Comp2007 - Assignment 1

Ashwin Ramesh

August 5, 2012

1 Problem 1

1.1 Overview of Problem 1

Polynomial calculation can be a very expensive computation to run and therefore efficiency is integral when creating algorithms to solve this problem. This problem introduces two algorithms. Algorithm 1 uses a naive approach, recalculating the new power of x on every iteration. On the other hand, Algorithm 2 uses Horner's rule by calculating the new power by using the previous power of x. Below is the asymptotically tight analysis of both algorithms.

1.2 Algorithm Analysis

Upper Bounds: $T(n)$ is $O(f(n))$ if there exists constants $c > 0$ and $n_0 \geqq 0$ such that for all $n \geqq n_0$, we have $T(n) \leqq c * f(n)$.

Lower Bounds: $T(n)$ is $\Omega(f(n))$ if there exists constants $c > 0$ and $n_0 \geqq 0$ such that for all $n \geqq n_0$, we have $T(n) \geqq c * f(n)$.

Tight Bounds: $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

1.3 Algorithm 1 - NAIVE(x,A) Analysis

- Line 3: $O(n)$ time as it is an iteration from 0 to n
- Line 4: $O(1)$ time
- Line 5: $O(n)$ time as i varies from 1 to n
- Line 6/7: $O(1)$ time

Upper Bound Time Complexity of Naive Approach: $O(n) * O(n) = O(n^2)$

Lower Bound Time Complexity of Naive Approach: similarly the lower bound will be $\Omega(n^2)$

This means that the **Tight Bound Complexity** will be $\Theta(n^2)$

1.4 Algorithm 2 - HORNER(x,A) Analysis

- Line 3: $O(n)$ time as it is an iteration from 0 to n
- Line 4: $O(1)$ as it uses the result from above to calculate addition

Upper Bound Time Complexity of Naive Approach: $O(n) * O(1) = O(n)$

Lower Bound Time Complexity of Naive Approach: similarly the lower bound will be $\Omega(n)$

This means that the **Tight Bound Complexity** will be $\Theta(n)$