Matthew Olsen
March 21, 2016

CSE 122: Homework 4A

1. Prove $T(n) = 5n^4 + 6n^2 + 2n + 4$ is O(n^4)
$$5n^4 + 6n^2 + 2n + 4 < 5n^4 + 6n^4 + 2n^4 + 4n^4 = 17n^4$$
Let c = 17 and n_0 = 1
2. Prove $T(n) = 1^2 + 2^2 + \cdots + n^2$ is O(n^4)
$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n < \frac{1}{3}n^3 + \frac{1}{2}n^3 + \frac{1}{6}n^3 = n^3$$
Let c = 1 and n_0 = 1
3. Prove T(n) = 2^{n+1} is O(2^n)
$$2^{n+1} = 2 * 2^n$$
Let c = 2 and n_0 = 0
4. If T(n) = O(nlog(n)), if n is doubled then…
$$Let\ t_1 = nlog(n)\ and\ t_2 = 2nlog(2n)$$
$$\frac{t_2}{t_1} = \frac{2nlog_2(2n)}{nlog_2(n)} = 2 * (\frac{\log_2(2)}{\log_2(n)} + 1) = 2 + \frac{2}{\log_2(n)}$$
If n is doubled, then the running time increases by $2+2/\log_2(n)$, where n is the original n.

5. The tabulated data for the seconds to execute an algorithm of size n of the given Big-O is the following:

n = 10
$O(logn) = 3/10^{-12}$
$O(n) = 10^{-11}$
$O(nlogn) = 3 * 10^{-11}$
$O(n^2) = 10^{-10}$
$O(n^3) = 10^{-9}$
$O(2^n) = 10^{-9}$
$O(n!) = 4 * 10^{-6}$

n = 10^3
$O(logn) = 10^{-11}$
$O(n) = 10^{-9}$
$O(nlogn) = 10^{-8}$
$O(n^2) = 10^{-6}$
$O(n^3) = 10^{-3}$
$O(2^n) = 10^{288}$
$O(n!) = 4 * 10^{2555}$

n = 10^6
$O(logn) = 2 * 10^{-11}$
$O(n) = 10^{-6}$

$O(nlogn) = 2 * 10^{-5}$
$O(n^2) = 1$
$O(n^3) = 10^6$
$O(2^n) = 10^{299988}$
$O(n!) = 8 * 10^{5565696}$

n = 10^9
$O(logn) = 3 * 10^{-11}$
$O(n) = 10^{-3}$
$O(nlogn) = 3 * 10^{-2}$
$O(n^2) = 10^6$
$O(n^3) = 10^{15}$
$O(2^n) = 10^{3*10^8-1}$
$O(n!) = 10^{8565705522}$

n = 10^{12}
$O(logn) = 4 * 10^{-1}$
$O(n) = 1$
$O(nlogn) = 40$
$O(n^2) = 10^{12}$
$O(n^3) = 10^{24}$
$O(2^n) = 10^{3*10^{11}-1}$
$O(n!) = 10^{11565705518091}$

n = 10^{15}
$O(logn) = 5 * 10^{-11}$
$O(n) = 10^3$
$O(nlogn) = 5 * 10^4$
$O(n^2) = 10^{18}$
$O(n^3) = 10^{33}$
$O(2^n) = 10^{3*10^{14}-12}$
$O(n!) = 10^{10^{16}-12}$

n = 10^{18}
$O(logn) = 6 * 10^{-11}$
$O(n) = 10^6$
$O(nlogn) = 6 * 10^7$
$O(n^2) = 10^{24}$
$O(n^3) = 10^{42}$
$O(2^n) = 10^{3*10^{17}-12}$
$O(n!) = 10^{10^{19}-12}$

n = 10^{21}
$O(logn) = 7 * 10^{-11}$

$O(n) = 10^9$
$O(nlogn) = 7 * 10^{10}$
$O(n^2) = 10^{30}$
$O(n^3) = 10^{51}$
$O(2^n) = 10^{3*10^{20}-12}$
$O(n!) = 10^{10^{22}-12}$

