HW3

1-3. credits to Tom Clunie

#1: $3T(\frac{n}{2}) + T(\frac{n}{2}) + cn = 4T(\frac{n}{2}) + cn$

$a = 4,\ b = 2 \Rightarrow \log_b a = \log_2 4 = 2 \quad n^{\log_b a} = n^2$

$n^2 > f(n) = n^1$, so this is case #1 of the master theorem

So $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

#2 Recursion Tree of $T(n) = 4T(\frac{n}{2}) + cn$

① cn $\quad T(\frac{n}{2}) = 4T(\frac{1}{4}n) + \frac{cn}{2}$

$T(\frac{n}{4})\ T(\frac{n}{4})\ T(\frac{n}{4})\ T(\frac{n}{4})$

② cn $\quad T(\frac{n}{4}) = 4(\frac{1}{8}n) + \frac{cn}{4}$

$\frac{cn}{2}\ \frac{cn}{2}\ \frac{cn}{2}\ \frac{cn}{2}$

$T(\frac{n}{4})\ T(\frac{n}{4})\ T(\frac{n}{4})\ T(\frac{n}{4}) \ \ldots\ldots$

③

	level	problem size	total cost
cn	1	n	cn
$\frac{cn}{2}\ \frac{cn}{2}\ \frac{cn}{2}\ \frac{cn}{2}$	2	$\frac{n}{2}$	$2cn$
$\frac{cn}{4}\ \frac{cn}{4}\ \frac{cn}{4}\ \frac{cn}{4}\ \ldots$	3	$\frac{n}{4}$	$4cn$
$\frac{cn}{8}\ \frac{cn}{8}\ \frac{cn}{8}\ \frac{cn}{8}\ \ldots$	4	$\frac{n}{8}$	$8cn$
$c\ c\ c\ c$	i	$\frac{n}{2^{i-1}}$	$2^{i-1}cn$

$\Rightarrow T(n) = cn + 2cn + 4cn + 8cn + \ldots + 2^{i-1}cn = cn(1 + 2 + 4 + 8 + \ldots + 2^{i-1})$

Solve for i: $\frac{n}{2^{i-1}} = 1$; $n = 2^{i-1}$, $\lg n = i - 1$, $i = \lg n - 1$

$\Rightarrow cn \sum_{k=0}^{\lg n} 2^k \leq cn(2^{\lg n + 1} - 1) = cn((n)^{\lg 2} - 1) = cn(n + 1 - 1) = cn^2$

$\therefore T(n) = \Theta(n^2)$

#3) Substitution Method $4T(\frac{n}{2}) + cn$, $c > 0$

Guess: $T(n) = O(n^2 - n)$

$T(n) \leq dn^2 - en$, $d, e > 0$ (constants)

Assume $T(\frac{n}{2}) = d(\frac{n}{2})^2 - e\frac{n}{2} = d\frac{n^2}{4} - e\frac{n}{2}$

Substitute: $4(\frac{1}{4}dn^2 - \frac{1}{2}en) + cn$

$= dn^2 - 2en + cn$

Compare: $\leq dn^2 - en$

$cn \leq en$

$c \leq e$ $\therefore T(n) = O(n^2 - n)$

Guess: $T(n) = \Omega(n^2 - n)$

$T(n) \geq dn^2 - en$, $d > 0$, $e > 0$

$T(\frac{n}{2}) = d(\frac{n}{2})^2 - e(\frac{n}{2}) = \frac{1}{4}dn^2 - \frac{1}{2}en$

Substitute: $4(\frac{1}{4}dn^2 - \frac{1}{2}en) + cn$

$= dn^2 - 2en + cn$

Compare: $\geq dn^2 - en$

$cn \geq en$

$c \geq e$ $\therefore T(n) = \Omega(n^2 - n)$

$\therefore T(n) = \Theta(n^2 - n) = \Theta(n^2)$

*Note: upper bound proof usually starts with a guess of $O(n^2)$ and it won't work. Then make a new guess by subtracting a lower order term using $O(n^2-n)$.

