VNU-HCM University of Science

Data Structure

CS163

ARTICULATION POINTS AND BRIDGES

23APCS02
Pham Gia Hung Khoa
Pham Nguyen Anh Tai

7th August 2024

Contents

1 Introduction

In graph theory, understanding the structural integrity of networks is crucial. Bridges are edges whose removal increases the number of disconnected components in a graph, acting as critical links. Articulation points are vertices that, if removed, also increase the number of disconnected components, serving as essential hubs. Identifying these elements helps improve network design and resilience, whether in communication networks, transportation systems, social networks, or biological systems. By pinpointing these vulnerabilities, we can create more robust and efficient networks across various domains.

2 Identifying Bridges and Articulation Points

2.1 Naive algorithm $O(NM)$

For each edge $e \in E$, remove it from the graph and count the number of connected components C', if $C' > C$ (the initial number of connected components), then we can claim that e is a bridge, otherwise it is not a bridge.

The similar algorithm for finding the articulation points, which tests each removal of the vertex v, whether it increases the number of connected components of the graph. Which results in a $O(N^2)$ time complexity.

Can we do better?

2.2 DFS Tree

To understand how we can effectively find the bridges and articulation points in terms of time complexity, we need to dive into the concept of DFS Tree.

Without loss of generality, let's assume that the graph G is connected. In general, DFS Tree is a spanning tree of the graph when we do the DFS traversal of the graph.

Consider an undirected connected graph G. Let's run a depth-first traversal of the graph. It can be implemented by a recursive function like this:

```
function visit(u)
    mark u as visited
    for each vertex v among the neighbours of u:
        if v is not visited:
            mark the edge u-v
            invoke visit(v)
```

[animation and demonstration]

This is the DFS tree of our graph:

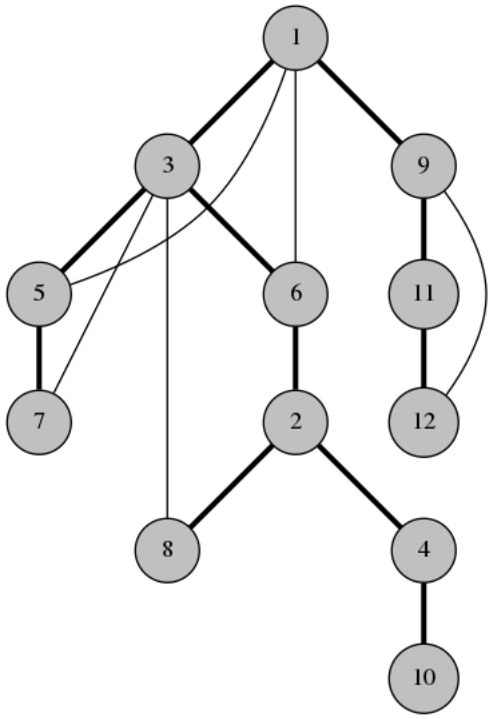

Figure 1: DFS Tree Illustration.

Let's look at all edges that were marked in line 5. They form a spanning tree of G, rooted at the vertex 1. We will call these edges *span-edges*, all other edges are *back-edges*.

Then we have the following observation:

Observation: The back-edges of the graph all connect a vertex with its descendant in the spanning tree. **This is why DFS tree is so useful.**

> **Why?** – Suppose that there is an edge uv, and without loss of generality the depth-first traversal reaches u while v is still unexplored. Then:
>
> - If the depth-first traversal goes to v from u using uv, then uv is a span-edge.
> - If the depth-first traversal doesn't go to v from u using uv, then v was already visited when the traversal looked at it at step 4. Thus it was explored while exploring one of the other neighbours of u, which means that v is a descendent of u in the DFS tree.

For example in the graph above, vertices 4 and 8 couldn't possibly have a back-edge connecting them because neither of them is an ancestor of the other. If there was an edge between 4 and 8, the traversal would have gone to 8 from 4 instead of going back to 2.

This is the most important observation about the DFS tree. The DFS tree is so useful because it simplifies the structure of a graph. Instead of having to worry about all kinds of edges, we only need to care about a tree and some additional ancestor-descendant edges. This structure is so much easier to think and write algorithms about.

2.3 How to find Bridges and Articulation Points

From this observation, we can come up with the following intuitive dynamic programming algorithm:

Let $dp[u] = \#$*of back-edges that connects v or its descendants to its ancestors*, then $dp[u]$ can be calculated using this formula:

$$\begin{aligned} dp[u] = {} & \#\ \textit{of back-edges going from u to its ancestors} \\ & - \#\ \textit{of back-edges going from u to its descendants} + \sum_{v \in child(u)} dp[v] \end{aligned}$$

3 Applications

4 Additional Insights

5 Reference