EECS 336 Fall 2015
Homework Problem 1.4

This algorithm finds whether there is cycle in given G. Provided function BFS takes input of undirected graph G and a vertex r as root, and returns search tree T with depth and pointers to parent and children of each vertex u, denoted as $u.Parent$ and $u.Child$ respectively.

Algorithm 1 Check whether G has cycle

```
1: procedure CHECK-CYCLE(G, r)
2:     Find tree traversal T ← BFS(G, r)
3:     Initialize Cycle ← false
4:     for each vertex u in T do
5:         Initialize LegalEdge as empty list
6:         if u.Parent, p, exists then
7:             Append edge (p, u) to LegalEdge
8:         end if
9:         for each u.Child, c do
10:            Append edge (u, c) to LegalEdge
11:        end for
12:        for each edge e incident to u in G do
13:            if e is not in LegalEdge then
14:                return Cycle = true
15:            end if
16:        end for
17:    end for
18:    return Cycle
19: end procedure
```

Runtime analysis:

BFS takes $O(n' + m')$ for a component of G with n'-node m'-edge . For n-node m-edge G, in this alg., each vertex is appended to list $LegalEdge$ $2n$ times (one as parent, one as child), which is $O(n)$. Let n_u denote the degree of vertex u, then time spent within the For loop considering edges incident to u is $O(n_u)$, so total over all vertices is $O(\sum_{u \in G} n_u)$, which is $O(m)$. Thus total runtime is $O(m + n)$.

Correctness:

a) If algorithm returns Yes, then G has a cycle.

Proof: By contradiction. Suppose when alg. returns Yes, G does not has cycle. Alg. only returns true when there is some edge, e, that connects u with a vertex that is neither its parent, p, nor any of its children, c_i. Three cases are possible:

(a) u is connected with an ancestor, a, which is not p. Then there is a polygon that connects at least u, a, and p. Thus, G has a cycle.

(b) u is connected with a descendant, d, which is not c. Then there is a polygon that connects at least u, d, and c. Thus, G has a cycle.

(c) u is connected with a vertex on another branch, z, which is neither an ancestor or descendant of u. Then there is a polygon that connects at least u, r, and z. Thus, G has a cycle.

In any case, G has a cycle, which is contradictory to supposition. Thus, if alg. returns true, then G has a cycle.

b) If alg. returns No, then G does not have cycle.

By contradiction: suppose if alg. returns No, G has a cycle. Alg. returning false leads to the fact that no vertex in T has an edge that connects with another vertex other than its parent or child. Thus, all vertices have pointers only to their parents or children. Following the pointers from the root or leaves, these vertices would form a search tree. By definition, search tree does not have a cycle. This contradicts with supposition. Thus, if alg. returns No, then G does not have a cycle.