CS 502 - Analysis Worksheet

Professor Mark W. Boady

1 Instructions

This worksheet is worth 35 points.

Answer the questions on this sheet or on in a seperate work document. If you choosed to use a blank document make sure to clearly number each question.

You do not need to copy the entire question contents if you use a seperate document. Just the numbers and answers.

2 Analysis Worksheet

Question 1 : 5 points

There are many notations in the Analysis of Algorithms. They are designed to convey information about how the algorithms work.

In a general sense, each of these notations is used to convey an approximation of an algorithm runtime. We will see the formal definitions for each of these later in this lab.

The first three notations we will start with are given below. More will be added later.

- $f(n) = o(g(n))$ means *The function $f(n)$ is always smaller than the function $g(n)$.*
- $f(n) = \Theta(g(n))$ means *The function $f(n)$ is approximated by $g(n)$.*
- $f(n) = \omega(g(n))$ means *The function $f(n)$ is always larger than the function $g(n)$.*

Translate each of the following sentences into notation of the form $f(n) = ?(g(n))$.

Example:

My sorting algorithm, $s(n)$, always takes more than n^2 comparisons to sort an array of size n.

translates to

$$s(n) = \omega(n^2)$$

(a) (1 point) My shuffle function, $f(n)$, always finishes in less than $2 * n^2$ swaps.

(A.) $f(n) = o(2n^2)$
B. $f(n) = \Theta(2n^2)$
C. $f(n) = \omega(2n^2)$

(b) (1 point) My function, $f(n)$, does about $3n^2 + 5$ multiplications.

A. $f(n) = o(3n^2 + 5)$
(B.) $f(n) = \Theta(3n^2 + 5)$
C. $f(n) = \omega(3n^2 + 5)$

(c) (1 point) My function, $f(n)$, always needs more than $2n$ subtractions before finishing.

A. $f(n) = o(2n)$
B. $f(n) = \Theta(2n)$
(C.) $f(n) = \omega(2n)$

(d) (1 point) My search function, $f(n)$, has never taken more than n^3 seconds to complete.

(A.) $f(n) = o(n^3)$
B. $f(n) = \Theta(n^3)$
C. $f(n) = \omega(n^3)$

(e) (1 point) My function, $f(n)$, takes more than $4n$ operations.

A. $f(n) = o(4n)$
B. $f(n) = \Theta(4n)$
(C.) $f(n) = \omega(4n)$

Question 2 : 1 points

We need a system to compare algorithms and determine which is **faster**.

Most importantly, we care about how algorithms scale. If we double the size of the input, will the runtime double or quadruple. If we need to chose between two algorithms, the one that doubles will perform better then the one that quadruples.

To talk about how algorithms scale with large inputs, we need to use limits. Some important rules of limits are given below. The limit determines what value the function approaches as the size of the input increases towards ∞.

$$\lim_{n\to\infty}(f(n)+g(n)) = \lim_{n\to\infty}(f(n)) + \lim_{n\to\infty}(g(n)) \quad \text{Sum Rule} \quad (1)$$

$$\lim_{n\to\infty}(f(n)\cdot g(n)) = \lim_{n\to\infty}(f(n)) \cdot \lim_{n\to\infty}(g(n)) \quad \text{Product Rule} \quad (2)$$

$$\lim_{n\to\infty}(C) = C \quad \text{Where C is a constant} \quad (3)$$

$$\lim_{n\to\infty}(Cf(n)) = C\lim_{n\to\infty}(f(n)) \quad \text{Constant Mult} \quad (4)$$

$$\lim_{n\to\infty}\left(n^C\right) = \infty \quad \text{Approaches Infinity} \quad (5)$$

$$\lim_{n\to\infty}\left(\frac{1}{n^C}\right) = 0 \quad \text{Approaches 0} \quad (6)$$

Imagine we have one algorithm that takes an array of size n and computes $f(n) = 2n^2+4n+2$ operations. Someone proposes an algorithm that takes $g(n) = 10n^3$ operations.

Which one is faster? We take the limit of the ratio between the expressions.

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \lim_{n\to\infty}\left(\frac{2n^2+4n+2}{10n^3}\right) \quad (7)$$

$$= \lim_{n\to\infty}\left(\frac{2n^2}{10n^3} + \frac{4n}{10n^3} + \frac{2}{10n^3}\right) \quad (8)$$

$$= \lim_{n\to\infty}\left(\frac{1}{5n} + \frac{2}{5n^2} + \frac{1}{5n^3}\right) \quad (9)$$

$$= \lim_{n\to\infty}\left(\frac{1}{5n}\right) + \lim_{n\to\infty}\left(\frac{2}{5n^2}\right) + \lim_{n\to\infty}\left(\frac{1}{5n^3}\right) \quad (10)$$

$$= 0+0+0 \quad (11)$$

$$= 0 \quad (12)$$

(a) (1 point) Which is true?

