Due:  Feb 21st  at the start of class

**Homework #2**

*CMSC351 - Spring 2013*

PRINT Name : 

- Grades depend on neatness and clarity.
- Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.
- Write your answers in the spaces provided. If needed, attach other pages.
- The grades would be out of 16. Four problems would be selected and everyone’s grade would be based only on those problems. You will also get 4 bonus points for trying to solve all problems.

1. Suppose you can only eat 1 or 2 apples every day. Let $F(0)=1$ and for $n \geq 1$ let $F(n)$ denote the different number of ways you can eat $n$ apples. Then $F(1)=1$ since your only option is to eat the apple on the first day. But $F(2)=2$ since you could either eat one apple each for two days, or eat both apples on the first day. Find a recurrence relation for $F(n)$. Show that $F(n) = \frac{1}{\sqrt{5}}(\alpha^n - \beta^n)$ where $\alpha = \frac{1+\sqrt{5}}{2}$ and $\beta = \frac{1-\sqrt{5}}{2}$.

For $n \geq 0$ suppose you want to eat $n+2$ apples. On the first day if you eat one apple then you have $F(n+1)$ choices to eat the remaining apples. However on the first day if you eat two apples then you have $F(n)$ choices to eat the remaining applies. Since these cases are disjoint we have the recurrence relation $F(n+2) = F(n+1) + F(n)$

We show the formula for $F(n)$ by induction.

**Base Case:** $n=0$ which is easy to verify.

**Inductive Hypothesis:** For all $k < n$ we have $F(k) = \frac{1}{\sqrt{5}}(\alpha^k - \beta^k)$

**Inductive Step:** Note that $\alpha, \beta$ are the roots of the equation $x^2 - x - 1 = 0$. Hence $\alpha^2 = \alpha + 1$ and $\beta^2 = \beta + 1$. Now $F(n) = F(n-1) + F(n-2)$ (by recurrence) $= \frac{1}{\sqrt{5}}(\alpha^{n-1} - \beta^{n-1}) + \frac{1}{\sqrt{5}}(\alpha^{n-2} - \beta^{n-2})$ (by I.H.) $= \frac{1}{\sqrt{5}}(\alpha^{n-2}(\alpha + 1) - \beta^{n-2}(\beta + 1))$ $= \frac{1}{\sqrt{5}}(\alpha^n - \beta^n)$

These are the famous Fibonacci numbers.
2. Solve the following recurrence relations using the Master Theorem, or just state that the Master Theorem does not apply.

(a) \( T(n) = T(n/2) + 2^n \)

Master Theorem does not apply since \( f(n) = 2^n \) is not equal to \( \theta(n^c) \) for any constant \( c \).

(b) \( T(n) = 2^n \cdot T(n/2) + n^n \)

Master Theorem does not apply since \( a \) is not a constant

(c) \( T(n) = 3 \cdot T(n/3) + \left(\frac{n}{2}\right) \)

This is Case 2 of Master Theorem with \( k = 0 \) since \( c = 1 = \log_b a \). Hence \( T(n) = \theta(n \cdot \log n) \)

(d) \( T(n) = 4 \cdot T(n/2) + n^{2.5} \)

This is Case 3 of Master Theorem since \( c = 2.5 > 2 = \log_b a \). Hence \( T(n) = \theta(n^{2.5}) \)
3. Function $T(n)$ is defined by the following recurrence relation.

$$T(n) = \begin{cases} 
8 & \text{if } 1 \leq n \leq 4, \\
4T\left(\left\lfloor \frac{n}{4} \right\rfloor \right) + 4n \log_2 n & \text{if } n > 4.
\end{cases}$$

Prove $T(n) = O(n \log^2 n)$.

(Hint: use induction)

We assume that there exist a constant $c$ such that $T(n) \leq c \cdot n \log^2 n$, for every $n \geq 2$. We prove this (and find the exact value of $c$) by strong induction.

