COMP 250, Fall 2014 - Homework 2

Hamed Hatami

October 5, 2014

This homework is due in whole on October 1st 2014, 23:59, on MyCourses. There is no programming assignment this time, so you only need to turn-in a paper copy of your work. Each student must submit his/her own solution - no team work is allowed for this assignment.

Question 1 (5 points)

Prove by induction that for all positive integers k, n, we have $(1+k)^n \geq 1+kn$.

Solution: *Base case:* Take $n = 1$, then obliviously for every $k > 0$ we have $(1+k)^1 = 1+k \geq 1+k\times 1$.

Induction hypothesis: Suppose that for a fixed n, the following holds: For every $k > 0$, we have $(1+k)^n \geq 1+kn$.

Induction Step: We want to show that for every $k > 0$, we have $(1+k)^{n+1} \geq 1+k(n+1)$. Indeed by induction hypothesis, we have $(1+k)^{n+1} = (1+k)^n(1+k) \geq (1+kn)(1+k) = 1+k(n+1)+k^2n \geq 1+k(n+1)$.

Question 2 (10 points)

Prove by induction that $6^{n+2}+7^{2n+1}$ is divisible by 43 for every integer $n \geq 0$.

Solution: *Base case:* Take $n = 0$, then $6^{n+2}+7^{2n+1} = 6^2+7 = 43$ which is divisible by 43.

Induction hypothesis: For a fixed n, $6^{n+2}+7^{2n+1}$ is divisible by 43.

Induction Step: We want to show that $6^{(n+1)+2}+7^{2(n+1)+1}$ is divisible by 43. Note

$$6^{(n+1)+2}+7^{2(n+1)+1} = 6\times 6^{n+2}+49\times 7^{2n+1} = 6\times(6^{n+2}+7^{2n+1})+43\times 7^{2n+1},$$

which is divisible by 43 as the first term is divisible by 43 by induction hypothesis, and the second term is divisible by 43 obviously.

Question 3 (10 points)

Recall the definition of the Fibonacci numbers: $F(0) = 0$ and $F(1) = 1$ and $F(n) = F(n-1) + F(n-2)$ for $n \geq 2$. Prove by induction that

$$F(n) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

Solution: *Base case:* For $n = 0$ and $n = 1$ we have

$$\frac{\left(\frac{1+\sqrt{5}}{2}\right)^0 - \left(\frac{1-\sqrt{5}}{2}\right)^0}{\sqrt{5}} = 1 - 1 = 0 = F(0),$$

and

$$\frac{\left(\frac{1+\sqrt{5}}{2}\right)^1 - \left(\frac{1-\sqrt{5}}{2}\right)^1}{\sqrt{5}} = \frac{\sqrt{5}}{\sqrt{5}} = 1 = F(1).$$

Strong induction hypothesis: Consider a fixed $n \geq 1$ and suppose that for every $0 \leq m \leq n$ we have

$$F(m) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^m - \left(\frac{1-\sqrt{5}}{2}\right)^m}{\sqrt{5}}.$$

Induction Step: By induction hypothesis, we have

$$\begin{aligned}
F(n+1) &= F(n) + F(n-1) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}}{\sqrt{5}} + \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n} - \left(\frac{1-\sqrt{5}}{2}\right)^{n}}{\sqrt{5}} \\
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\left(1 + \frac{1+\sqrt{5}}{2}\right) - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\left(1 + \frac{1-\sqrt{5}}{2}\right)}{\sqrt{5}} \\
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\left(\frac{1+\sqrt{5}}{2}\right)^{2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\left(\frac{1-\sqrt{5}}{2}\right)^{2}}{\sqrt{5}} \\
&= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}{\sqrt{5}}.
\end{aligned}$$

Question 4 (15 points)

Write a recursive algorithm that, given a positive integer $n \geq 1$, prints all sequences of numbers $i_1, \ldots, i_k$ such that

- $k \geq 1$ and

- $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.

For example if $n = 3$, then the output will be:
1
1, 2
1, 2, 3
1, 3
2
2, 3
3

Solution:
Algorithm recursiveSequence(A, k, n)
Input: An array $A[]$ currently containing k elements
if $(k == 0)\{\text{min} \leftarrow 0\}$ else {
    min $\leftarrow A[k-1]$
    printArray$(A, k)\backslash\backslash$ prints the sequence in the array
}
for $i \leftarrow min + 1$ **to** n **do**
    $A[k] \leftarrow i$
    recursiveSequence$(A, k+1, n)$

Question 5 (15 points)

The "Towers of Hanoi" problem is the following: You are given three vertical rods (the towers), numbered 1, 2, and 3. On the first rod are stacked n rings of decreasing size (see figure). The goal is to transfer all rings on the second rod. The only operation allowed is to take a single ring from the top of one of the towers and to move it to the top of another tower, *provided the ring is not larger than the ring already on top of that tower (if any).* In other words, for each towers and at any point in time, the rings have to be stacked in decreasing size.

