CSE 373 Fall 2015, Homework 2
Due Friday, October 16th in homework Dropbox

Please show work where applicable
Name: Changming Feng
uwid (not your student number): cmf927

1) For each of the following, show that $f \in O(g)$. That is, you will need to find values for c and n_0 such that the definition of big-O holds true as we did with the examples in lecture.

a) $f(n) = 12n$ $\qquad$ $g(n) = \frac{n}{5}$

choose $c=60$, $n_0=1$, $cg(n)=12n$,
for all $n \geq n_0$, $f(n) \leq cg(n)$, that is $f \in O(g)$

b) $f(n) = 6n^2 + 1000$ $\qquad$ $g(n) = n^4$

choose $c=1$, $n_0=10$,
for all $n \geq 10$, $f(n) \leq cg(n)$

c) $f(n) = 6\log(n)$ $\qquad$ $g(n) = .5n$

choose $c=12$, $n_0=1$, $cg(n)=6n$
for all $n \geq 1$, $f(n) \leq cg(n)$

2) For each of the following program fragments, determine the asymptotic runtime in terms of n

a)

```
public void mysteryOne(int n) {    n+1
    int x = 0;
    for (int i = n; i >= 0; i—) {
        if ((i % 5) == 0) {                      log(n)
            break;
        } else {
            for (int j = 1; j < n; j *= 2) {
                x++;
            }
        }
    }
}
```

$T(n) = \theta(n\log(n))$

b)

```
public void mysteryTwo(int n) {    n+1
    int x = 0;                                          n²-1/3 +1
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < ((n * n - 1)/ 3);  j++) {
            x += j;
        }
    }
}
```

$T(n) = \theta(n^3)$

c)

```
public void mysteryThree(int n) {
    for (int i = 0; i < n; i++) {   (n+1)
        methodTwo(i);
    }
}

private void methodTwo(int x) {    n
    if (x > 0) {
        methodTwo(x -1);
    }
}
```

$T(n) = \theta(n^2)$

3) For each of the following, determine if $f \in O(g)$, $f \in \Omega(g)$, $f \in \Theta(g)$, several of these, or none of these.

a) $f(n) = \log n$ $\quad$ $g(n) = \log \log n$

$f \in \Omega(g)$

b) $f(n) = 2^{2n}$ $\quad$ $g(n) = 2^n$

$f \in \Omega(g)$.

c) $f(n) = 25n^3$ $\quad$ $g(n) = n^3 + 25n$

$f \in O(g)$, $f \in \Omega(g)$, $f \in \Theta(g)$

4) Psuedocode and recurrence relations

a) Write pseudocode for a function that calculates the largest difference between any two numbers in an array of positive integers with a runtime in $\Theta(n^2)$.

For example, the largest difference between any two numbers in the following array would be 19.

```
a = [4, 6, 3, 9, 2, 1, 20]
```

```
int max=0
  for (each index i in array) {
     for (each index j > i) {
         update max if difference between element i and element j is larger than max
     }
  }
return max
```

b) Can this function be written with a runtime in $\Theta(n)$? If yes, write pseudocode below. If no, why? What would have to be different about the input in order to do so?

Yes.

```
int max=element 0;
int min= element 0;
for (each index i in array){
   update max if element i is larger than max;
   update min if element i is smaller than min;
   }
}
return max-min;
```

Don't need any difference about the input order.

c) Can this function be written with a runtime in $\Theta(1)$?. If yes, write pseudocode below. If no, why? What would have to be different about the input in order to do so?

Yes.

```
max= (element at the end of array)-(element at the beginning of array);
return max;
```

The input of array need to be sorted, with the largest element at the end and the smallest one at the beginning.

5) Recurrence Relations

a) Find the tightest Big-Oh bound for the following recurrence relation $T(n) = n + T(n/2)$. Justify your answer.

$T(n) = n + T(n/2)$

$= n + n/2 + T(n/4)$

$= \ldots$

$= n(1 + 1/2 + 1/4 + \cdots + 1/2^{k-1}) + T(n/2^k)$

$= n(2 - \frac{1}{2^{k-1}}) + T(n/2^k)$

set $n/2^k = 1$, $k = \log_2(n) = \log(n)$.

$T(n) = T(1) + n(2 - \frac{2}{n})$

$= T(1) + 2n - 2$

$= \theta(n)$

$T(n) = O(n)$

b) Find a Big-Oh bound for the following recurrence relation $T(n) = n + 2T(n/2)$. Justify your answer.

$T(n) = n + 2T(n/2)$

$= n + n + 4T(n/4)$

$= n + n + n + 8T(n/8)$

$= \ldots$

$= n * k + 2^k T(n/2^k)$

set $n/2^k = 1$, $k = \log_2 n = \log(n)$

$T(n) = n \cdot \log n + n T(1)$

$= \theta(n \log n + n)$

$T(n) = O(n \log n)$

6) Growth Rates

a) Order the following functions from slowest to fastest growth rate

- 2^{72}
- $n^2 \log n$
- $2^{n/2}$
- $\log n$
- $n \log n^2$
- n^6
- $n \log \log n$
- $n \log^2 n$
- n
- n^2
- $n \log n$
- 2^n
- $\log^2 n$
- $2/n$
- $n^{1/2}$

slow

↓

fast

$2/n$,

2^{72},

$\log n$,

$\log^2 n$,

$n^{1/2}$,

n,

$n \log \log n$,

$n \log n$,

$n \log n^2$

$n \log^2 n$,

n^2

$n^2 \log n$

n^6

$2^{n/2}$

2^n

7) Big-Oh Definition

Suppose T1(n) is O(f(n)) and T2(n) is O(f(n)). Which of the following are always true (for all T1, f, and T2)? You do not need to prove an item is true (just saying true is enough for full credit), but if an item is false, you need to give a counterexample to demonstrate it is false. To give a counterexample, give values for T1(n), T2(n), and f(n) for which the statement is false (for example, you could write, "The statement is false if T1(n) = 100n, T2(n) = $2n^2$, and f(n) = n^3"). Hints: Think about the definitions of big-O, big-Ω, and big-Θ.

a) T1(n)/T2(n) is O(1). False

The statement is false if $T1(n)=n$, $T2(n)=n^{1/2}$, $f(n)=n^2$, $T1(n)/T2(n)=n^{1/2}$

b) T1(n) + T2(n) is Ω(f(n)). False

It is false if $T1(n)=n$, $T2(n)=n^2$, $f(n)=n^3$; $T1(n)+T2(n)=n+n^2$

c) T1(n) – T2(n) is O(f(n)). True

Assuming $T1(n)\geq 0$, $T2(n)\geq 0$. for all valid n, then this statement is true.

d) T1(n) is O(T2(n)). False

It is false if $T1(n)=n^2$, $T2(n)=n$, $f(n)=n^3$,

e) T2(n) is Θ(T1(n)). False

It is false if $T1(n)=n^2$, $T2(n)=n$, $f(n)=n^3$,