ECS 32B – Introduction to Data Structures
Homework 01

Name: Camden Possinger
Student ID: 916955731

Due: October 9, 2020, 5:00pm PST

Make sure to clearly identify your name and student number. Submit only one pdf file via Gradescope. Do not submit a separate file for each problem. If you handwrite your solutions, make sure they are clear and readable.

Problem 1 (10 points each)

For each of the following functions, do the following:

- Calculate $T(n)$, making sure not to discard constants or low-order terms yet (you may or may not count the counting variable in a for loop)
- Given what you think is the Big-O performance for the function
- Prove that your proposed Big-O for the function is correct (make sure to show your work; do not just provide values for c and n_0)

(1)

```
def fun1(n):
    x = 0   +1
    for i in range(n/2):    +(n/2)
        x = 2 * 5 - i  +1
    return x  +1
```

- $T(n)$ is $\frac{n}{2} + 2$
- $O(n)$ is n
- Prove $T(n)$ is $O(n)$ showing that $T(n) \leq c \cdot f(n)$ where $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$

$$\frac{n}{2} + 2 \leq c \cdot n$$

when $n_0 = 2$ and $c = 2$

$3 \leq 4$ is true

$c > 0$ and $n_0 \geq 0$ where $n \geq n_0$

$T(n)$ is $O(n)$ ■

(2)

```
def fun2(n):
    x = 0 +1
    for i in range(1, n): +(n−1)
        for j in range(1, n):   +(n−1)
            for k in range(1, n):   +(n−1)
                x = x + 1   +1
    return x     +1
```

- $T(n)$ is $(n-1)^3 + 2$ which simplifies to $n^3 - 3n^2 + 3n + 1$
- $O(n)$ is n^3
- Prove $T(n)$ is $O(n)$ showing that $T(n) \leq c \cdot f(n)$ where $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$

$$n^3 - 3n^2 + 3n + 1 \leq c \cdot n^3$$

$$\text{when } n_0 = 2 \text{ and } c = 1$$

$$3 \leq 8 \text{ is true}$$

$$\text{since } c > 0 \text{ and } n_0 \geq 0 \text{ where } n \geq n_0$$

$$T(n) \text{ is } O(n) \blacksquare$$

---

(3)

```
def fun3(n):
    x = 0 +1
    i = n   +1
    while i >= 0: +(n+1)
        x = x − i +1
        i = i − 1 +1
    return x +1
```

- $T(n)$ is $2(n+1) + 3$ which simplifies to $2n + 5$
- $O(n)$ is n
- Prove $T(n)$ is $O(n)$ showing that $T(n) \leq c \cdot f(n)$ where $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$

$$2n + 5 \leq c \cdot n$$

$$\text{when } n_0 = 3 \text{ and } c = 4$$

$$11 \leq 12 \text{ is true}$$

$$\text{since } c > 0 \text{ and } n_0 \geq 0 \text{ where } n \geq n_0$$

$$T(n) \text{ is } O(n) \blacksquare$$

---

(4)

```
def fun4(n):
    x = n +1
    for i in range(n): +n
        x = x + 1 +1
    for i in range(n * n): +n^2
        x = x + 2 +1
    return x +1
```

- $T(n)$ is n^2+n+2
- $O(n)$ is n^2
- Prove $T(n)$ is $O(n)$ showing that $T(n) \leq c \cdot f(n)$ where $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$

$$n^2+n+2 \leq c \cdot n^2$$
$$\text{when } n_0 = 2 \text{ and } c = 3$$
$$8 \leq 12 \text{ is true}$$
$$\text{since } c > 0 \text{ and } n_0 \geq 0 \text{ where } n \geq n_0$$
$$T(n) \text{ is } O(n) \blacksquare$$

---

(5)

```
# lst is a list of integers
def fun5(lst):
    x = 0   +1
    for y in lst: +n
        x = x + y  +1
    return x  +1
```

- $T(n)$ is $n+2$
- $O(n)$ is n
- Prove $T(n)$ is $O(n)$ showing that $T(n) \leq c \cdot f(n)$ where $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0$

$$n+2 \leq c \cdot n$$
$$\text{when } n_0 = 2 \text{ and } c = 3$$
$$4 \leq 6 \text{ is true}$$
$$\text{since } c > 0 \text{ and } n_0 \geq 0 \text{ where } n \geq n_0$$
$$T(n) \text{ is } O(n) \blacksquare$$

Problem 2 (10 points each)

Suppose an algorithm solves a problem of input size n in at most the number of steps listed for each $T(n)$ given below. Calculate the Big-Θ (not just Big-O) for each $T(n)$. Show your work, including values for c and n_0.