6. All of the values of n = 10 will complete under a second, and all values of n for O(logn). For n = 10^3, functions will also complete under a second except for O(2^n) and O(n!). For n = 10^6, O(n), O(nlogn), and O(n^2) will complete under second, as well as n = 10^9 for O(n), O(nlogn) and for n = 10^{12} O(n) will complete under a second. For n = 10^{12} O(nlogn) will complete under a second. For n = 10^{15} O(n) and O(nlogn) will complete in under a day. For n = 10^6 for O(n^3), for n = 10^9 for O(n^2) and n = 10^{18} for O(n) will complete in under 10 days. For n = 10^{18} for O(nlogn) and n = 10^{21} for O(n) and O(nlogn) will take several years. The functions that will take longer than the age of the universe are all O(2^n) and O(n!) for n >= 10^3, n = 10^9 for O(n^3), and all O(n^2), O(n^3) for all n >= 10^{12}.
7. The cost table for the given code is the following:

Line #	cost	# of times
1	C_1	1
2	C_2	$n+1$
3	C_3	n
4	C_4	$\frac{n(n+1)}{2} - 1$
5	C_5	$\frac{n(n+1)}{2}$
6	C_6	n

$$T(n) = \frac{1}{2}n^2(C_4 + C_5) + n\left(C_2 + C_3 + \frac{3}{2}C_4 + \frac{1}{2}C_5 + C_6\right) + C_1 + C_2$$
$$Big - O: O(n)$$

8. The cost table for the given code is the following:

Line #	Cost	# of times
1	C_1	1
2	C_2	$n+1$
3	C_3	n
4	C_4	1

$$T(n) = n(C_2 + C_3) + C_1 + C_2 + C_4$$
$$Big - O: O(n)$$

9. It would be better to use Horner's Rule, because of algorithms of O(n) of n large will take significantly less time than algorithms of O(n^2).

10. The cost table for the code provided by question 10 is the following:

Line #	Cost	Best Case # of times	Worst Case # of times
1	C_1	n	n
2	C_2	$\frac{(n+1)(n+2)}{2}-3$	$\frac{(n+1)(n+2)}{2}-3$
3	C_3	$\frac{n(n+1)}{2}-1$	$\frac{n(n+1)}{2}-1$
4	C_4	N/a	$\frac{n(n+1)}{2}-1$
5	C_5	N/a	$\frac{n(n+1)}{2}-1$
6	C_6	N/a	$\frac{n(n+1)}{2}-1$

$$Best\ Case: T(n) = \frac{1}{2}n^2(C_2 + C_3) + n\left(C_1 + \frac{3}{2}C_2 + \frac{1}{2}C_3\right) - 2C_2 - C_3$$

$$Best\ Case\ Big - O: O(n^2)$$

$$Worst\ Case: T(n) = \frac{1}{2}n^2(C_2 + C_3 + C_4 + C_5 + C_6) + n\left(C_1 + \frac{3}{2}C_2 + \frac{1}{2}C_3 + \frac{1}{2}C_4 + \frac{1}{2}C_5 + \frac{1}{2}C_6\right) - 2C_2 - C_3 - C_4 - C_5 - C_6$$

$$Worst\ Case\ Big - O: O(n^2)$$

Because both the best case and the worst case are both Big-O of O(n^2), we can say that the average case must also be **O(n^2)**.

11. The cost table for the code provided with question 11 is the following:

Line #	Cost	Best Case # of Times	Worst Case # of Times
1	C_1	n	n
2	C_2	$n-1$	$n-1$
3	C_3	$\frac{n(n+1)}{2}-1$	$\frac{n(n+1)}{2}-1$
4	C_4	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
5	C_5	N/a	$\frac{n(n-1)}{2}$
6	C_6	$n-1$	$n-1$
7	C_7	N/a	$n-1$
8	C_8	N/a	$n-1$
9	C_9	N/a	$n-1$

$$Best\ Case: T(n) = \frac{1}{2}n^2(C_3 + C_4) + n\left(C_1 + C_2 + \frac{1}{2}C_3 - \frac{1}{2}C_4 + C_6\right) - C_2 - C_3 - C_6$$

$$Big - O: O(n^2)$$

$$Worst\ Case: T(n) = \frac{1}{2}n^2(C_3 + C_4) + n\left(C_1 + C_2 + \frac{1}{2}C_3 - \frac{1}{2}C_4 - \frac{1}{2}C_5 + C_6 + C_7 + C_8 + C_9\right) - C_2 - C_3 - C_6 - C_7 - C_8 - C_9$$

$$Big - O: O(n^2)$$

Because both the best case and the worst case are Big-O of O(n^2), we can say the average case must also be **O(n^2)**.