(A.) $f(n) = o(g(n))$

B. $g(n) = o(f(n))$

Question 3 : 5 points

Compute the limits of the following function ratios. Your answers should always fall into one of three categories

- $\lim_{n\to\infty}(\cdots) = \infty$
- $\lim_{n\to\infty}(\cdots) =$ Some Constant Number (Determine specific constant if possible)
- $\lim_{n\to\infty}(\cdots) = 0$

$$f(n) = 1 \tag{13}$$
$$g(n) = 3n + 2 \tag{14}$$
$$h(n) = 5n \tag{15}$$
$$j(n) = 2n^2 \tag{16}$$
$$k(n) = 25n^3 \tag{17}$$

(a) (1 point) What is $\lim_{n\to\infty}\left(\frac{g(n)}{f(n)}\right)$?

A. 0
B. Constant
(C.) ∞

(b) (1 point) What is $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right)$?

(A.) 0
B. Constant
C. ∞

(c) (1 point) What is $\lim_{n\to\infty}\left(\frac{g(n)}{h(n)}\right)$?

A. 0
(B.) Constant 3/5
C. ∞

(d) (1 point) What is $\lim_{n\to\infty}\left(\frac{j(n)}{g(n)}\right)$?

A. 0
B. Constant
(C.) ∞

(e) (1 point) What is $\lim_{n\to\infty}\left(\frac{h(n)}{k(n)}\right)$?

(A.) 0
B. Constant
C. ∞

Question 4 : 6 points

We classify functions using bounds. A function is an **upper bound** if it is always larger. For example, $x^3 \geq x^2$. In this case x^3 is an upper bound for x^2.

- If $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \infty$ then $f(n)$ is an **upper bound** on $g(n)$
- If $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) =$ Some Constant Number then $f(n)$ is a **tight bound** on $g(n)$
- If $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$ then $f(n)$ is a **lower bound** on $g(n)$

Complete the following sentences. Use the same functions as the previous page.

$$f(n) = 1 \tag{18}$$
$$g(n) = 3n + 2 \tag{19}$$
$$h(n) = 5n \tag{20}$$
$$j(n) = 2n^2 \tag{21}$$
$$k(n) = 25n^3 \tag{22}$$

(a) (1 point) The function $g(n)$ is a/an [?] bound on $f(n)$.
- (A.) upper
- B. tight
- C. lower

(b) (1 point) The function $f(n)$ is a/an [?] bound on $g(n)$.
- A. upper
- B. tight
- (C.) lower

(c) (1 point) The function $g(n)$ is a/an [?] bound on $h(n)$.
- A. upper
- (B.) tight
- C. lower

(d) (1 point) The function $j(n)$ is a/an [?] bound on $g(n)$.
- (A.) upper
- B. tight
- C. lower

(e) (1 point) The function $h(n)$ is a/an [?] bound on $k(n)$.
- A. upper
- B. tight
- (C.) lower

(f) (1 point) The function $k(n)$ is a/an [?] bound on $f(n)$.
- (A.) upper
- B. tight
- C. lower

Question 5 : 6 points

The idea of bounding limits is a good way to compare algorithms. We need a formal methodology to apply it consistently.

We define three **asymptotic notations** used to classify algorithm runtime.

- **Little O**: $T(n) = o(f(n))$ If for all positive constant $c > 0$ there exists an $n_0 \geq 0$ such that for all $n \geq n_0$ we have $0 \leq T(n) \leq cf(n)$.
- **Theta**: $T(n) = \Theta(f(n))$ if there exists positive constants $c_1 > 0$, $c_2 > 0$, $n_0 \geq 0$ such that for all $n \geq n_0$ we have $c_1 f(n) \leq T(n) \leq c_2 f(n)$
- **Little Omega**: $T(n) = \omega(f(n))$ if for all positive constant $c > 0$ there exists an $n_0 \geq 0$ such that for all $n \geq n_0$ we have $0 \leq cf(n) \leq T(n)$.

Match each of these definitions to a limit condition.

