CSCI-570: Homework # 2

Due on Friday, September 12 , 2014

Saket Choudhary
skchoudh@usc.edu
2170058637

Contents

HW1

(2: Ch#2 Ex#3)

Part (a) n^2
Doubling the input size make it slower by $\frac{(2n)^2}{n^2} = 4$
Consider increasing input size by 1: $\frac{(n+1)^2}{n^2} = \frac{n^2+2n+1}{n^2} = 1 + \frac{1}{n} + \frac{1}{n^2}$
For $\lim_{n\to\infty}$, $1 + \frac{1}{n} + \frac{1}{n^2} = 1$, Thus the algorithm with $n+1$ input is as slow as with input size n for $n \to \infty$.
Part (b): n^3
Doubling the input size: $\frac{(2n)^3}{n^3} = 8$, thus it is 8 times slower.
Increasing the input size by 1: $\frac{(n+1)^3}{n^3} = \frac{n^3+3n^2+3n+1}{n^3} = 1 + \frac{3}{n} + \frac{3}{n^2} + \frac{1}{n^3}$.
For $\lim_{n\to\infty}$, $1 + \frac{3}{n} + \frac{3}{n^2} + \frac{1}{n^3} = 1$.
Part (c): This is similar to Part(a), since the factor of 100 is common. The solution is exactly similar to Part(a)
Part (d): $nlogn$
Doubling the input size: $\frac{2nlog2n}{nlogn} = \frac{2log2n}{logn}$. For $\lim_{n\to\infty}$, this would blow up.
Increasing the input size by 1: $\frac{n+1log(n+1)}{nlogn}$.
For $\lim_{n\to\infty}$, $\frac{n+1log(n+1)}{nlogn} = 1$.
Hence input with $n+1$ is as slow as n $\lim_{n\to\infty}$
Part (e): 2^n
Doubling the input size: $\frac{2^{2n}}{2^n} = 2^n$, which blows up as $\lim_{n\to\infty}$.
Increasing the input size by 1 : $\frac{2^(n+1)}{2^n} = 2$.
Thus increasing the input size by 1 causes it to be 2 times slower.

(3: Ch#2 Ex#4)

Given: Operating speed = 10^10 operations per second.
To Find: Maximum possible n, for 3600s operation
Part (a): n^2
$n^2 = 36 * 10^12 \implies n = 6 * 10^6$
Part (b): n^3
$n^3 = 36 * 10^12 \implies n = (36)^0.333 * 10^4$
Part (c): $100n^2$
$100n^2 = 36 * 10^12 \implies n = 6 * 10^5$
Part (d): $nlogn$
$log(n^n) = 36 * 10^12$
Part (e): 2^{2^n}
$2^{2^n} = 36 * 10^12 \implies n = log_2(log_2(36 * 10^12))$

(4: Ch#2 Ex#5)

$f1 = n^2.5, f2 = \sqrt{2n}, f3 = n + 10, f4 = 10^n, f5 = 100^n, f6 = n^2 logn$
Consider the square of f_i:

$f1' = n^5; f2 = 2n; f3 = (n + 10)^2; f4 = 10^{2n}; f5 = 100^{2n}; f6 = n^4(logn)^2$
Since exponentials always grow faster than polynomials, in the order of running time complexity higher to lower:
$100^2n > 10^2n > n^5 > n^4(logn)^2 > (n + 2)^2 > n2n$
Thus:
$f5 > f4 > f1 > f6 > f3 > f2$

(5: Ch#2 Ex#6)

(6: Ch#3, Ex#2)

(7: Ch#3, Ex#6)