The base case is for $n = 2, 3$ and $4$. Let us verify for $n=2$. We should have $8 \leq c \cdot 2 \log^2 2 = 2c$. Thus we should choose $c$ to be greater than or equal to 4. It is easy to verify that $c \geq 4$ works for $n=3$ and $n=4$ as well.

Now assuming that for all $4 < m < n$ we have $T(m) \leq c \cdot m \log^2 m$, we want to prove that $T(n)$ is also less than or equal to $c \cdot n \log^2 n$ (for every $n$ greater than 2). We have:

$$T(n) = 4T\left(\left\lfloor \frac{n}{4} \right\rfloor \right) + 4n \log_2 n \leq 4c \cdot \left\lfloor \frac{n}{4} \right\rfloor \log^2 \left(\left\lfloor \frac{n}{4} \right\rfloor \right) + 4n \log n$$

$$\leq 4c \cdot \frac{n}{4} \log^2 \left(\frac{n}{4}\right) + 4n \log n = cn(\log n - 2)^2 + 4n \log n$$

$$= cn(\log^2 n - 4 \log n + 4) + 4n \log n = cn \log^2 n - 4n (c \log n - \log n - c)$$

Since $n > c \geq 4$ we have $\log n \geq 2 > \frac{4}{3} \geq \frac{c}{c-1}$, and so the term $(c \log n - \log n - c)$ would be always positive. Therefore $T(n)$ would not be greater than $cn \log^2 n$.

[This proof is complete. However, since now we know that $c \geq 4$ is sufficient, one may find it easier to rewrite all the proof by putting 4 instead of $c$.]
4. [Prob 3.22-a,Pg 58] Solve the following recurrence relation. It is sufficient to find the asymptotic behavior of $T(n)$. (Hint: Substitute another variable for $n$)

$$
T(n) = \begin{cases} 
1 & \text{if } n = 2, \\
4T(\lceil \sqrt{n} \rceil) + 1 & \text{if } n > 2.
\end{cases}
$$

We substitute $n = 2^k$. Define a new function $F(k)$ which is equal to: $F(k) = T(2^k)$.

We note that $k = \log n$ and $\sqrt{n} = 2^{k/2}$. By the definition of $T(n)$, we have

$$
F(k) = \begin{cases} 
1, & k = 0 \\
4F(k/2) + 1, & k > 0
\end{cases}
$$

Use Theorem 3.4 with $a=4$, $b=2$ and $k=0$. Since $a > b^k$ we have $F(k) = O(k^2)$ and therefore $T(n) = O(\log^2 n)$. 
5. Finding the majority: We have \( n \) numbers such that one number has appeared at least \( \left\lfloor \frac{n}{2} \right\rfloor + 1 \) times. Design an algorithm that finds this number by at most \( n \) comparisons between given numbers.

Let \( A[1], ..., A[n] \) be the input. Assume that for some \( i \), there are exactly \( i/2 \) appearances of element \( A[1] \), in the range \( A[1], ..., A[i] \). One can easily prove that if \( v \) is the element with majority in the whole input, then \( v \) is still the majority in the range \( A[i+1], ..., A[n] \). Thus we can completely ignore the first \( i \) elements and process the rest of elements similarly, i.e., find the first position that \( A[i+1] \) has the same occurrences as all other elements together. Ignore this part and continue to the rest of elements. In any of these steps, if we couldn’t find such position, it means that the first of the remaining elements is the majority. The pseudocode is presented below:

Input: Array \( A[1..n] \)
Output: key
key ← \( A[1] \)  //key is the first element of each part as mentioned above.
num ← 0  //num is the number of occurrences of key in the remaining elements as mentioned above.

