Algorithms, Design & Analysis

Introduction

Dr. Mudassir Shabbir

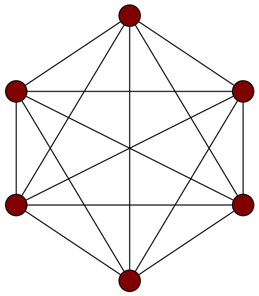
Information Technology University

March 04, 2025

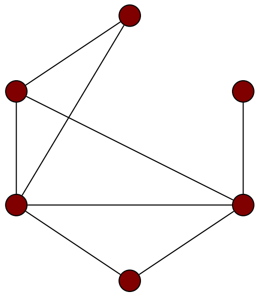
Lecture notes prepared by Jalal Ahmed and Ammar Khan.

Connectivity in Graphs

- Today, we are talking about connectivity in graphs.
- Following are examples of two connected graphs.



Graph (A)

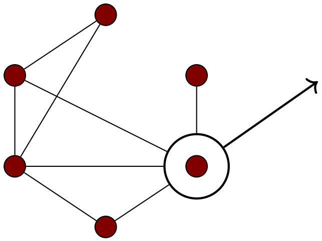


Graph (B)

Although both of these graphs are connected, intuitively they have different levels of connectivity.

Articulation point in graph G

Definition: In a connected component G, $\exists$ a vertex that if removed,disconnects increases the number of connected components of the remaining graph.
Example: Graph(B) in the last slide.

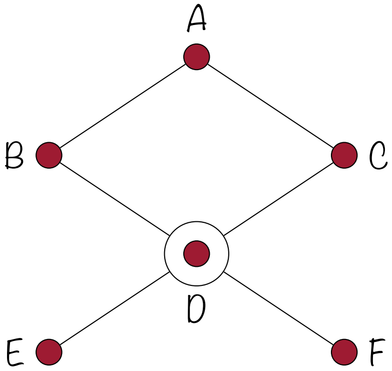


Note: If this vertex is removed, the graph will become disconnected.
Hence, it is an Articulation Point of Graph(B).

P.S. There is no articulation point in Graph(A) because removing any vertex from it would not disconnect the graph.

Continuation

For another Graph G below:



Vertex D is an articulation point. Removing it will disconnect the graph.

- Removing vertex D causes the graph to become disconnected, making it an articulation point.
- Other vertices are not articulation points because their removal does not

Real World Example

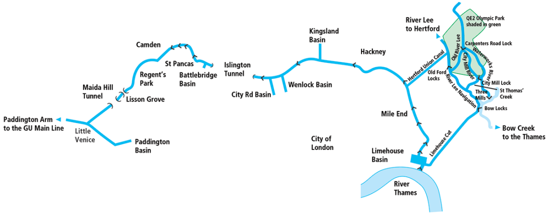
Consider critical infrastructure transmission networks of electricity containing single points of failure. These bottlenecks can disrupt entire systems when compromised.

Electrical Grid Vulnerability

A single transmission tower acting as bridge node between regional networks

Real World Example

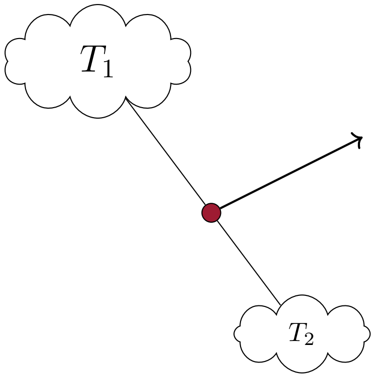
Bottlenecks create systemic risk - their failure partitions networks into disconnected components, halting flow and communication.



Waterway Chokepoint

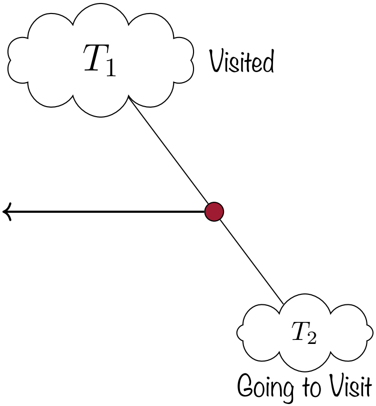
Critical canal section enabling continental shipping routes

Observation



This is an Articulation Point as no edge connects subgraph T_1 to subgraph T_2 (or vice versa).

