CMPSC 465 – Spring 2021 — Solutions to Homework 2

Josiah Kim, SID 948500821, `juk483`

February 4, 2021

1. Acknowledgements

(a) I did not work in a group.

(b) I did not consult with any of my group members.

(c) I did not consult any non-class materials.

2. Compare growth rates

(a) Polynomials grow according to their exponents. Since $1.5 > 1.3$:

$$f = \Omega(g)$$

(b) Exponentials grow according to their base. The base is 2 in both cases.

$$f = \Theta(g)$$

(c) f is an exponential while g is a polynomial.

$$f = \Omega(g)$$

(d) f is $O(3^n)$ and g is $O(2^n)$.

$$f = \Omega(g)$$

(e) f is $O((logn)^100)$; g is $O(n^0.1)$

$$f = \Omega(g)$$

(f) Polynomials grow slower than logs.

$$f = \Omega(g)$$

(g) Factorials grow faster than exponentials.

$$f = O(g)$$

(h) $f = log(e^n) = nlog(e)$. f is $O(n)$ and g is $O(nlogn)$.

$$f = O(g)$$

(i) f is $O(n)$, g is $O((logn)^2)$

$$f = O(g)$$

(j) f is $O(n)$, g is $O(n)$

$$f = \Theta(g)$$

3.Polynomials and Horner's Rule

(a) The first multiplication is $a * x$. The second is $a * x * x$. The third is $a * x * x * x$. The number of multiplications are dependent on n and its previous n-number of multiplications which can be translated to $\frac{n(n-1)}{2}$.

The first addition is $(a * x) + (a * x * x)$. The second addition is $firstSum + (a * x * x * x)$. The last sum will always be $nSum + a_0$. Therefore, the number of additions can be represented by $n + 1$.

Therefore, $T(n) = \frac{n(n-1)}{2} + n + 1$ which gives us the time complexity of $O(n^2)$

(b) LOOP INVARIANT:
$i = 3$: $z = zx_0 + a_3$
$i = 2$: $z = (zx_0 + a_3)x_0 + a_2$
$i = 1$: $z = ((zx_0 + a_3)x_0 + a_2)x_0 + a_1$
$i = 0$: $z = (((zx_0 + a_3)x_0 + a_2)x_0 + a_1)x_0 + a_0$
By expanding, we get: $z = zx_0^4 + x_0^3 a_3 + x_0^2 a_2 + x_0^1 a_1 + a_0$
After j iterations, we get: $z = \sum_{j=0}^{i-1} x_0^j a_j$

INITIALIZATION:
We will assume that $a_n \subset \mathbb{N}$.
Before the loop, set $n = 4$ so $z = a_4 = 4$.
This should equal: $4 + 3x_0^3 + 2x_0^2 + 0$
$z = x_0^4 + x_0^3 a_3 + x_0^2 a_2 + x_0^1 a_1 + a_0$
True.

MAINTENANCE:
We will assume LI holds for i. Need to prove it holds for $i + 1$.
z_{after} = val of z after $i + 1$ iterations
$z_{after} = z_{before} x_0 + a_i =$
(By LI)
$= (\sum_{j=0}^{i-1} x_0^j a_j)x_0 + a_i = \sum_{j=0}^{i-1} x_0^{j+1} a_j + a_i = \sum_{j=0}^{i} x_0^j a_j = \sum_{j=0}^{j_{after}-1} x_0^j a_j$
The last a_0 will always be 0.
True.

TERMINATION:
At termination, z returns with the value after n loops: $x_0^n a_n + ... + x_0 a_1 + a_0)$
This is the same format as $p(x_0)$.
True.

(c) This algorithm is similar to the brute-force method in the fact that it will wait until the end to calculate the sum. However, the product is also calculated at the end.
When $i = 3$, z returns: $(((zx_0 + a_3)x_0 + a_2)x_0 + a_1)x_0 + a_0$
There are n-number of additions and products (assuming a_0 is always zero).
Therefore, the numbers of sums can be represented by $O(n)$ and the number of products can also be represented by $O(n)$. Therefore, the time complexity results in $O(n)$.

4.Solving Recurrences

(a) Branching Factor: 2
Height: $log_2(n)$
Size of Subproblems at Depth k: $(\frac{n}{2^k})^{\frac{1}{2}} = \frac{\sqrt{n}}{2^{\frac{k}{2}}}$
Number of Subproblems at Depth k: 2^k
$T(n) = \sum_{k=1}^{log_2 n} 2^k * 1 * \frac{\sqrt{n}}{2^{\frac{k}{2}}}$
$T(n) = \sqrt{n} * \sum_{k=1}^{log_2 n} \frac{2^k}{2^{\frac{k}{2}}}$
$T(n) = \sqrt{n} * \sum_{k=1}^{log_2 n} 2^{\frac{k}{2}}$
$T(n) = \sqrt{n} * \frac{\Theta(2^{\frac{1}{2}log_2 n})}{\Theta(\sqrt{n})}$
$T(n) = \Theta(2^{\frac{1}{2}log_2 n})$
$T(n) = \Theta(\sqrt{n})$

(b) Branching Factor: 2
Height:$log_3(n)$
Size of Subproblems at Depth k: $(\frac{n}{3^k})^0 = 1$
Number of Subproblems at Depth k: 2^k
$T(n) = \sum_{k=0}^{log_3 n} 2^k * 1 * 1$
$T(n) = \sum_{k=0}^{log_3 n} 2^k$
$T(n) = \frac{\Theta(2^{log_3 n})}{\Theta(1)}$
$T(n) = \Theta(2^{log_3 n})$

(c) Branching Factor: 5
Height: $log_4 n$
Size of Subproblems at Depth k: $\frac{n}{4^k}$
Number of Subproblems at Depth k: 5^k
$T(n) = \sum_{k=0}^{log_4 n} 5^k * 1 * \frac{n}{4^k}$
$T(n) = n \sum_{k=0}^{log_3 n} \frac{5^k}{4^k}$
$T(n) = n(\frac{5}{4})^{log_4 n}$
$T(n) = n * \frac{\Theta((\frac{5}{4})^{log_4 n})}{\Theta(n)}$
$T(n) = \Theta((\frac{5}{4})^{log_4 n})$

(d) Branching Factor: 6
Height: $log_7 n$
Size of Subproblems at Depth k: $\frac{n}{7^k}$
Number of Subproblems at Depth k: 7^k
$T(n) = \sum_{k=0}^{log_7 n} 7^k * 1 * \frac{n}{7^k}$
$T(n) = n \sum_{k=0}^{log_7 n} \frac{7^k}{7^k}$
$T(n) = \Theta(1)$

(e) Branching Factor: 9
Height: $log_9 n$
Size of Subproblems at Depth k: $(\frac{n}{3^k})^2 = \frac{n^2}{3^{2k}} = \frac{n^2}{9^k}$
Number of Subproblems at Depth k:9^k
$T(n) = \sum_{k=0}^{log_9 n} 9^k * 1 * \frac{n^2}{9^k}$

$T(n) = n^2 \sum_{k=0}^{log_9 n} 1$
$T(n) = n^2 * \frac{\Theta(1)}{\Theta(n^2)}$
$T(n) = \Theta(1)$