Spring 2019 CS 372 Midterm Exam 1

Name: ______________Solutions_________

1. Order the following functions by growth rate from lowest to highest. If any are of the same order, circle them on your list.

2^n	$n^3 + \lg n$	$\lg n$	$\lg\lg n$	2^{n+1}	$n \lg n$
n	100	$n!$	3^n	$n^{1.03}$	$\sqrt{n}$

$$100$$
$$\lg\lg n$$
$$\lg n$$
$$\sqrt{n}$$
$$n$$
$$n \lg n$$
$$n^{1.03}$$
$$n^3 + \lg n$$
$$2^n, 2^{n+1}$$
$$3^n$$
$$n!$$

2. (a) Write a definition of the big-O notation.

$f(n) = O(g(n))$ if there exist two positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$

(b) Let $f(n) = 6n^4 + 40n^3 - 50n^2 + 8n - 1$ and $g(n) = n^4$. Prove that $f(n) = O(g(n))$ using the definition of Big-O notation. (You need to find constants c and n_0).

$$f(n) = 6n^4 + 40n^3 - 50n^2 + 8n - 1 \le_{if\ n \ge 1} 6n^4 + 40n^3 + 8n \le 6n^4 + 40n^4 + 8n^4 = 54n^4 = 54g(n)$$

Take $c = 54, n_0 = 1$

3. Assume that the partitioning algorithm in the Quicksort always produces a 8 - to - 3 proportional split.
   (a) Write a recurrence for the running time of Quicksort in this case. (Do not solve it.)

$$T(n) = \begin{cases} T\left(\frac{8n}{11}\right) + T\left(\frac{3n}{11}\right) + \Theta(n) & n > 1 \\ \Theta(1) & n = 1 \end{cases}$$

   (b) What is the running time of Quicksort in this case? (Just write your answer in asymptotic notation).

$$T(n) = \Theta(n \lg n)$$

4. (a) Write a recurrence which describes the running time of Mergesort. (Do not solve it.)

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + \Theta(n) & n > 1 \\ \Theta(1) & n = 1 \end{cases}$$

(b) Write a recurrence which describes the worst case running time of Quicksort. (Do not solve it.)

$$T(n) = \begin{cases} T(n-1) + \Theta(n) & n > 1 \\ \Theta(1) & n = 1 \end{cases}$$

(c) What is the best-case, worst-case and average-case running time of Mergesort?

Best-case: $\Theta(n \lg n)$
Worst-case: $\Theta(n \lg n)$
Average-case: $\Theta(n \lg n)$

(d) What is the best-case, worst-case and average-case running time of Insertion sort?

Best-case: $O(n)$
Worst-case: $O(n^2)$
Average-case: $O(n^2)$

(e) What is the best-case, worst-case and average-case running time of Quicksort?

Best-case: $\Theta(n \lg n)$
Worst-case: $\Theta(n^2)$
Average-case: $\Theta(n \lg n)$

5. Decide whether the following statements are True or False.

a) If f(n) = Θ(g(n)) and g(n) = Θ(h(n)), then h(n) = Θ(f(n)).

True

b) If f(n) = O(g(n)) and g(n) = O(f(n)) then f(n) = g(n).

False

c) $\frac{n}{100}$ = Ω(n).

True

6. Consider the following recursive function SlowSort:

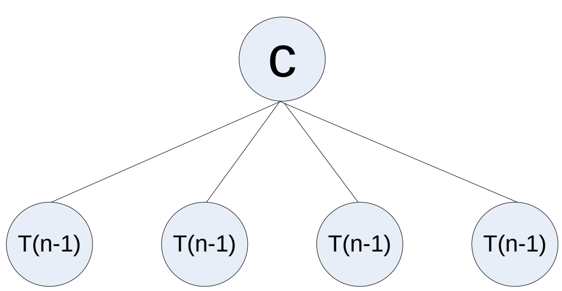
```
function SlowSort(A,p,q)
// Sorts the subarray A[p..q]
if ( q – p < 2) then
        if (q – p = 1) then
                // exchange A[p] and A[q]
                temp = A[q]
                A[q] = A[p]
                A[p] = temp
else
   middle = (p + q)/2
   SlowSort(A, p, middle)
   for i = middle to q–1
        if A[i] > A[i+1] then
            // exchange A[i] and A[i+1]
            temp = A[i]
            A[i] = A[i+1]
            A[i+1] = temp
   SlowSort(A, p, q – 1)
```