In Computer Terminology

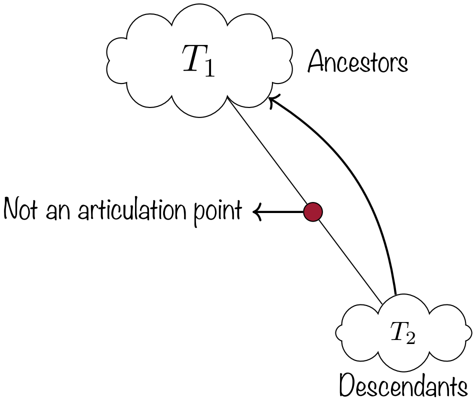


This is the only point connecting the two subgraphs T_1 and T_2.
If you need to go from T_1 to T_2, you must pass through this point.

We will use a modified version of DFS.

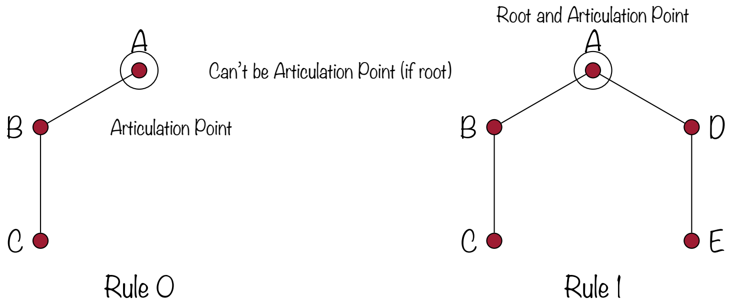
Informal Definition

Informally, an edge from descendants to ancestors means it is not an articulation point.



Checks

1. Root is an articulation point if it has multiple (> 1) children
2. For non-root vertices: Check if any child's low $\geq$ discovery time of current node

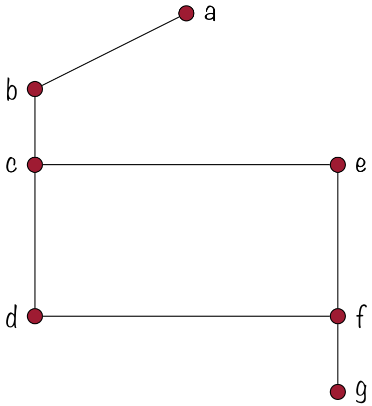

Root and Articulation Point
A
Can't be Articulation Point (if root)
B
Articulation Point
C
Rule 0
A
B
D
C
E
Rule 1

Children means unvisited neighbors

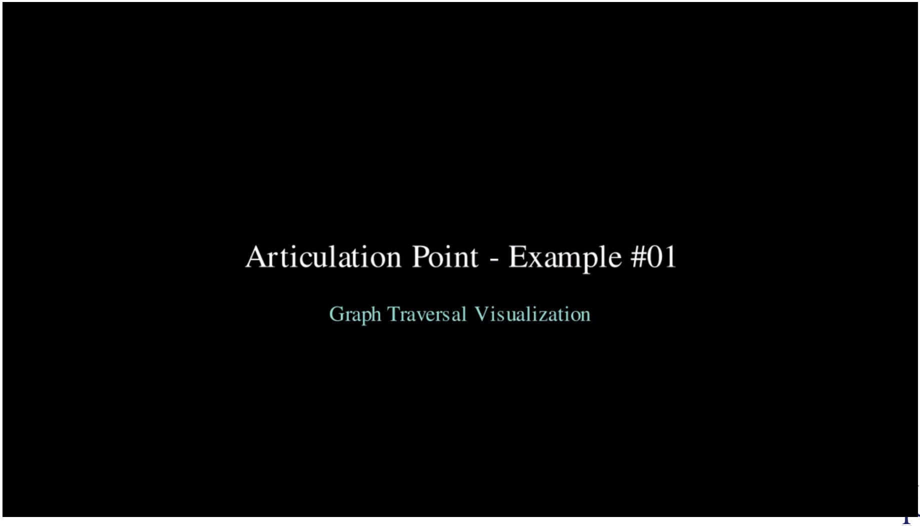
Example # 01

Let a Graph G:

Algorithm will start from (e)

