Swetha Jayapathy

Student ID : 934041047

Instructor : Julianne Schutfort

January 15th, 2020

CS 325 HW 1 – 30 points

1) (4 pts) For each of the following pairs of functions, select the best relationship from the options:

f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) is Θ(g(n))

a. $f(n) = n^{0.25}$; $g(n) = n^{0.5}$

$\lim_{n\to\infty} n^{0.25}/ n^{0.5 = 0}$

f(n) is O(g(n))

b. $f(n) = \log n^2$; $g(n) = \lg n$

$\lim_{n\to\infty} (\log n^2)/\lg n$

$\lim_{n\to\infty} (2\log n)/\lg n$ = goes to infinity

f(n) is Θ(g(n))

c. $f(n) = \log \log n$ $g(n) = \log n$

$\lim_{n\to\infty} (\log \log n)/\log n$

f(n) is Ω(g(n))

d. $f(n) = 5000n^3 + n^2$ $g(n) = 0.000001\, n^4$

$\lim_{n\to\infty} (5000n^3 + n^2)/(0.000001\, n^4)$ -> Denominator grows faster = 0

f(n) is O(g(n))

e. $f(n) = n\log n + n$; $g(n) = n\sqrt{n}$

f(n) is O(g(n))

f. $f(n) = e^n$; $g(n) = 2^n$

$\lim_{n \to \infty} (e/2)^n$

f(n) is Ω(g(n))

g. $f(n) = 2^n$; $g(n) = 2^{n+1}$

$\lim_{n \to \infty} 2^n / 2^{n+1} = ½$

f(n) is Θ(g(n))

h. $f(n) = n^n$; $g(n) = n!$

$\lim_{n \to \infty} n^n / n! = \infty$

f(n) is Ω(g(n))

2) (6 pts) Determine the theoretical running time of the following algorithms. Use theta notation and give a brief explanation.

a.

```
int Algo1(int n)
{
    int sum = 0;
    for (int i = n; i > 0; i--) {
        for (int j = i+1; j <=n; j++) {
            sum = sum + j;
            cout << i << " " << sum << endl;
        }
    }
}
```

The first for loop runs for n times

The second for loop also runs for n times, lets take n = 5, The second loop fails for the first time, but then runs upto n times after i gets decremented. And it takes constant time for the sum and Print statements.

Therefore, the Run time complexity is $\Theta(n^2)$

b.

```
int Algo2(int n)
{
    int total = 0;
    for (long int i = 2; i <= n; i=i*i) {
        total = total + 1;
        cout << i << " " << total << endl;

    }
}
```

We can see that i value is doubled for each loop as below

$i^2 <= n$

$=O(\sqrt{n})$

The other statements are run for a constant time, hence it is negligible.

Thereby we would get 2^k ,since it is raised to the powers of 2, we can write it as Θ (lg n) c.

```
int Algo3(int n, int m)
{
    int total = 0;
    int sum = 0;
    for (int i = 1; i <= n; i +=2 ) {
        total = total + 1;
        cout << i << " " << total << endl;
    }
    for (int j = 1; j <= m; j++ ) {
        sum = sum + 1;
        cout << j << " " <<sum << endl;
    }

    for (int i = 1; i <= n; i++ ) {
        for (int j = 1; j <= m; j++ ) {
            sum = sum + 1;
            cout << i << j << " " <<sum << endl;
        }
    }

}
```

- The first for loop runs for n^2 times, the inner statements takes constant time, therefore we can say the complexity of this as n^2
- The second for loop runs m times
- The third for loop runs n*m times

Hence we would get a polynomial as $n^2+(n*m)+m$

Therefore the run time complexity is $\Theta(n^2)$.