Sean Kunz

CS 575 HW #1

1. Prove the following using the original definitions of O, Ω, θ, and ω.

1.1. 15n2 + 5n + 8 = O(n2)

g(n) is O(f(n)) if for positive constants c and N, 0 <= g(n) <= cf(n) for all n >= N

g(n) = 15n2 + 5n + 8

f(n) = n2

0 <= 15n2 + 5n + 8 <= cn2

0 <= 15 + 5/n + 8/n2 <= c (Divide both sides by n2)

For N = 1, c = 28, 0 <= 28 <= 28

Expression is satisfied.

1.2. n3 + 2n + 5 = Ω(n3)

g(n) is Ω(f(n)) if for positive constants c and N, 0 <= cf(n) <= g(n) for all n >= N

g(n) = n3 + 2n + 5

f(n) = n3

0 <= cn3 <= n3 + 2n + 5

0 <= c <= 1 + 2/n2 + 5/n3 (Divide both sides by n3)

For N = 1, c = 8, 0 <= 8 <= 8

Expression is satisfied.

1.3. = Ω(nk) for k > 1. Assume that ai > 0 for all i.

g(n) is Ω(f(n)) if for positive constants c and N, 0 <= cf(n) <= g(n) for all n >= N

g(n) =

f(n) = nk

0 <= cnk <=

0 <= cnk <= a(1-nk)/(1-n) (Geometric Series)

0 <= cnk <= a/(1-n) - ank/(1-n) (Separate terms)

0 <= c <= a/nk(1-n) - a(1-n) (Divide both sides by nk)

For N = 2, k = 1, a = 1, c = ½, 0 <= ½ <= ½

Expression is satisfied.

1.4. n7 + 3n2 = θ(n7)

g(n) is θ(f(n)) if for positive constants c, d, and N, 0 <= cf(n) <= g(n) <= df(n) for all n >= N

0 <= cn7 <= n7 + 3n2 <= dn7

0 <= cn7 <= n7 + 3n2

0 <= c <= 1 + 3/n5

For N = 1, c = 4, 0 <= 4 <= 4

Expression is satisfied.

n7 + 3n2 <= dn7

For N = 1, d = 4, 4 <= 4

Expression is satisfied.

2. Prove the following using limits.

{c then f(n) = θ(g(n)) if c > 0

= {0 then f(n) = o(g(n))

{∞ then f(n) = ω(g(n))

2.1. nk = o(3n) where k >= 0

f(n) = nk

g(n) = 3n

(L’Hopital’s)

= = … (Continue taking derivative)

= 0

2.2. n ln n7 = o(n3)

f(n) = n ln n7

g(n) = n3

= 0 (L’Hopital’s)

3. Prove that 7n -1 is divisible by 6 for n = 1, 2, 3, … using induction

m mod n = m - n floor(m / n)

Base: For n = 1

7n - 1 = 71 - 1 = 6. 6 - 6 floor(1) = 0. Satisfied.

Hypothesis: Assume true for (n+1).

Induction: 7n+1 - 1 is divisible by 6.

7\*7n - 1 - (7n - 1). If we subtract a multiple of 6 from a value, we should get another multiple of 6.

6\*7n is a multiple of 6 as it is multiplied by 6.

4. Analyze the worst-case time complexity in terms of Big Oh notation. Briefly, yet clearly explain your answer.

4.1. O(lg(n)). By instruction count, we see that lg(n) + 1 instructions are performed in the loop due to the statement i=2\*i.

4.2. O(n lg(n)). By instruction count, the outer for loop executes n times. The three function calls have time complexities of O(1), O(lg(n)), and O(1), respectively. Since each of these are called n times, the highest order term is then n lg(n).

4.3. O(2s), where s is the number of bits for a value n. When the value computed for a factorial gets large enough, it will no longer fit in a word and multiplication will no longer be a constant time operation.