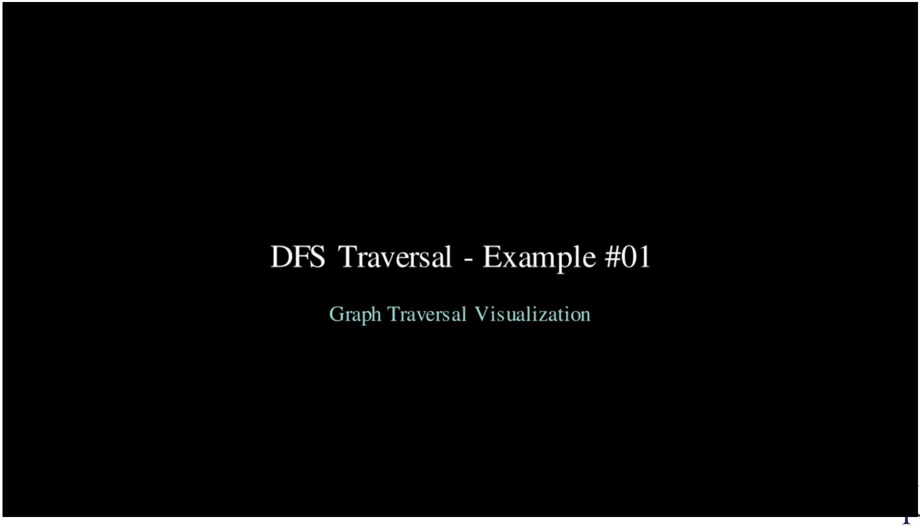


Example # 01 - Continue


Articulation Point - Example #01

Graph Traversal Visualization

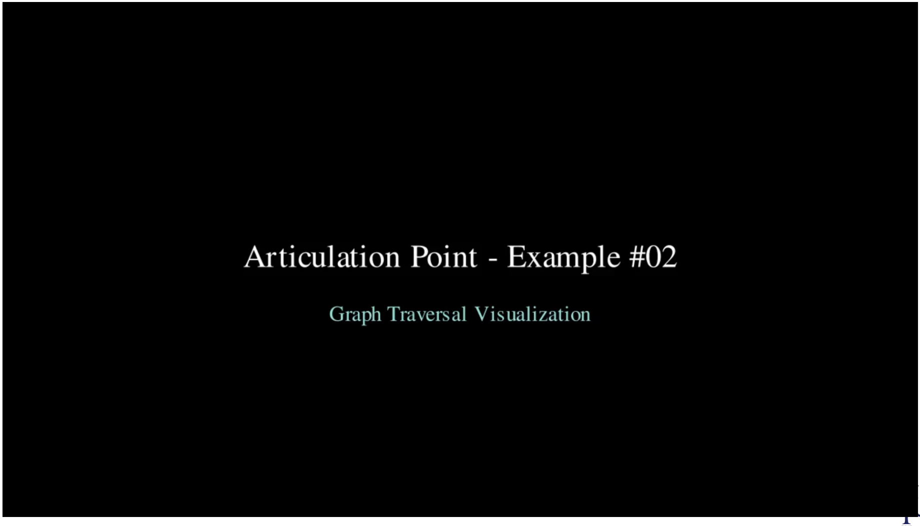
Example # 01 - DFS Graph traversal



DFS Traversal - Example #01

Graph Traversal Visualization

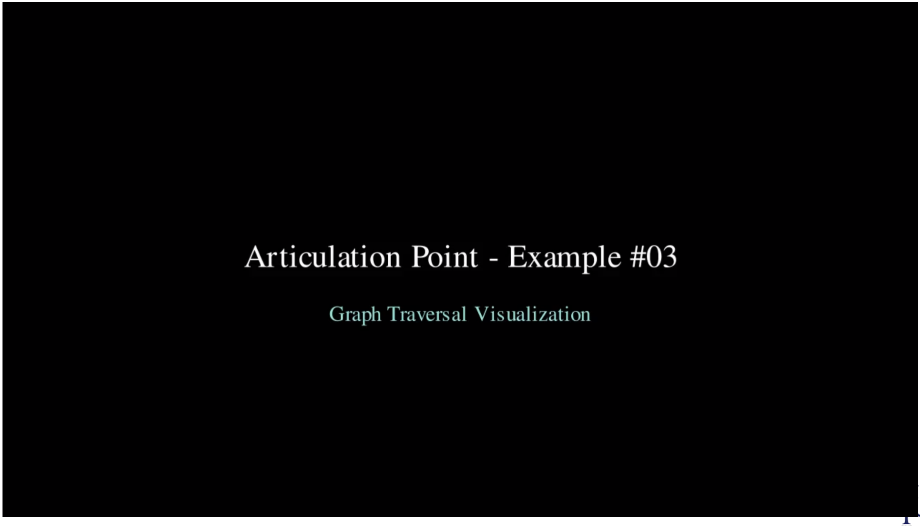
Example # 02



Articulation Point - Example #02

Graph Traversal Visualization

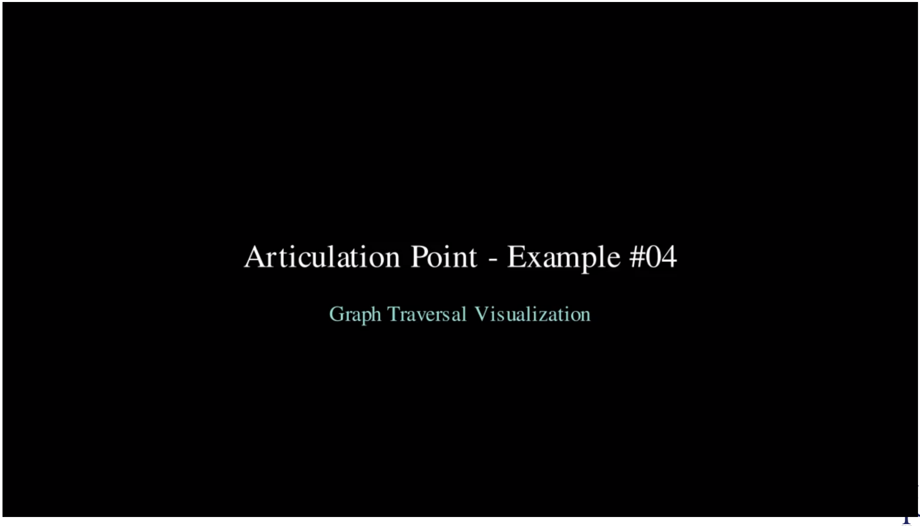
Example # 03



Articulation Point - Example #03

Graph Traversal Visualization

Example # 04



Articulation Point - Example #04

Graph Traversal Visualization

SCC - Strongly Connected Components

- Strongly Connected Components (SCC) are known as connected components in a undirected graph G.
- Removing articulation point(s) causes increase in the connected components of the graph G

Algorithm Design

How do you define articulation point if the given graph G is already disconnected?

- It's easy to check for Directed graph G, but for undirected graph G, let's say we are 50 level down and the current node has an edge with someone 50 levels up, that;s what we are going to do in our algorithm.
