Class Exercise 3

COSC600 - Advanced Data Structures and Algorithm Analysis

Devere Anthony Weaver

---

1. For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) milliseconds (ms).

$f(n)$	1 second	1 minute	1 hour	1 day	1 month	1 year
$log\ n$	$2^{1,000}$	$2^{60,000}$	$2^{3.6\times10^6}$	$2^{2.84\times10^7}$	$2^{2.628\times10^9}$	$2^{3.154\times10^{10}}$
$\sqrt{n}$	1000^2	$60,000^2$	$(2^{3.6\times10^6})^2$	$(2^{2.84\times10^7})^2$	$(2^{2.628\times10^9})^2$	$(2^{3.154\times10^{10}})^2$
n	1000	60,000	3.6×10^6	2.84×10^7	2.628×10^9	3.154×10^{10}
n^2	31	244	1,897	9,295	51,264	146,765
n^3	10	39	153	442	1,389	3,159
2^n	9	15	21	26	31	34
n	6	8	9	11	12	13

2. Indicate, for each pair of expressions (A, B) in the table below, whether A is O, Ω, Θ of B. Assume that $k \geq 1$, and $c > 1$ are constants.

A	B	O	Ω	Θ	o
n^k	c^n	Yes	No	No	No
$\sqrt{n}$	n	Yes	No	No	Yes
$log\ n^5$	n^2	Yes	No	No	Yes

3. Let $f(n)$ and $g(n)$ be asymptotically positive functions. Is it true or false each of the following conjectures?

a) $f(n) = O(g(n)) \Rightarrow g(n) = O(f(n))$ is FALSE.

b) $f(n) + g(n) = \Theta(max(f(n), g(n))$ is TRUE.

c) $f(n) = O(g(n)) \Rightarrow log(f(n)) = O(log(g(n)))$, where $log(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n is a TRUE statement.

d) $f(n) = O((f(n))^2)$ where $f(n)$ is an increasing function and $f(n) \geq 1$ is TRUE.

e) $f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$ is TRUE.

4. Rank the given functions in order of growth

$$1 < \sqrt{log\, n} < \sqrt{n} < (log\, n)^5 < n = 2^{log_2\, n} < n + log\, n < n\, log\, n < n^3 < 2^n < n!$$

Note that $n = 2^{log_2\, n},\ \forall n \geq 1$.

5. Give a tight asymptotic bound for the follow recurrences.

The asymptotic tight bounds for the following recurrences can all be computed using the Master Theorem.

a) $T(n) = 2T\left(\frac{n}{4}\right) + 1 \Rightarrow T(n) = \Theta(n^{log_4{}^2})$.

b) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} \Rightarrow T(n) = \Theta(\sqrt{n}\ log\, n)$.

c) $T(n) = 2T\left(\frac{n}{4}\right) + n \Rightarrow T(n) = \Theta(n)$.

d) $T(n) = 2T\left(\frac{n}{4}\right) + n^2 \Rightarrow T(n) = \Theta(n^2)$.

e) $T(n) = 2T\left(\frac{n}{2}\right) + n^4 \Rightarrow T(n) = \Theta(n^4)$.

f) $T(n) = T\left(n-2\right) + n^2 \Rightarrow T(n) = \Theta(n^2\ log\, n)$.

g) $T(n) = 2T\left(\frac{n}{2}\right) + 1 \Rightarrow T(n) = \Theta(n^{log_2{}^2})$.

---

6. Suppose $T_1 = O(f(n))$ and $T_2 = O(f(n))$.

a) Is $T_1(N) + T_2(N) = O(f(n))$ true? - This is a TRUE statement.

b) Is $T_1(N) - T_2(N) = o(f(n))$ true? - This is a FALSE statement. It does not make sense in context.

c) Is $T_1(N)/T_2(N) = O(1)$ true? - This is a FALSE statement.

---

7. In a recent court case, a judge cited a city for contempt and ordered a fine of $1 for the first day. Each subsequent day, until the city followed the judge's order, the fine was squared (that is, the fine progressed as follows: $1, $2, $4, $16, $256, $65,536, ...).

a) What would be the fine on day N?

Observing that for each day the sequence of values is squared, we can come up with a general formula to find the value of dollars given an arbitrary day N. Ultimately, after trial and error, the formula that works is given by,

$$T(n) = 2^{2^{n-2}}.$$

b) How many days would it take the fine to reach D dollars? (A Big-Oh answer will do.)

