Hannan, Michael
CSCI 2114: Tashfeen's Data Structures
October 6, 2025

Homework 2

Let's remind ourselves of the asymptotic order notation[1]. Let $f(x)$ and $g(x)$ be functions of a positive x.

$$f(x) = \mathcal{O}(g(x))$$

when there is positive constant c such that,

$$f(x) \leq cg(x)$$

for all $x \geq n$. Function f is then stated as *big oh* of g. Similarly, f is *big omega* of g, i. e.,

$$f(x) = \Omega(g(x))$$

when there exist positive constants c, n such that for all $x \geq n$ we have,

$$f(x) \geq cg(x)$$

Lastly, if f is both big-$\mathcal{O}$ and big-Ω of g,

$$\Omega(g(x)) = f(x) = \mathcal{O}(g(x))$$

Then such a tight bound is stated as f is big-Θ of g,

$$f(x) = \Theta(g(x))$$

Question 1. Consider the following Java subroutine,

```
1  public static void conditionalWork(int n) {
2      for (int i = 0; i < n; i++)
3          if (Math.random() < 0.5)
4              taskA()
5          else
6              taskB()
7  }
```

1) If on a certain machine the function `taskA()` takes 2 seconds on each call in the loop and the function `taskB()` takes 4 seconds then for $n = 10$, what is the total number of *expected* seconds taken by the subroutine `public static void conditionalWork(int n)`?

   The total number of expected seconds is 30 seconds

$$a(x) = 2(\frac{x}{2}) \quad b(x) = 4(\frac{x}{2})$$

$$result(10) = 2 * 5 \ + 4 * 5 = 30 \ seconds$$

2) The subroutine `conditionalWork(int n)` is ran on a much faster machine reducing the time taken by `taskA()` on each call down to $\frac{1}{25}^{\text{th}}$ of a second and `taskB()` now takes $\frac{1}{5}^{\text{th}}$ of a second. For $n = 10$, what is the new total number of seconds *expected* by the subroutine `conditionalWork(int n)`?

$$A = \frac{1}{25} \quad B = \frac{1}{5}$$

$$result(10) = \frac{1}{25} * 5 \ + \frac{1}{5} * 5$$

$$0.2 \ + 1 = 1.2 \ seconds$$

[1]Sometimes also referred to as the Bachmann-Landau notation.

The new total of expected seconds is 1.2 seconds

3) Assume that for any arbitrary n, `taskA()` takes $\log(n)$ steps while `taskB()` takes $2n^2$ steps. Let $T(n)$ be the total number of steps *expected* by `conditionalWork(int n)`, what is $T(n)$?
   $T(n) = n\log(n)/2 + n^3$
4) What is the *best expected asymptotic* runtime complexity written as $T(n) = \Omega(g(n))$?
   $\Omega(nlogn/2)$
5) What is the *worst expected asymptotic* runtime complexity written as $T(n) = \mathcal{O}(g(n))$?
   $\mathcal{O}(n^3)$
6) If `taskB()` took $4\log(n)$ steps, what is the *tight expected asymptotic* runtime complexity written as $T(n) = \Theta(g(n))$?
   $\Theta = n^2logn$ $n/2$ $n^2logn = nlogn$

Question 2. In the code snippet bellow, we see two ways of getting the tenth place digit of a Java `int` n. Give the runtime complexity of each method using the *big oh* notation. Justify your answer.

```
1 System.out.println(n % 10);
2 System.out.println(String.valueOf(n).charAt(String.valueOf(n).length() - 1));
```

The first line is equal to O(1) or constant time due to no iteration.

The second line is o(n)5 or linear time because it checks for the .length which iterates through the loop once.

Question 3. Consider the following functions,

$$a(x) = 2(x^3 + 1)(x^2 - 1) + 2$$
$$b(x) = 2x^2$$
$$\alpha(x) = x^2$$
$$\beta(x) = \pi x^2$$

1) Observe that $b(x) = \mathcal{O}(a(x))$ because $b(x) \leq a(x)$ past a certain $x = n$. What is the value of n in this case?
   Set Both equations equal to each-other and solve

$$a(x) = b(x)$$
$$2(x^3 + 1)(x^2 - 1) + 2 = 2x^2$$
$$(x^3 + 1)(x^2 - 1) = x^2 - 1$$
$$(x^3 + 1) = 0$$
$$x^3 = 1$$
$$x = 1$$

   x = 1 would be the intersection

   Also If you were to graph both equations and find the intersects they would intersect at x = 1.

2) Observe that $\beta(x) = \mathcal{O}(\alpha(x))$ because there exists a c such that $\beta(x) \leq c\alpha(x)$ for all $x > 0$. What is the value of c in this case?
   The difference between $\beta(x)$ and $c\alpha(x)$ is the constant π so therefore c is π

$$C = \pi$$

Question 4. Given in listing **??** is a Java subroutine that sorts an array of non-negative Java integers in ascending order.

Assume that the length of the input parameter `int[] toSort` is $n+1$, the max(`toSort`) $\leq n$ and that `toSort` has unique elements. Let $T(n)$ be the worst and the average case complexity of the algorithm's runtime and $S($`toSort`$)$ be the worst case complexity of the memory-space.

1) What is $T(n)$?

$$T(n) = 3n$$

2) What are $\mathcal{O}(T(n)), \mathcal{O}(S($`toSort`$))$?
   The $\mathcal{O}$ of T(n) would be just $\mathcal{O}$(n) as constants are insignificant. This is linear time.
   Similarly there is an array made with elements the size of the input so

$$O(S(toSort)) == (O(S(N)))$$

   This would be linear time.

3) Look up and state the *big-*$\mathcal{O}$ of the average case complexity of Java's built-in `Arrays.sort(int[])`. Is this better than $\mathcal{O}(T(n))$ from the previous step of this question?
   The Big $\mathcal{O}$ of Java's implementation using Arrays.sort(int[]) is O(n log(n)).

$$O(nlog(n)) > O(n)$$

   Therefore, Java's implementation would be 'worse'.

Question 5. Given bellow is the Java implementation of the sieve of Eratosthenes. This particular implementation marks all the composite numbers as `true` since Java allocates `false` to all the elements of a newly created boolean arrays.

```
1  public static void eratosthenes(boolean[] toSieve) {
2      toSieve[0] = true;
3      toSieve[1] = true;
4      for (int i = 2; i < Math.sqrt(toSieve.length); i++)
5          if (!toSieve[i])
6              for (int j = i*i; j < toSieve.length; j += i)
7                  toSieve[j] = true;
8  }
```

Give an upper-bound on the runtime complexity of this particular implementation of the sieve of Eratosthenes. The better upper-bound you give, the more credit you get. Justify your answer.

The solution above loops through the the data set twice. Although one of the solutions loops through the equivalent of SQRT times which does make a moderate difference in time complexity.

$$O(n) = \sqrt{n} * n$$
$$N * N^{0.5}$$
$$O(n) = N^{1.5}$$

This would be considered Quasi-linear time or O(N log n)

Submission Instructions

Submit a PDF file with your answers.