Christopher Bass

CECS 302 50

Assignment 2

10/07/14

Christopher Bass
9-28-14
Assignment # 2

2.1) Order the following functions by growth rate:
$\cancel{N}$, $\cancel{\sqrt{N}}$, $\cancel{N^{1.5}}$, $\cancel{N^2}$, $\cancel{N \log N}$, $\cancel{N \log \log N}$, $\cancel{N \log^2 N}$,
$\cancel{N \log(N^2)}$, $\cancel{2/N}$, $\cancel{2^N}$, $\cancel{2^{N/2}}$, 37, $\cancel{N^2 \log N}$, $\cancel{N^3}$

From least to greatest growth rate:

$2/N$, 37, $\sqrt{N}$, N, $N \log \log N$, $N \log N$,
$N \log(N^2)$, $N \log^2 N$, $N^{1.5}$, N^2, $N^2 \log N$,
N^3, $2^{N/2}$, 2^N

$N \log N$ + $N \log(N^2)$ grow at the same rate.

2.3) Which function grows faster: $N \log N$ or $N^{1+\varepsilon/\sqrt{\log N}}$, $\varepsilon < 0$?

$n \log n \approx n^{1+\frac{\varepsilon}{\sqrt{\log N}}}$ $\quad \varepsilon > 0$

$n \log n \approx n \cdot n^{\varepsilon/\sqrt{\log N}}$

$\log n \approx n^{\varepsilon/\sqrt{\log n}}$ $\quad \varepsilon > 0$

smaller $\qquad$ faster

$N^{1+\varepsilon/\sqrt{\log N}}$ grows faster

2.4) Prove that for any constant K, $\log^K N = o(N)$

$\log_b^{k_1} N = o(\log_b^{k_2} N)$, if $k_1 < k_2$

$\log_b^i N = o(N)$

$$(\log_b^i N)' = ((\log_b N)^i)'$$

$$= i \cdot (\log_b N)^{i-1} \cdot \frac{\log_b e}{N}$$

$$N' = 1$$

$$\lim_{N \to \infty} \frac{(\log_b^i N)'}{N'} = \lim_{N \to \infty} i \cdot (\log_b^{i-1} N) \cdot \frac{\log_b e}{N}$$

$$= \lim_{N \to \infty} \frac{i \cdot \log_b e \cdot \log_b^{i-1} N}{N}$$

$$= i \cdot \log_b e \lim_{N \to \infty} \frac{\log_b^{i-1} N}{N}$$

$$\lim_{N \to \infty} \frac{\log_b^{i-1} N}{N} = 0$$

$$\lim_{N \to \infty} \frac{(\log_b^i N)'}{N} = 0$$

$$\lim_{N \to \infty} \frac{\log_b^i N}{N} = 0$$

$\log_b^i N = o(N)$

2.6.) In a recent Court Case, a judge Cited a City for Contempt & ordered a fine for ~~\$~~ 2 for the first day. Each Subsequent day, until the City followed the judges order, the fine was squared. ie. (2, 4, 16, 256, 65,536, ...)

a) What would the fine be on day N?

$2^{(2^{(N-1)})}$

b.) $O(N) = 2^N$

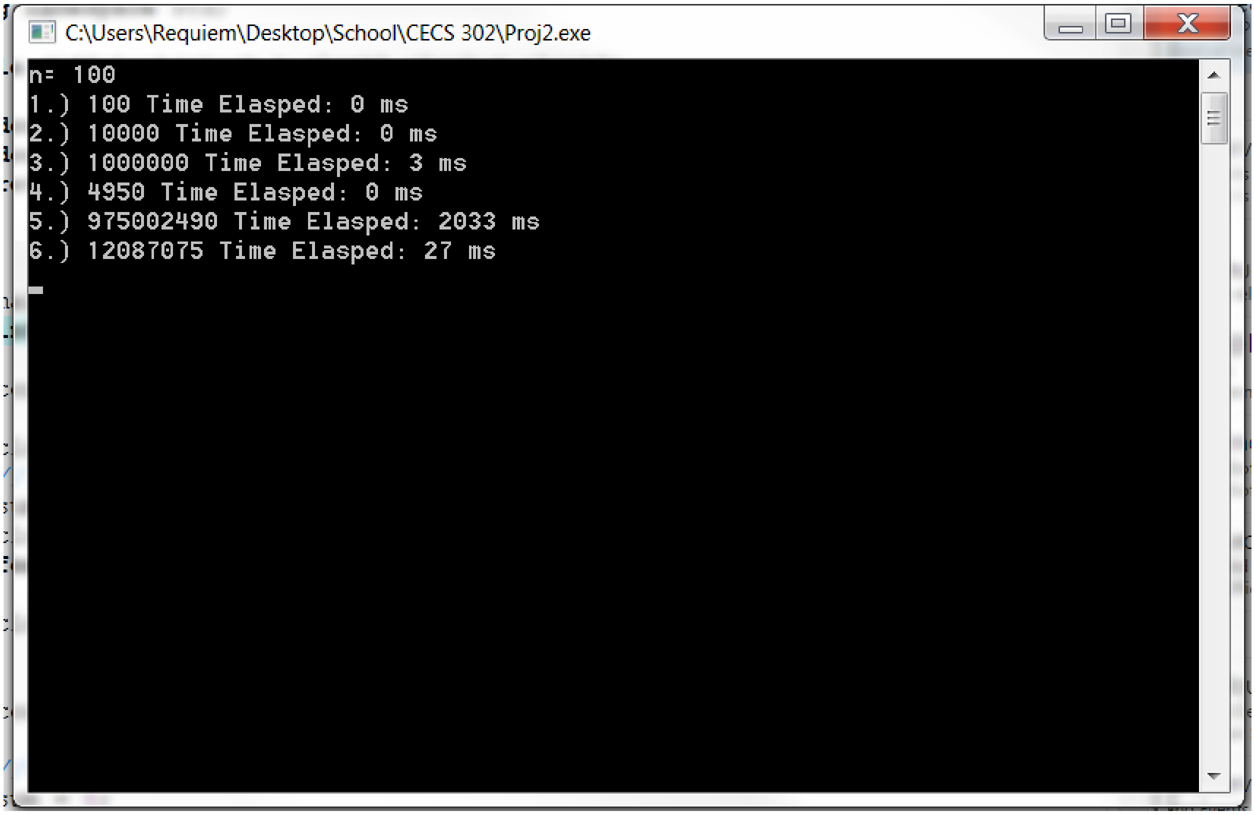
2.7.) For each of the following six program fragments:

a.) Give an analysis of the running time (Big-Oh will do.)

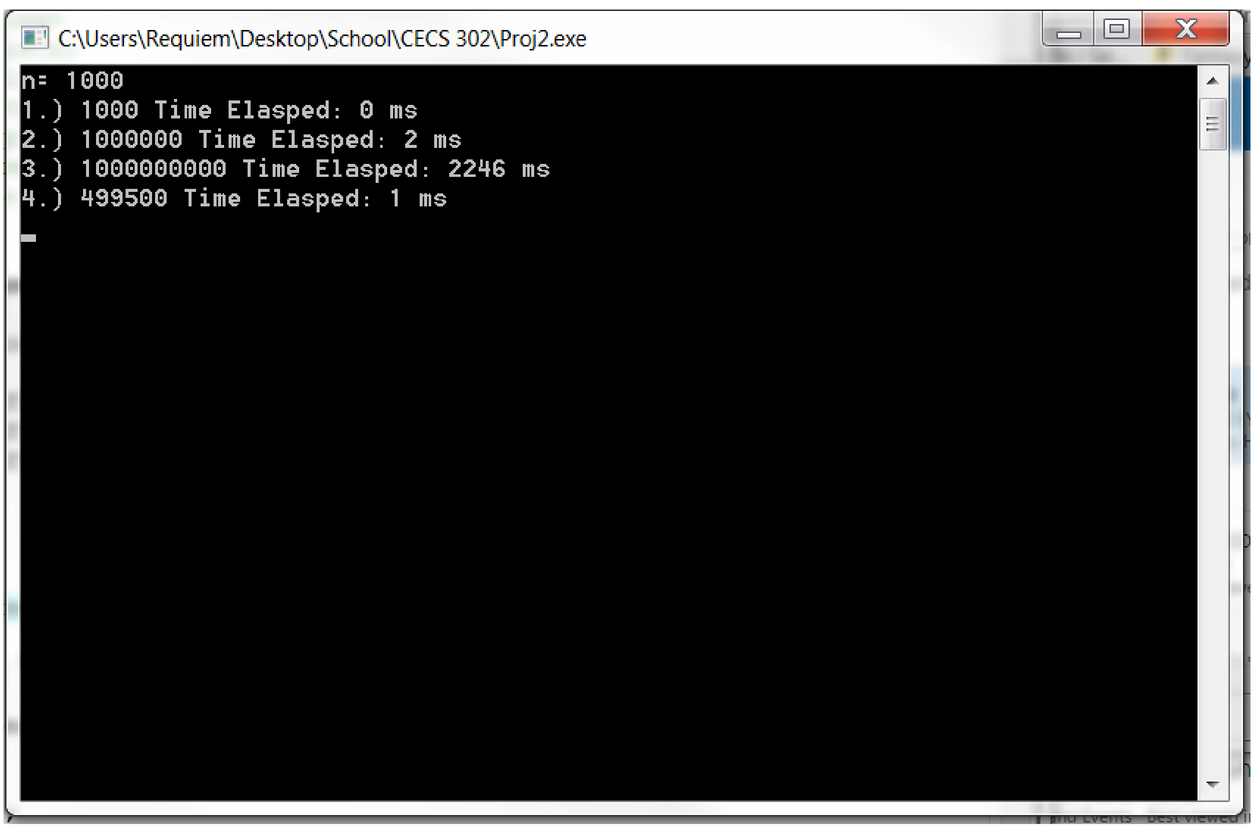
b.) Implement the Code in the language of your choice, & give running time for several values of N.

c.) Compare the analysis with the actual running time.

1.) a.) $O(N)$
2.) a.) $O(N^2)$
3.) a.) $O(N^3)$
4.) a.) $O(N^2)$
5.) a.) $N \cdot N^2 \cdot N^2 = O(N^5)$
6.) a.) $O(N^4)$



The last two take too long to process when N=1000 (I just changed the value of N in the source code)



c.) Compare the analysis with the actual run times.

	Running Times	
	N=100	N=1,000
1.) N	0 MS	0 MS
2.) N^2	0 MS	2 MS
3.) N^3	3 MS	2246 MS
4.) N^2	0 MS	1 MS
5.) N^5	2033 MS	Unknown took too long to run
6.) N^4	27 MS	unknown took too long to run

The analysis seems to be in line with the actual run times. Numbers 5 and 6 grow faster than the rest exponentially therefore they take the longest to run. Number 3 takes longer to run than 1 and 2 because it grows faster exponentially. When compared using larger numbers you can tell that Number 2 takes longer to process than number 1. Also number 4 and number 2 take about the same amount of time on both tests proving that they are equal.