(a) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) = \infty$

A. $f(n) = o(g(n))$
B. $f(n) = \Theta(g(n))$
(C.) $f(n) = \omega(g(n))$

(b) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) =$ Some Constant Number

A. $f(n) = o(g(n))$
(B.) $f(n) = \Theta(g(n))$
C. $f(n) = \omega(g(n))$

(c) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) = 0$

(A.) $f(n) = o(g(n))$
B. $f(n) = \Theta(g(n))$
C. $f(n) = \omega(g(n))$

(d) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) = \infty$

(A.) $g(n) = o(f(n))$
B. $g(n) = \Theta(f(n))$
C. $g(n) = \omega(f(n))$

(e) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) =$ Some Constant Number

A. $g(n) = o(f(n))$
(B.) $g(n) = \Theta(f(n))$
C. $g(n) = \omega(f(n))$

(f) (1 point) $\lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right) = 0$

A. $g(n) = o(f(n))$
B. $g(n) = \Theta(f(n))$
(C.) $g(n) = \omega(f(n))$

Question 6 : 6 points

There are two other **asymptotic notations**. We can think of the notations using the following relationship to integers.

Asymptotic Notation	Similar Integer Operation
$f(n) = o(g(n))$	$F < G$
$f(n) = O(g(n))$	$F \leq G$
$f(n) = \Theta(g(n))$	$F \approx G$
$f(n) = \Omega(g(n))$	$F \geq G$
$f(n) = \omega(g(n))$	$F > G$

- **Big Oh**: $T(n) = O(f(n))$ if there exists constants $c > 0$ and $n_0 \geq 0$ such that $0 \leq T(n) \leq cf(n)$ for all $n \geq n_0$
- **Big Omega**: $T(n) = \Omega(f(n))$ if there exists constants $c > 0$ and $n_0 \geq 0$ such that $0 \leq cf(n) \leq T(n)$ for all $n \geq n_0$

We can define notations in terms of limits here as well.

- Loose Upper Bound
  - $\lim_{n=\infty} \frac{f(n)}{T(n)} = \infty$
  - $T(n)$ is $O(f(n))$ and $o(f(n))$
- Tight Bound
  - $\lim_{n=\infty} \frac{f(n)}{T(n)} = k$ for some constant k
  - $T(n)$ is $O(f(n))$ and $\Omega(f(n))$ and $\Theta(f(n))$
- Loose Lower Bound
  - $\lim_{n=\infty} \frac{f(n)}{T(n)} = 0$
  - $T(n)$ is $\Omega(f(n))$ and $\omega(f(n))$

(a) (1 point) $3n^2 - 2n = O(n^2)$

(A.) True
B. False

(b) (1 point) $3n^2 - 2n = O(n^4)$

(A.) True
B. False

(c) (1 point) $3n^2 - 2n = O(n)$

A. True
(B.) False

(d) (1 point) $3n^2 - 2n = \Omega(n)$

(A.) True
B. False

(e) (1 point) $3n^2 - 2n = \Omega(n^2)$

(A.) True
B. False

(f) (1 point) $3n^2 - 2n = \Omega(n^4)$

A. True
(B.) False

Question 7 : 6 points

Determine the most accurate $\Theta(\cdots)$ for each function. Assume every array has size n.

(a) (2 points) The `max` method is $\Theta(\quad)$.

```
1 int max(int* A, int n){
2         if(n < 2)
3                 return -1;
4         else if( A[0] > A[1])
5                 return A[0];
6         else
7                 return A[1];
8 }
```

(A.) $\Theta(1)$
B. $\Theta(\log_2(n))$
C. $\Theta(n)$
D. $\Theta(n^2)$

(b) (2 points) The `maxElement` method is $\Theta(\quad)$.

```
1 int maxElement(int* A, int n){
2         int largest = A[0];
3         for(int i=0; i < n; i++){
4                 if(A[i] > largest)
5                         largest=A[i];
6         }
7         return largest;
8 }
```

A. $\Theta(1)$
B. $\Theta(\log_2(n))$
(C.) $\Theta(n)$
D. $\Theta(n^2)$

(c) (2 points) The `maxSubseqSum` method is $\Theta(\quad)$.

```
1  int maxSubseqSum(int* A, int n){
2          int result = A[0];
3          for(int i=0; i < n; i++){
4                  int sum = 0;
5                  for(int j=i; j < n; j++){
6                          sum+=A[j];
7                          if(sum > result)
8                                  result=sum;
9                  }
10         }
11         return result;
12 }
```

A. $\Theta(1)$
B. $\Theta(\log_2(n))$
C. $\Theta(n)$
(D.) $\Theta(n^2)$