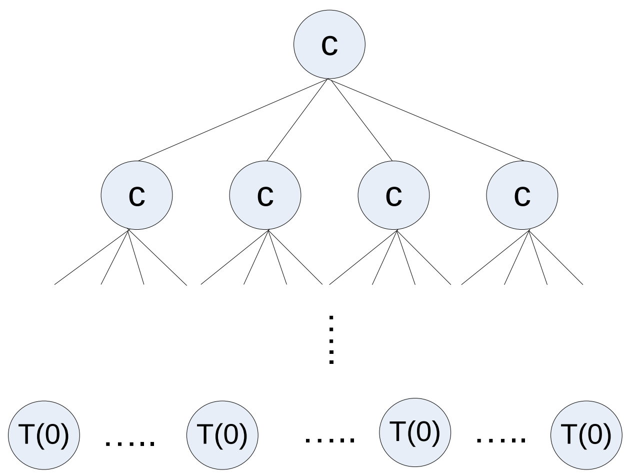
Let T(n) denote the running time of SlowSort on an input of size n (that is, n = q – p + 1). Derive a recurrence relation for T(n) including the base case. (Do not solve your recurrence.)

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + T(n-1) + \Theta(n) & n > 2 \\ \Theta(1) & n \le 2 \end{cases}$$

7. Use a recursion tree to solve the following recurrence:

$$\mathrm{T(n)} = \begin{cases} c & if\ n = 0 \\ 4T(n-1) + c & if\ n > 0 \end{cases}$$





Level	Number of nodes	Cost per node	Argument of T
0	1	c	n
1	4	c	n-1
2	4^2	c	n-2
…	…	….	…
i	4^i	c	n-i
…	…	….	…
k	4^k	T(0)=c	n-k

At the last level the argument is 0, so $n - k = 0 \rightarrow n = k$
Total cost:

$$T(n) = T(0)\left(4^k\right) + c\left(4^{k-1}\right) + \cdots + c(4^2) + c(4) + c = 4^k T(0) + c\sum_{i=0}^{k-1} 4^i = 4^k c + c\frac{4^k - 1}{4 - 1} = 4^n c + \frac{c(4^n - 1)}{3} = O(4^n)$$

8. For each of the following recurrences, solve it using one of the methods we had in class (the Master Theorem, recursion tree method, iteration method). Express final answers for T(n) in Big-O notation. Show your work.

1) T(n) = $\begin{cases} c & if\ n = 0 \\ \mathrm{T(n-1) + cn} & if\ \mathrm{n} > 0 \end{cases}$ 2) T(n) = $\begin{cases} c & if\ n = 1 \\ 9T\left(\frac{n}{3}\right) + cn^2 & if\ n > 1 \end{cases}$ 3) T(n) = $\begin{cases} c & if\ n = 1 \\ 8T\left(\frac{n}{4}\right) + cn^2 & if\ n > 1 \end{cases}$

1)

$$T(n) = \begin{cases} c & if\ n = 0 \\ T(n-1) + cn & if\ n > 0 \end{cases}$$

Using the iteration method:
$T(n) = T(n-1) + cn$
$\quad = T(n-2) + c(n-1) + cn$
$\quad = T(n-3) + c(n-2) + c(n-1) + cn$
⋮
The general term is:
$\quad = T(n-i) + c\big(n - (i-1)\big) + \cdots + c(n-1) + cn$
We stop at the base case when $n - i = 0 \rightarrow i = n$

$$T(n) = T(0) + c + c * 2 + \cdots + c(n-1) + cn = c + c\sum_{i=1}^{n} i = c + \frac{c(n+1)(n)}{2} = O(n^2)$$

2) Using the Master Theorem:

$$a = 9, b = 3, d = 2; \log_3 9 = 2 = d \rightarrow T(n) = O(n^2 \lg n)$$

3) Using the Master Theorem:

$$a = 8, b = 4, d = 2;\ \log_4 8 < 2 \rightarrow T(n) = O(n^2)$$