

Biconnected Components

Definitions

K-Connectivity [K-vertex connectivity]

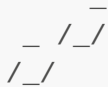
- Given a Graph $G = (V, E)$, if K is smallest size of subset of vertices to be deleted so that the graph becomes into more than 1 component, then graph G is K -connected

K-edge connectivity

- Given a Graph $G = (V, E)$, if K is smallest size of subset of edges to be deleted so that the graph becomes into more than 1 component, then graph G is K -edge connected

Notes on K-connectivity K-edge connectivity

- If Graph G is K -connected, then it is also K -edge connected
- If Graph G is K -edge connected, it is not necessary to be K -connected
- Example:



-

Articulation points

- Given a Graph $G = (V, E)$, a vertex u is called articulation point if removing u makes the the graph(component) into more than 1 component
- Articulation points are important in finding Bridges

Bridges

- Given a Graph $G = (V, E)$, a bridge is a edge that connects two vertices of G and when deleted makes the graph into more than 1 component [disconnects the graph]

Back edge

- It is an edge (u, v) such that v is ancestor of node u but not part of DFS tree. Edge from 6 to 2 is a back edge.

Finding articulation Points [Tarjan's Algorithm]

- Obtain a rooted DFS tree (T) of the graph G rooted at vertex R
- While doing DFS compute the discovery time(and if wanted finishing times also) of each vertex

Case:Root R

- If R has more than 1 children Then R is a articulation point

Case:Non - Root vertex

- For each non root vertex v find the $low(v)$

$low(v)$:

1. $low(v) = d(v)$
2. $low(v) = \min(low(v), (low(w)))$ for all w which a child of v
3. $low(v) = \min(low(v), (low(x)))$ for all x where there exists a backedge from v to x

Articulation point condition:

1. For a given root vertex v if one of children's $low(w)$ is greater than or equal to $d(v)$ then v is an articulation point
2. Condition: $d(v) \leq low(w)$ for atleast one child w than v is a articulation point,
3. In other words $d(v) \leq \max(low(w))$ then v is a articulation point.

How to compute this:

1. First do a DFS find the rooted tree with its discovery times
2. Bottom up do the $low(v)$ calculation
3. Now for each node check the articulation point condition.

Finding Bridges from this:

1. Bridges have at least one end-point as an articulation point.
2. For a edge $vw(v \rightarrow \text{parent}, w \rightarrow \text{child}) \Rightarrow$ If $d(v) < low(w)$ then vw is a bridge

finding connected components

- Lets take a u, v, w , such that u is the child of v and v is child of w
- if v is articulation point due to u , then uv and vw belong to different components
- Else both belong to same biconnected components

Tarjan-Vishkin Algorithm

1. Take a spanning tree T of graph G
2. Find the preorder number of vertices in T
3. Create an auxiliary graph G' where G' contains one vertex for each edge of G
4. Add edges to G' using the cases below
5. Now find connected components of G' using the algorithm discussed in the previous section
6. Now a component of G' is a biconnected component of G

Finding edges in G'

1. Case a: Add a Edge connecting uw to vw in G' whenever there is a tree edge $w \rightarrow u$ with u as the parent of w and a nontree edge vw with $\text{pre}(v) < \text{pre}(w)$
2. Case b: Add a Edge connecting uv to xw in G' whenever there is a tree edge $v \rightarrow u$ with u as the parent of v and a tree edge $w \rightarrow x$ with x as the parent of w and a nontree edge vw with v and w not having an ancestor-descendant relationship in T
3. Case c: Add a Edge connecting uv to vw in G' whenever there is a tree edge $v \rightarrow u$ with u as the parent of v and a tree edge $w \rightarrow v$ with v as the parent of w and a nontree edge joins a descendant of w to a non-descendant of v in T

Cong and Bader Preprocessing Addition to Tarjan-Vishkin Algorithm

- We identify edges of G the removal of which does not affect the biconnected components of G
- Let T be a **BFS tree** of G . Let F be a spanning forest of $G \setminus T$. Then, then the edges of $G \setminus (T \cup F)$ are not relevant for the biconnectivity of G .
 - **BFS tree** - The tree obtained while doing BFS on the graph
 - **spanning forest** - The collection of trees obtained while finding spanning tree of a graph (basically the graph is not connected)
 - $G \setminus T$ - Edges that are in G but not in T
 - $T \cup F$ - Edges that are in T or F
 - $G \setminus (T \cup F)$ - Edges in G but not in T or F .
- Let H be a graph obtained after removing edges, then $|E(H)|$ is at most $2n$.
 - $E(X)$ - Edges in X
 - $|E(X)|$ - Number of edges in X
- The property extends to k -connectivity also due to the results of **Cheriyán and Thurimell**.

