In-Class Activity #2

Vancouver Summer Program

Algorithms

---

1. The following is a purported proof of the statement "Every tree with n vertices has $n-1$ edges."

   Proof. We show this by induction. For $n = 1$, the graph consists entirely of an isolated vertex, so it has zero edges.

   Suppose that for some $m \geq 1$, every tree with m nodes has $m-1$ edges. Let T be a tree with m nodes. Then T has $m-1$ edges by the induction hypothesis. Adjoin a new vertex (not in T) to T and connect this vertex by an edge to any other vertex. Then this new graph is a tree and has $m+1$ vertices and m edges.

   □

   Is this proof correct? If not, pinpoint where the argument went wrong and provide a correction.

2. Design an algorithm that takes as input an undirected, connected, n-vertex graph that contains exactly one cycle, and return a spanning tree in time $O(n)$.

3. Here's a proposal for how to find the length of the shortest cycle in an undirected graph with unit edge lengths.

   When a back edge, say (v, w), is encountered during a depth-first search, it forms a cycle with the tree edges from w to v. The length of the cycle is $\texttt{level}[v] - \texttt{level}[w] + 1$, where the `level` of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:

   - Do a depth-first search, keeping track of the level of each vertex.
   - Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

   Does this strategy always work? If it does, prove its correctness, and it does not give a counterexample and devise another algorithm to find the length of the shortest cycle in a graph.