(1) $T(n)$ is 5
$O(n)$ is 1
for $5 \leq c \cdot 1$
$n_0 = 0$ and $c = 5$ gives
$5 \leq 5$ which is true
so $T(n)$ is $O(n)$

$$T(n) \text{ is } 5$$
$$\Omega(n) \text{ is } 1$$
$$\text{for } 5 \geq c \cdot 1$$
$$n_0 = 0 \text{ and } c = 5 \text{ gives}$$
$$5 \geq 5 \text{ which is true}$$
$$\text{so } T(n) \text{ is } \Omega(n)$$

Since $T(5)$ is $O(1)$ and $T(5)$ is $\Omega(1)$
$T(5)$ is $\Theta(1)$

(2) $T(n)$ is $2n^2+1$
$O(n)$ is n^2
for $2n^2+1 \leq c \cdot n^2$
$n_0 = 3$ and $c = 3$ gives
$19 \leq 27$ which is true
so $T(n)$ is $O(n)$

$T(n)$ is $2n^2+1$
$\Omega(n)$ is n^2
for $2n^2+1 \geq c \cdot n^2$
$n_0 = 1$ and $c = 2$ gives
$3 \geq 2$ which is true
so $T(n)$ is $\Omega(n)$

Since $T(2n^2+1)$ is $O(n^2)$ and $T(2n^2+1)$ is $\Omega(n^2)$
$T(2n^2+1)$ is $\Theta(n^2)$

---

(3) $T(n)$ is $3n^4+2n^3+2n$
$O(n)$ is n^4
for $3n^4+2n^3+2n \leq c \cdot n^4$
$n_0 = 1$ and $c = 8$ gives
$7 \leq 8$ which is true
so $T(n)$ is $O(n)$

$T(n)$ is $3n^4+2n^3+2n$
$\Omega(n)$ is n^4
for $3n^4+2n^3+2n \geq c \cdot n^4$
$n_0 = 1$ and $c = 2$ gives
$7 \geq 2$ which is true
so $T(n)$ is $\Omega(n)$

Since $T(3n^4+2n^3+2n)$ is $O(n^4)$ and $T(3n^4+2n^3+2n)$ is $\Omega(n^4)$
$T(3n^4+2n^3+2n)$ is $\Theta(n^4)$

---

(4) $T(n)$ is $log(5 \times 2^n)$
which simplifies to
$log5 + log2^n$
$n + log5$
where the base of the log is 2
$O(n)$ is n
for $n + log5 \leq c \cdot n$
$n_0 = 2$ and $c = 3$ gives
$2 + log5 \approx 4.3219 \leq 6$ which is true
so $T(n)$ is $O(n)$

$T(n)$ is $log(5 \times 2^n)$
which simplifies to
$log5 + log2^n$
$n + log5$
where the base of the log is 2
$\Omega(n)$ is n
for $n + log5 \geq c \cdot n$
$n_0 = 1$ and $c = 2$ gives
$1 + log5 \approx 3.3219 \geq 2$
which is true
so $T(n)$ is $\Omega(n)$

Since $T(log(5 \times 2^n))$ is $O(n)$ and $T(log(5 \times 2^n))$ is $\Omega(n)$
$T(log(5 \times 2^n))$ is $\Theta(n)$

(5) $T(n)$ is $4n^2logn$
$O(n)$ is n^2logn
for $4n^2logn \leq c \cdot n^2logn$
$n_0 = 2$ and $c = 5$ gives
$16 \leq 20$ which is true
so $T(n)$ is $O(n)$

$T(n)$ is $4n^2logn$
$O(n)$ is n^2logn
for $4n^2logn \geq c \cdot n^2logn$
$n_0 = 2$ and $c = 3$ gives
$16 \geq 12$ which is true
so $T(n)$ is $\Omega(n)$

Since $T(4n^2logn)$ is $O(n^2logn)$ and $T(4n^2logn)$ is $\Omega(n^2logn)$
$T(4n^2logn)$ is $\Theta(n^2logn)$