ASSIGNMENT 6

DUE: DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies.

1. [12 marks] **2-neighbours.** The *degree* of a vertex \( v \) in an undirected graph \( G = (V, E) \) is \( d(v) = |\{u \in V : (u, v) \in E\}| \), the number of vertices in the adjacency list of \( v \). Define two other “extended degree” measures of \( v \) as follows:

\[
s(v) = \sum_{u : (u, v) \in E} d(u) \quad \text{and} \quad d_2(v) = |\{w \in V : w \neq v \text{ and } \exists u \in V \text{ s.t. } (u, v) \in E \text{ and } (u, w) \in E\}|.
\]

(i) [2 marks] Give an example to show that \( s(v) \) can be different from \( d_2(v) \).

(ii) [4 marks] Give a linear-time algorithm to compute \( s(v) \) for all vertices \( v \). Assume that the graph is given as adjacency lists and that addition costs \( O(1) \).

(iii) [4 marks] Give an \( O(n^3) \) time algorithm to compute \( d_2(v) \) for all vertices \( v \). Assume that the graph is given as adjacency lists and that addition costs \( O(1) \).

(iv) [2 marks] Give a very simple \( O(n) \) time algorithm to compute \( d_2(v) \) for all vertices \( v \) in the case where the input graph is a tree. Assume that the graph is given as adjacency lists and that addition costs \( O(1) \).

**Note:** Do not use BFS or DFS for any of your solutions in parts (ii)–(iv). (It may be possible to use BFS or DFS to solve these problems, but the resulting algorithms will be more cumbersome than a straight-forward algorithm.)
2. [10 marks] **BFS/DFS.** Use BFS or DFS to solve the following two graph problems.

(i) [5 marks] Given an undirected graph $G$ and a vertex $s$ in $G$, determine the minimum value $k \geq 1$ such that for every vertex $v$ in $G$, there is a path of length at most $k$ from $s$ to $v$ in $G$. If no such value $k$ exists, output ⊥.

(ii) [5 marks] Given an undirected graph $G$ and an edge $e = (s, t)$ in $G$, determine if there is a cycle in $G$ that includes the edge $e$.

**Note:** Recall that the basic outlines for BFS and DFS (with an explicit stack) are as follows:

**Algorithm 1: BFS($G, s$)**

| for each $v \in V(G)$, visited[$v$] $\leftarrow$ False; visited[$s$] $\leftarrow$ True; Q.enqueue($s$); while Q is not empty do $v \leftarrow$ Q.dequeue(); for each $w \in G$.adjList($v$) do if !visited[$w$] then Q.enqueue($w$); return; |

**Algorithm 2: DFS($G, s$)**

| for each $v \in V(G)$, visited[$v$] $\leftarrow$ False; visited[$s$] $\leftarrow$ True; S.push($s$); while S is not empty do $v \leftarrow$ S.pop(); for each $w \in G$.adjList($v$) do if !visited[$w$] then S.push($w$); return; |

For both problems,

- You must modify either the BFS or DFS algorithm above to solve the problem. (In particular, all the lines in the base BFS/DFS algorithm should be as-is in your final algorithm, except the last Return statement which should be modified to return some value).
- The total time complexity of the algorithm must remain $O(n + m)$.
- Your algorithm should not revisit the entire BFS/DFS tree at the end to generate its output.