CS5800: Algorithms Spring 2018
Assignment 6.2

Saptaparna Das

April 2, 2018

Algorithm:

```
Let the islands are representated by Nodes in a Graph and the bridges connecting
them are edges.
The graph is represented by adjacency list i.e. array of linkedlists where each slot
 in an array indicates a vertex and all the verices,
counnected to the vertex in form of edges, are stored in linklist.

To get articulation point, the basic idea is to perform DFS on the graph.
For any vertex, if it is a root of the DFS tree and it has atleast two independent children
or if it is not a root but it has child  such that no vertex in subtree of child has backedge
parent or any ancestor of parent.


Let alreadyVisited be an array indicating if a parcular node is visited in dfs traversal.
Let firstSeenTime be an array indicating the time, a vertex is first discovered.
Let parentVertex be an array which stores the parent or source vertex of
 a vertex in the graph.
Let backEdge be an arraywhich keeps track of the backedges for a vertex.
Let time indicates time of dicovery of each vertex and starts with 0.
Let criticalNodes be an array which holds if a vertex is critical i.e.
removing it will disconnect islands/nodes.

Step 1: Initialize alreadyVisited, criticalNodes and parentVertex array all vertices in
        the graph with values false, false and null respectively.
Step 2: For each vertex u, if it's not already visited, do the following:
        Step a: Store adj as zero and Mark it as visited by setting corresponding
                 value in alreadyVisited array as true.
        Step b: increase time and Initalize corresponding entry in backEdge array and
                 firstSeemTime array with value as time.
        Step c: Get all the adjacent verticex of current vertex. For each of them (v),
                 do the following:
                Step c1: If the vertex v is not already visited, do the following:
                        Step c1.1: increase adj by 1 and Set vertex u as parent of v in
                                   parentVertex
                        Step c1.2: Recursively go back to Step a for current vertex v
                        Step c1.3: Set backEdge[u] as minimum of
                                   backEdge[u] and backEdge[v]
                        Step c1.4: If u is root vertex and adj is greater than 1
                                   set corresponding entry in criticalNodes
                                   array to be true.
                        Step c1.5: If u is not root and backEdge[v] is greater or equals
                                   firstSeenTime[u], set corresponding entry in
                                   criticalNodes array to be true.
                Step c2: If v is not prent vertex for u, update backEdge[u] by
                           taking minimum of backEdge[u] and firstSeenTime[v]
Step 3: Print the critical nodes from criticalNode array.
```

Analysis:

The graph is represented by adjacency list. Since we are basically doing DFS operation, it visits all vertices exactly once. For any vertex then it has to traverse the linkedlist of edges

So, the time complexity is $O(|V| + |E|)$. where V is vertex and E is edge. Space complexity is also $O(|V| + |E|)$ for DFS (for adjacency list) and to store the additional data we need 3 arrays of size V. So, total space complexity is $O(|V| + |E|)$.