for (\( i \) ← 1 to \( n \))
  if (\( A[i] \) == \( key \))
    num ← num + 1  //found another occurrence of key
  else
    num ← num - 1  //found an element different from key
    if (num == 0)
      key ← \( A[i+1] \)  //here we have found a position where key has the same occurrences. Thus we should update the key.
  end if
end for
Print key
6. [Prob 5.22, Pg 116] Towers of Hanoi: There are \( n \) disks of different sizes arranged on a peg in decreasing order of sizes. There are two other empty pegs. The purpose of the puzzle is to move all the disks, one at a time, from the first peg to another peg in the following way. Disks are moved from the top of one peg to the top of another. A disk can be moved to a peg only if it is smaller than all other disks on that peg. In other words, the ordering of disks by decreasing sizes must be preserved at all times. The goal is to move all the disks in as few moves as possible.

a) Design an algorithm (by induction) to find a minimal sequence of moves that solves the towers of Hanoi problem for \( n \) disks.

Consider \( move(n, i, j) \) as the set of moves which can move the top \( n \) disks of peg \( i \) to peg \( j \), assuming that all the previous disks in peg \( j \) and peg \( k \) (the third peg) are bigger than the first \( n \) disks in peg \( i \). Now we can find this set of moves by this recursive definition (why?):

\[
move(n, i, j) = move(n - 1, i, k). move(1, i, j). move(n - 1, k, j)
\]

where \( move(1, i, j) \) is just moving the topmost disk from \( i \) to \( j \).

b) How many moves are used in your algorithm? Construct a recurrence relation for the number of moves, and solve it.

Note that the number of moves in \( move(n, i, j) \) depends only on \( n \). We call this \( h(n) \). Using the recursive definition of the algorithm, we have \( h(n) = 2h(n - 1) + 1 \) and \( h(1) = 1 \). Solving this recursive formula we get \( h(n) = 2^n - 1 \).

c) Prove that the number of moves in part b is optimal; that is, prove that no algorithm can use fewer moves (use induction).

Let \( opt_n \) denote the minimum number of moves for moving \( n \) disks from one peg to another. We use induction to show \( opt_n \geq 2^n - 1 \). The base case is obvious. Suppose that we know the optimum movement scheme. To prove the inductive step, WLOG, we assume that at the end of optimum movement all the disks would be in peg 3. At some point, we have to move the biggest disk to peg 3 from some peg, say peg 1. One crucial observation is that in order to move the biggest disk from peg 1 to peg 3, we need to first move all the other \( n-1 \) disks from peg 1 to peg 2, hence using at least \( opt_{n-1} \) moves. Then we move the biggest disk and after that we again need to move the disks on peg 2 back to peg 3, thus paying another \( opt_{n-1} \). Therefore:

\[
\begin{align*}
\text{opt}_n \geq opt_{n-1} + 1 + opt_{n-1} \geq (by \ induction) \ 2^{(n-1)} - 1 + 1 = 2^n - 1.
\end{align*}
\]
7. A board of size $2^n \times 2^n$ has an arbitrary square cut out of it. Prove the remaining board can be tiled using tiles of the following shape (rotation and reflection are allowed)? (Prove the statement for every $n>0$)

Here is the solution for a board of size $2^2 \times 2^2$.

Decompose the $2^n \times 2^n$ square into four squares of size $2^{n-1} \times 2^{n-1}$ and then use induction.

**Base Case:** $n=1$. This is easy to verify.

**Inductive Hypothesis:** For all $k<n$ given a board of size $2^k \times 2^k$ with an arbitrary square cut out of it, we can cover the remaining board with tiles of given shape.

**Inductive Step:** Consider a board of size $2^n \times 2^n$. Decompose it into four sub-squares say A, B, C, D of size $2^{n-1} \times 2^{n-1}$. Now the $1 \times 1$ square cut out of our board must be from one of the sub-squares, say A. Hence by induction hypothesis we can now cover the remaining part of A by tiles. Now cover the central $1 \times 1$ squares of B, C and D using one tile. Now each of B, C and D can be thought as having one $1 \times 1$ square “cut out”. Hence by induction hypothesis we can cover the remaining parts of B, C, D as well.