CSC505

Liam Adams

January 27 2019

1 Unique1 Analysis

The algorithm is implemented in unique1.cpp in the function *getLongestSubsequence*. Prior to calling that function, a function called *readInput* scans the file and processes it into a vector. Processing the file takes $O(n)$ time.
There are 3 nested for loops in *getLongestSubsequence* - inside each of which comparisons, assignments, and arithmetic operations are performed in constant time. In the worst case scenario, each loop can run n times. Therefore the worst case running time is $O(n^3)$.
The n^3 term dominates the runtime as n approaches infinity, so the overall worst case running time of the program is $O(n^3)$

2 Unique2 Analysis

The algorithm is implemented in unique2.cpp in the function *getLongestSubsequence*. Prior to calling that function, a function called *readInput* scans the file and processes it into a vector. Processing the file takes $O(n)$ time.
There are 2 nested for loops in *getLongestSubsequence* - inside each of which comparisons, assignments, and arithmetic operations are performed in constant time. Inside the second loop a search on a balanced binary tree is performed, which takes $O(lg(n))$ time. In the worst case scenario, each loop can run n times. Therefore the worst case running time is $O(n^2 lg(n))$.
The n^2 term dominates the runtime as n approaches infinity, so the overall worst case running time of the program is $O(n^2 lg(n))$