Rewriting $T(n)$, the Big-O can be computer easier. The result is $O(log\, log\, n)$.

---

8. An algorithm takes 0.1 second for input size 50. How long will it take for input size 800 if the running time is the following (assume low-order terms are negligible).

a) Linear:

$$\Rightarrow \frac{800}{50} = \frac{N}{.1} = 1.6 \text{ seconds.}$$

b) $O(\log\, n)$:

$$\Rightarrow \frac{log\, 800}{log\, 50} = \frac{N}{.1} = 0.175 \text{ seconds.}$$

c) Quadratic:

$$\Rightarrow \frac{800^2}{50^2} = \frac{N}{.1} = 25.65 \text{ seconds.}$$

d) Cubic:

$$\Rightarrow \frac{800^3}{50^3} = \frac{N}{.1} = 409.65 \text{ seconds.}$$

---

9. An algorithm takes 1 second for input size 100. How large a problem can be solved in 8 seconds if the running time is the following (assume low-order terms are negligible).

a) Linear:

$$\Rightarrow \frac{N}{100} = \frac{8}{1} = 800.$$

b) $O(\log\ n)$:

$$\Rightarrow \frac{log\, N}{log\, 100} = \frac{8}{1} = 25,600.$$

c) Quadratic:

$$\Rightarrow \frac{N^2}{100^2} = \frac{8}{1} = \lfloor 282.84 \rfloor = 282.$$

d) Cubic:

$$\Rightarrow \frac{N^3}{100^3} = \frac{8}{1} = 200.$$

10. For each of the following program fragments,
a. What is run time function, $T(n)$?
b. Give an analysis of the running time in Θ notation.

1.

a) To compute $T(n)$ for this function, consider:

- 1 assignment
- 1 assignment in the for loop
- n+1 total tests executed, including the final test to exit
- n increments of i
- n(1) for one increment statement executed n times

Thus, $T(n)$ is given by, $T(n) = 3n + 3$.

b) If $T(n) = 3n + 3$ then $T(n) = \Theta(n)$.

2.

a) To compute $T(n)$ for this function, consider:

- 1 assignment
- Outer loop: 1 assignment + (n+1) comparisons + n increments = 2n + 2
- Inner loop: 1 assignment + (n+1) comparisons + n increments = 2n + 2
- n(1) constant statements inside the inner loop

Thus $T(n)$ is given by, $T(n) = 2n^2 + 4n + 3$.

b) To find the tight bound on the code snippet, evaluate

$$\sum_{i=0}^{n-1}\sum_{k=0}^{i-1} 1 \Rightarrow \sum_{i=1}^{n}\sum_{k=1}^{i} 1 = \frac{n^2+n}{2}.$$

Thus $T(n) = \Theta(n^2)$.

3.

a) To compute $T(n)$ for this function, consider the same criteria as the first two snippets; however, the first loop executes $6n$ times. Thus, $T(n) = 12n^2 + 24n + 3$.

b) To find the tight bound on the code snippet, evaluate

$$\sum_{i=0}^{6n-1}\sum_{k=0}^{i-1} 1 \Rightarrow \sum_{i=1}^{6n}\sum_{k=1}^{i} 1 = \frac{36n^2+6}{2}.$$

Thus $T(n) = \Theta(n^2)$.

4.

a) To compute $T(n)$ for this function, observe this function extends the previous two by adding an additional for loop. Thus $T(n) = n^3 + 6n + 7$.

b) To find the tight bound on the code snippet, evaluate

$$\sum_{i=0}^{n-1}\sum_{j=0}^{i-1}\sum_{k=0}^{j-1} \Rightarrow \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} = \frac{2n^3 + 6n^2 + 4n}{12}.$$

Thus $T(n) = \Theta(n^3)$.

5.

a) To compute $T(n)$ for this function, we can add the assignment plus the two runtimes for the separate for loops to obtain $T(n) + 3n^2 + 3n + 4$.

b) To find the tight bound on the code snippet, evaluate

$$\sum_{i=0}^{n-1} + \sum_{j=0}^{n^2-1} \Rightarrow \sum_{i=1}^{n} + \sum_{j=1}^{n^2} i = n + n^2.$$

Thus $T(n) = \Theta(n^2)$.