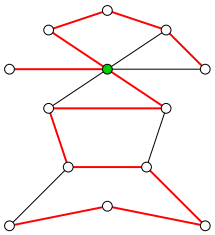
- Is G *connected*? How do you know?



G

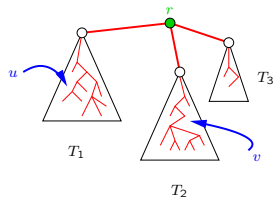
- Is G *biconnected*? How do you know?

Looking at DFS



If the root has more than one child, it is a cutvertex!

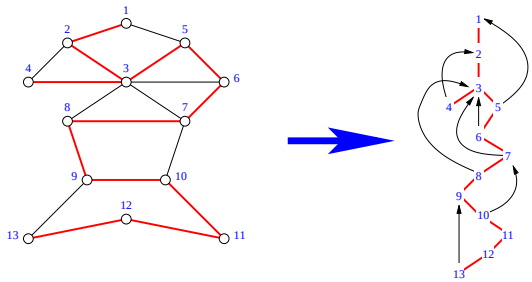
The Intuition



If we remove r, then there is no path between u and v

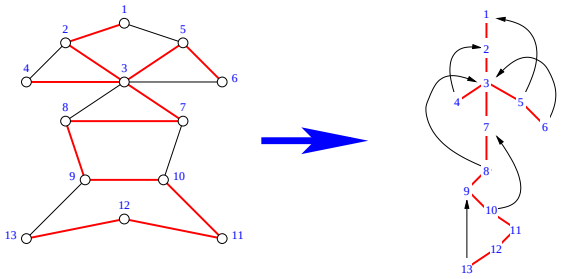
- Why?

The DFS Tree



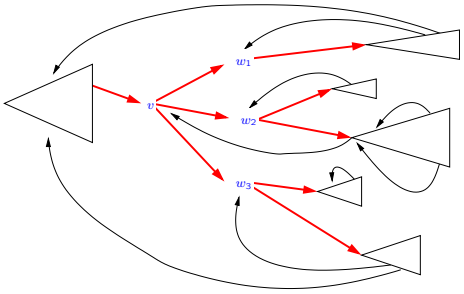
Red edges are *followed*; Black edges are *skipped*

DFS Tree for a Different Graph



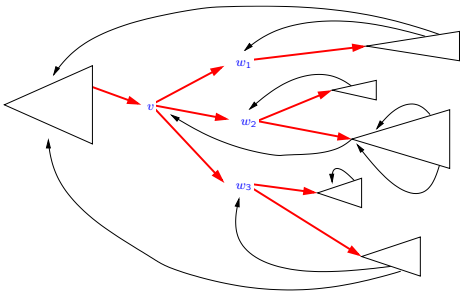
Which node is the cutvertex here?

The Lemma



v is a cutvertex $\Leftrightarrow$ *some* child of v has *no* skipped edges to an ancecstor of v

The Lemma, Right to Left

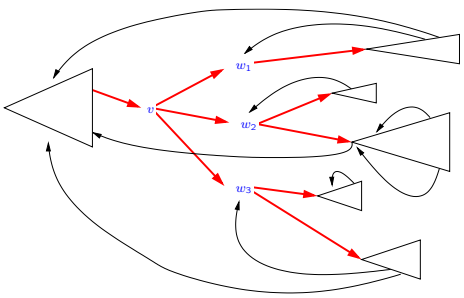


some child of v has *no* skipped edges to an ancecstor of v

$\Rightarrow$

v is a cutvertex

The Lemma, Left to Right

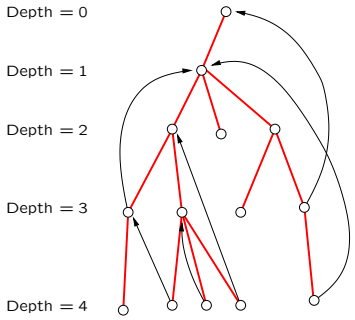


all children of v have skipped edges to an ancecstor of v

$\Rightarrow$

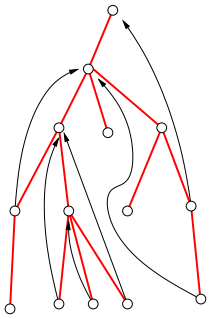
v is *not* a cutvertex

What We Need to Know



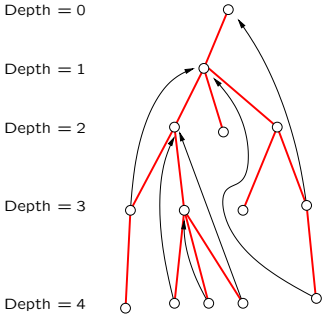
- v is an *ancestor* of w iff Depth(v) < Depth(w)
- w is an *descendant* of v iff Depth(w) > Depth(v)
- For those falling asleep: this DFS tree is impossible. Why?

Computing Depth



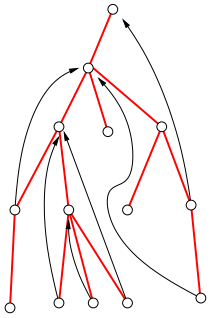
```
DepthDFS(Graph G, Vertex *root)
{
  for each (v in G) Depth(v) = -1;
  Depth(root) = 0;
  Recurse(root);
}

Recurse(Vertex *v)
{




}
```


Checking Skipped Edges




```
int MinDepthSeen(Vertex *w)
{




}

bool IsCutVertex(Vertex *v)
{




}
```

Doing Everything at the Same Time



```
FindCutVerticies(Graph G, Vertex *root)
{
  for each (v in G) Depth(v) = -1;
  Depth(root) = 0;
  Recurse(root);
  if (#(depth 1 nbrs of root) > 1)
    CutVertex(root) = true;
}

int Recurse(Vertex *v)
{
  int min_depth = Depth(v);
  for each (nbr w of v) {
    if (Depth(w) == -1) {
      Depth(w) = Depth(v) + 1;
      int d = Recurse(w);
      min_depth = min(min_depth, d);
      if (d >= Depth(v) and v != root)
        CutVertex(v) = true;
    } else
      min_depth = min(min_depth, Depth(w));
  return min_depth;
}
```


Running Time?

```
FindCutVerticies(Graph G, Vertex *root)
{
  for each (v in G) Depth(v) = -1;
  Depth(root) = 0;
  Recurse(root);
  if (#(depth 1 nbrs of root) > 1)
    CutVertex(root) = true;
}

int Recurse(Vertex *v)
{
  int min_depth = Depth(v);
  for each (nbr w of v) {
    if (Depth(w) == -1) {
      Depth(w) = Depth(v) + 1;
      int d = Recurse(w);
      min_depth = min(min_depth, d);
      if (d >= Depth(v) and v != root)
        CutVertex(v) = true;
    } else
      min_depth = min(min_depth, Depth(w));
  return min_depth;
}
```