Analysis of Algorithms Homework 1

Joseph Petitti

November 7, 2019

1. Functions ordered by ascending order of growth rate (*i.e.* the last item grows the fastest):
   - $f_7(n) = n^{1/\log n}$
   - $f_2(n) = \sqrt{2n}$
   - $f_3(n) = 10 + n$
   - $f_6(n) = n^2 \log(n)$
   - $f_1(n) = n^{2.5}$
   - $f_4(n) = 10^n$
   - $f_5(n) = 10^{2n}$

2. Proof that G being a tree implies that it has exactly $n-1$ edges:

   If G has fewer than $n-1$ edges then it cannot be connected by the handshake lemma. If G has more than $n-1$ edges:

   - Suppose G has more then $n-1$ edges but is not a tree.
   - Because G is connected but not a tree, it must contain a cycle.
   - Remove an edge from the cycle. We are left with a connected graph with $n-2$ edges, which is impossible by the handshake lemma.

   Proof that G being connected and having exactly $n-1$ edges implies it is a tree:

   - Suppose G is connected and has exactly $n-1$ edges but is not a tree.
   - We know G is connected, so to not be a tree it must contain a cycle.
   - In any connected graph with a cycle, you can remove one edge from the cycle and the graph will still be connected.
   - Remove one edge from the cycle in G. We are left with a connected graph with $n-2$ edges, which is impossible by the handshake lemma.

3. **Efficient Algorithm**: Breadth-first search (assume the graph is given in the adjacency list representation):

```
Choose a starting node s
Set Discovered[s] = true
Set Discovered[v] = false for all other v in G
Initialize L[0] to consist of the single element s
Set the layer counter i = 0
```

```
While L[i] is not empty
        Initialize an empty list L[i+1]
        For each node u in L[i]
                Consider each edge (u, v) incident to u
                If Discovered[v] = false then
                        Set Discovered[v] = true
                        Add v to the list L[i + 1]
                Else
                        Return true
                Endif
        EndFor
        Increment the layer counter i by 1
endWhile
Return false
```

Proof of Correctness: This algorithm, which is essentially just breadth-first search, goes through each node in G by "layer," where each layer n is the set of nodes exactly n distance away from starting node s. This algorithm always moves "down" the graph, if you encounter a node twice it means there is a loop.

Analysis of Running Time: This algorithm runs in $O(m+n)$ time, because it is really just breadth-first search. There are m edges and n nodes, and in the worst case you iterate over each of them once.

4. (a) Assuming there is a node $v \in V$ with degree$(v) = 1$ (*i.e.* v is a leaf node):

    - Because the graph is connected, the one node that v is connected to is also connected to the rest of the graph.
    - Therefore if you remove v the rest of the graph will still be connected.

    If there is no node $v \in V$ with degree$(v) = 1$ (*i.e.* no leaf nodes):

    - If all nodes $v \in V$ have degree$(v) > 1$ then the graph must contain a cycle.
    - In a set of nodes S, where S is a cycle, removing a single node will result in a sequence of connected nodes.
    - Therefore, you can remove any node from the cycle and the graph will still be connected.

   (b) Proof that for every strongly connected directed graph $G = (V, E)$ there exists a vertex $v \in V$ such that removing v from G results with a strongly connected graph.

    - In a strongly connected graph, every node is reachable from every other node by definition.
    - Let S be the set of set of all vertices in G except for v.
    - All nodes in S are reachable by all other nodes in S, because S is a subset of G.
    - If v is removed from the G, you are left with S, which is still strongly connected.

5. Proof that the existence of an edge that crosses every cut (S, T) implies that G is connected:

   - Suppose G is connected, and there exists a cut (S, T) with no edge $e \in E$ that crosses it.
   - Therefore no edge connects a node in S with a node in T.
   - If none of the nodes in S are connected to any node in T then there must exist some node $s \in S$ that is not connected to some node $t \in T$.
   - Therefore G is not connected, contradicting our assumption.

   Proof that G being connected implies an edge crosses (S, T):

   - If G is connected, then there are no two disjoint subsets of nodes in G that are not reachable from each other.
   - Therefore, for any cut (S, T) that separates G into two disjoint subsets, some node $s \in S$ must be able to reach some node $t \in T$.
   - Therefore the cut must cross the edge between s and t.