University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020
Yiliang Wang

January 29, 2020

Homework 3 Problem 1

Problem 1 (10 points):

Suppose that an n-vertex undirected graph $G = (V, E)$ has vertices s and t with the distance from s to t strictly greater than $n/2$. Show that there must exist some vertex v, not equal to either s or t, such that deleting v destroys all $s-t$ paths. (This could be phrased as: show that a graph with *diameter* strictly greater than $n/2$ has an *articulation point*.) Give an algorithm that finds v in $O(n + m)$ time, you can assume that you are given the vertices s and t that have separation of greater than $n/2$.

Answer:

Define the set of nodes in all path between vertex s and t is $\{S\}$.
Since $|V| = n$, then $|\{S\}| \leq n-2$, And since $\forall s-t$ path p, $|p| \geq n/2+1$, which means there are at least $n/2+1$ levels, assume $\forall$ level l, each level has 2 parallel nodes, then $|\{S\}| = n+2 \geq n-2$, this conflic with $|\{S\}| \leq n-2$. Therefore, $\exists$ level l that there are less than 2 nodes in level l. And since only single level at level l, deleting v would break all of the node in that level, and this break all path between $s-t$.

Algorithm:
(using Breadth-First Search)

```
# g is nested list representation of adjacent list
# g[i] equal to node i's neighbors
def find_articulation_point:
    ans = list()
    visited = [0 for i in range(n)]
    queue = list()
    counter = 0
    # use queue to construct a bfs, iterate through the graph,
    # and find any level with single node
    for node in range(len(g)):
        if visited[node] == 0:
            queue.append(node)
            while len(queue)!= 0:
                # only iterate through one level
                for i in counter:
```

```python
            cur = queue.pop(0)
            visited[counter] += 1
            for neighbor in g[cur]:
                if visited[neighbor] == 0:
                    queue.append(neighbor)
        count = len(queue)

        # if the set of node on this level contain only one
        # node, add the node to articulation point list
        if count == 1:
            ans += queue
    return ans
```

Proof of correctness: This algorithm use Breadth-First Search (hence the time complexity is $O(n+m)$) to check any level with single node, given the proof of above, any level with single node would be a articulation point of this graph.