Notes

- approach of Cong and Bader is therefore very useful for dense graphs
- $|E(G)|$ can go down from m to $O(n)$.
- For sparse graphs, there is not much to gain as m is usually very small to begin with.
- The size of $E(G')$ in the Tarjan and Vishkin algorithm can be as high as $O(n^2)$ even when G is sparse

Slota and Madduri Algorithm

1. Perform a BFS of G .
2. For each tree edge uv , with $u = p(v)$, remove the edge from G and perform another BFS from v .
 - $p(x) \rightarrow$ Parent on vertex x
3. If this BFS on $G \setminus \{uv\}$ can reach to some vertex that is an ancestor of u , then v is not an articulation point.

- $A \setminus B$ - Set of items in A but not in B

Finding whether it possible to reach to some vertex that is an ancestor of u , when uv is removed from G

1. Consider a BFS tree of G stored along with two other pieces of information: P and L
2. Basically we do bfs from each vertex X removing(masking) the edge connecting vertex X and $P(X)$, if we are able not able to reach a vertex Y with $L(Y) < L(X)$ then $P(X)$ is an articulation point.
 - $P(X)$ - Parent of X
 - $L(X)$ - Level of X
 - Alt: Vertex ' Y ' is articulation point if for atleast one children w , when masked BFS is done we are not able reach a vertex Z with $L(Z) < L(w)$
3. Different ways of expressing the claim:
 - A non-root vertex v in the BFS tree $\langle P, L \rangle$ is an articulation vertex if and only if it has at least one child w that cannot reach any vertex of depth at least $L(v)$ when v is removed from G
 - A non-root vertex v in the BFS tree $\langle P, L \rangle$ is not an articulation vertex if and only if all its children w in the BFS tree ($P(w) = v$) can reach all other vertices in the graph G when v is removed from G
 - $P(w) = v$ - Parent of w in the BFS tree
 - If a traversal from any $u_i \in V, P(u_i) = v$ is not able to reach all other $u_j \in G, P(u_j) = v$ when v is removed from the graph, then v is an articulation point
4. if the only path in G between u_i and u_j requires v , then u_i and u_j are in separate biconnected components with v as an articulation point. We term v as the parent articulation vertex

Handling Root:

- One way is doing BFS rooted at a different vertex

Notes:

- +ve : Each BFS can be done in parallel without any dependencies on the other BFS.
- -ve : The downside is that one has to do additional BFS computations.
- Surprisingly, works better than the Cong and Bader approach as in principle, Tarjan and Vishkin algorithm is slow in practice.

Chaitanya and Kothapalli

Observations:

1. We notice that in a 2-edge-connected component articulation points are necessarily vertices that are the least common ancestor of some nontree edge according to any spanning tree

Algorithm

1. We Separate Vertex into Potential articulation points and Non-articulation points by using observation-1
 - Potential articulation points: - Vertices that are the least common ancestor of some nontree edge
 - Non-articulation points: - Vertices that are not the least common ancestor of any nontree edge
- Now two approaches Possible:
2. Approach 1: Check articulation point status only for vertices in potential articulation points set while doing Slota and Madduri
3. Approach 2:
 1. Find the Bridges and remove the bridges ,so we have components which are two edge connected.
 2. We transform each two edged connected component (G_i) into a new graph using a novel way (G_i')
 3. Case Root: Vertex r is an articulation point in G if only if (iff) r is the LCA of more than one non-tree edge of G_i' according to a BFS in G_i' from r , and r is also an end point of some bridge in G_i'
 4. Case Non Root: For vertices u in G_i' with $u \neq r$, u is an articulation point of G iff u is an end point of some bridge uv in G_i' with $u \in G$ and $v \notin G$.
 - Note u belongs to G
 - Note v does not belong to G
5. Novel Modification:
 1. For a non tree edge e , if v is LCA, with fundamental cycle of e passing through vx and vy , then:
 - Remove edges vx and vy
 - Add a vertex v'
 - Add an edge $v'x-v'y$
 - x, y two children of v in the fundamental cycle of e