Problem 4&5 from Jeffrey Ma

4. $T(n) = T(n/3) + T(n/3) + T(n/3) + T(n/3) + 2^{\lg n}$
$= 4T(n/3) + n^1$

$a=4 \quad b=3 \quad f(n) = n^1$

$n^{\log_3 4}$ vs. n^1 case 1 of MM
as $\log_3 4$ is greater than 1

$T(n) = \theta(n^{\log_3 4})$

5. ① $a=2 \quad b=4 \quad f(n)=n$

$n^{\log_4 2}$ vs n case 3 of MM
$n^{1/2}$ vs n as $1/2 < 1$ $T(n) = \theta(n)$

② $a=3 \quad b=2 \quad f(n) = \sqrt{10}\, n^2$

$n^{\log_2 3}$ vs $\sqrt{10}\, n^2$ case 3 of MM
$\log_2 3 < 2$ $T(n) = \theta(n^2)$

③ $a=2$ $n-1$ is the wrong format thus we cannot use the MM to solve the recurrence given

④ $a=3 \quad b=3/2 \quad f(n)=n \quad \log_{3/2} 3 \approx 2.71$

$n^{2.71}$ vs. n $2.71 > 1$ case 1 of MM
$T(n) = \theta(n^{\log_{3/2} 3})$

⑤ $a=3^n$ a is not a constant thus MM does not apply

6. from Victor M Espaillat

(1) (10 points) Pseudocode (*please use the textbook conventions*).

```
   Find_i_eq_elem(A, l, r)                                  Cost      Times
1.      if l == r                          // base case      c1        1
2.          if A[l] == l                                    c2        1
3.              return l                                    c3        1
4.          else
5.              return -1                  // none found     c5        1
6.      m = ⌊(l+r)/2⌋                                       c6        1
7.      if A[m] == m                                        c7        1
8.          return m                                        c8        1
9.      if A[m] > m                                         c9        1
10.         return Find_i_eq_elem(A, l, m)                  T(n/2)    1
11.     if A[m] < m                                         c11       1
12.         return Find_i_eq_elem(A, m+1, r)                T(n/2)    1
```

(2) (10 points) Analysis: Provide a tight upper bound on the worst-case asymptotic running time of your pseudocode (provide the recurrence and resolve it). Justify your answer (i.e., list the cost for executing each line of code and how you calculate the total running time).

Worst case: either line 10 or 12 is executed every time, until the base case.

$T(n) = T(n/2) + c$
$a = 1, b = 2, f(n) = cn^0$
$n^{\log_b a} = n^{\log_2 1} = n^0$

$$f(n) = \Theta(n^{\log_b a})$$
$$cn^0 = \Theta(n^0)$$
$$c = \Theta(1)$$

$f(n)$ is polynomially equal to $n^{\log_b a}$.
∴ Case 2 of Master Theorem : $T(n) = \Theta(n^{\log_b a} \lg n)$
$= \Theta(n^0 \lg n)$
$= \Theta(\lg n)$

(3) (10 points) Prove your answer using the **Substitution Method.**

Recurrence: $T(n) = T(n/2) + c$

① Upper bound

Guess: $T(n) = O(\lg n)$

$T(n) \le d \lg n$ (d : pos. constant)

$$T(n/2) \le d \lg(n/2)$$
$$\le d(\lg n - \lg 2)$$
$$\le d(\lg n - 1)$$
$$\le d \lg n - d$$

Substitute: $T(n) \le T(n/2) + c$

$$\le d \lg n - d + c$$
$$\le \underbrace{d \lg n}_{\text{guess}} \quad \text{if } (-d + c) \le 0$$
$$c \le d$$

$\therefore T(n) = O(\lg n)$

② Lower bound

Guess: $T(n) = \Omega(\lg n)$

$T(n) \ge d \lg n$ (d : pos. constant)

$$T(n/2) \ge d \lg(n/2)$$
$$\ge d \lg n - d$$

Substitute: $T(n) \ge T(n/2) + c$

$$\ge d \lg n - d + c$$
$$\ge \underbrace{d \lg n}_{\text{guess}} \quad \text{if } (-d + c) \ge 0$$
$$c \ge d$$

$\therefore T(n) = \Omega(\lg n)$

From ① and ②

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