- There are many more directions to account for and in worst cases sometimes a vertex is discovered which connects the same path from the descendents graph to the ancestors graph, contradicting with the articulation point found and nullifying it.

And the way we are going to do is to keep a track of this by time.

- for each node, we will keep track of their top-level node (ancestor) by time.
- Now, designing our algorithm we will keep a specific order of the nodes discovered through a means called, Discovery time. Helping keep track of the top level node (ancestor).

Algorithm - Articulation Point

```
DFS($v$)
    $d(v) = 1, l(v) = 1$, flag = $F$
    for $\forall u \in N(v)$ (where $u$ is unvisited):
        IF flag == $F$:
            flag = $T$
        IF flag == $T$:
            $u$.articulation $= T$
```

Algorithm - Continue

WIP: To be continued next time

```
DFS($v$)
    $Time = 1$
    $d(v) = Time, l(v) = Time$, flag = $F$, $Time++$
    for $\forall u \in N(v)$ (where $u$ is unvisited):
      IF flag == $F$:
          flag = $T$
      IF flag == $T$:
          $u$.articulation $= T$
    IF $d(u) < d(v)$ AND $u \neq$ parent$(v)$:
       $l(v) = \text{MIN}(l(u), d(v))$
```

Note: $d(v)$ = Discovery Time, $l(v)$ = Least Discovery Time