Algorithmic Strategies 2024/25

Week 10 – Graph Algorithms

· U C ·

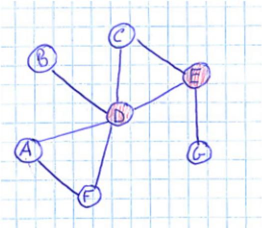
Universidade de Coimbra

Outline

1. Articulation Points
2. Strongly Connected Components

Introduction

Given a connected graph $G = (V, E)$, an articulation point is a vertex v in V that, if removed, it disconnects G, that is, there exists at least two vertices in $V \setminus \{v\}$ that are not connected by a path.

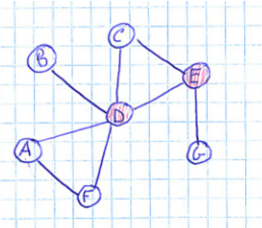


Vertices D and E are articulation points

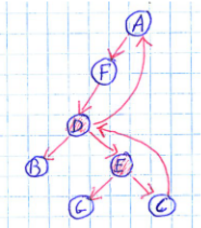
Simple approach: For each vertex $v \in V$, remove it and run DFS in the resulting graph to check its connectedness. This approach takes $O(|V| \cdot (|V| + |E|))$.

An Algorithm for Articulation Points

Perform a DFS traversal in G



A graph



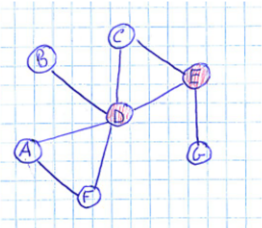
DFS tree

A vertex v is an articulation point if, in the DFS tree:

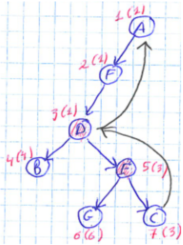
1. it is the root with two or more children
2. otherwise, it has a child w for which there is no backedge between w (or descendants) and a predecessor of v.

An Algorithm for Articulation Points

- $dfs[v]$ is the traversal index
- $low[v] = \min \begin{cases} dfs[v] & \\ dfs[x] & \text{for all } x = pred(v) \text{ with backedge} \\ low[w] & \text{for all } w = child(v) \end{cases}$



A graph



DFS tree - dfs (low)

A vertex (non-root) v is an articulation point if it has a child w such that $low[w] \geq dfs[v]$

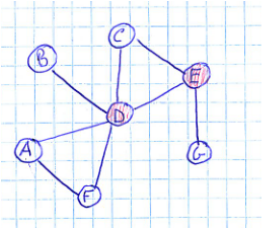
An Algorithm for Articulation Points

Function $AP(v)$

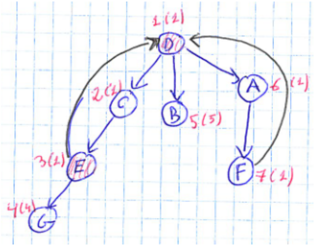
$low[v] = dfs[v] = t$
$t = t + 1$
for each edge $\{v, w\} \in E$ **do**
  if $dfs[w]$ has no value **then**
    $parent[w] = v$
    $AP(w)$
    $low[v] = \min(low[v], low[w])$
    if $dfs[v] = 1$ and $dfs[w] \neq 2$ **then**
      v is a (root) AP
    if $dfs[v] \neq 1$ and $low[w] \geq dfs[v]$ **then**
      v is a (non-root) AP
  else if $parent[v] \neq w$ **then**
    $low[v] = \min(low[v], dfs[w])$

It has complexity $O(|V| + |E|)$

An Algorithm for Articulation Points



A graph

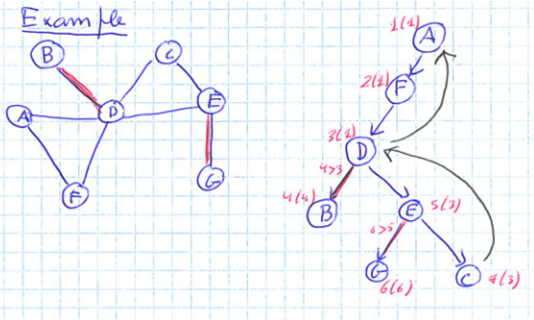


DFS tree - dfs (low)

The algorithm is invariant with respect to the starting vertex.

An Algorithm for Bridges

This algorithm can also be used to find bridges, that is, an edge that, if removed, disconnects the graph. An edge (w, v) is a bridge if $low[w] > dfs[v]$ in the DFS tree.



A graph

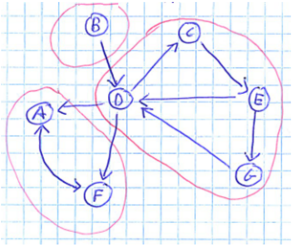
DFS tree - dfs (low)

Is there always a bridge between two articulation points?

Counter-example: the edge between the two articulation points is not a bridge.

Strongly Connected Components

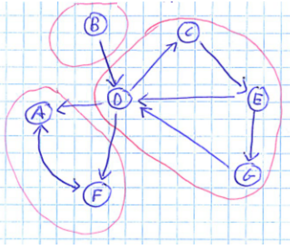
Given a directed graph $G = (V, A)$, a subgraph G' is a strongly connected component if there exists a path between a vertex and every other vertex in G' and this subgraph has maximal size.



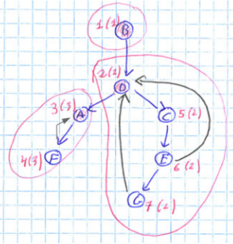
3 strongly connected components

Strongly Connected Components

A vertex v is the root of a connected component if $low[v] = dfs[v]$



A graph



DFS tree

The vertices in each strongly connected component under the root are stored in a stack. It is possible to solve it with Tarjan Algorithm in $O(|V| + |A|)$.

Strongly Connected Components

```
Function Tarjan(v)
    low[v] = dfs[v] = t
    t = t + 1
    push(S, v)
    for each arc (v, w) ∈ A do
        if dfs[w] has no value then
            Tarjan(w)
            low[v] = min(low[v], low[w])
        else if w ∈ S then
            low[v] = min(low[v], dfs[w])
    if low[v] = dfs[v] then
        C = ∅
        repeat
            w = pop(S)
            push(C, w)
        until w = v
        push(Scc, C)
```

Scc collects all stacks of strongly connected components. This algorithm must be called for every unvisited vertex in the graph.