The question is to write a recursive algorithm that prints the series of moves required to transfer all n rings from rod i to rod j. This seems awfully difficult until you realize the following. Assume rod i has the m smallest rings of all three towers. To transfer the m smallest rings from rod i to rod j, all one needs to do is to
1) first transfer the $m-1$ smallest rings from rod i to rod $k = 6 - i - j$,
2) then transfer the m-th ring from rod i to rod j,
3) finally re-transfer the $m-1$ smallest rings from rod k to rod j.

Notice that step (1) and step (3) are just smaller versions of the same original problem. See for example the procedure described in the figure for $m = 3, i = 1, j = 2$. The question is to write a procedure Hanoi(m, i, j) that prints (using the pseudocode keyword "print") the series of moves needed to

move the top m rings from rod i to rod j. For example, Hanoi(3,1,2) should print:
Move top ring from rod 1 to rod 2
Move top ring from rod 1 to rod 3
Move top ring from rod 2 to rod 3
Move top ring from rod 1 to rod 2
Move top ring from rod 3 to rod 1
Move top ring from rod 3 to rod 2
Move top ring from rod 1 to rod 2
a) (7 points) Complete the pseudocode below:
Algorithm $Hanoi(m, i, j)$
Input: The number m of rings to be moved from rod i to rod j. It is assumed that rod i has at least m rings and that these are the m smallest of all three rods.
Output: Prints the series of step required to move the m smallest rings from rod i to rod j.
/***Solution:***/
if $(m > 0)$ then
. Hanoi$(m-1, i, 6-i-j)$
. print "Move top ring from rod" $+ i +$ " to rod "$+j$
. Hanoi$(m-1, 6-i-j, j)$

b) (3 points) Let $T(m)$ be the number of moves required to move m rings from one rod to another using your algorithm. Give a recurrence expressing $T(m)$ in terms of $T(m-1)$.

Solution: By analyzing the algorithm, we get that $T(0) = 0$ and $T(m) = T(m-1) + 1 + T(m-1) = 2T(m-1) + 1$ for $m > 0$.
c) (5 points) Guess an explicit formula for $T(m)$ (i.e. a formula that doesn't have $T(m-1)$ on the right part) and prove by induction that your guess was correct for any $m \geq 0$.

Solution: A little bit of intuition (or a more methodic substitution approach) yields the guess $T(m) = 2^m - 1$. This is easy to prove by induction on m.
Base case: for $m = 0$, the definition of $T(m)$ says that $T(0) = 0$, which is indeed equal to $2^0 - 1$.
Induction hypothesis: Assume that $T(b) = 2^b - 1$.
We need to prove that $T(b+1) = 2^{b+1} - 1$. We have:

$$\begin{aligned} T(b+1) &= 2T(b) + 1 \text{ by definition of T} \\ &= 2(2^b - 1) + 1 \text{ by I.H.} \\ &= 2^{b+1} - 1 \end{aligned}$$

Thus, by the base case and the inductive step, we have shown that $T(m) = 2^m - 1$ for any $m \geq 0$. It actually turns out that one can show that this recursive

algorithm for the Hanoi Towers problem is the one that yields the minimal number of moves needed to transfer the rings.

Question 6 (7 points)

Prove, using only the definition of $O()$, that $n^2 + 10n\log(n)$ is $O(n^2)$.

Solution: Since $\log(n) \leq n$ for every n, we have $n^2 + 10n\log(n) \leq n^2 + 10n^2 = 11n^2$. Hence taking $n_0 = 1$ and $c = 100$ we see that for every $n \geq n_0$ we have

$$n^2 + 10n\log(n) \leq cn^2.$$

Question 7 (8 points)

Prove, using only the definition of $O()$, that $2^{\sqrt{n}}$ is *not* $O(n^{10})$.
Answer: By L'Hopital we can see that $\lim_{n\to\infty} \frac{\sqrt{n}}{10\log_2 n} = \infty$. This in turn implies that

$$\lim_{n\to\infty} \frac{2^{\sqrt{n}}}{n^{10}} = \infty.$$

Hence for every $n_0 > 0$ and $c > 0$, there always exists an $n > n_0$ such that $2^{\sqrt{n}} > cn^{10}$.

Question 8 (15 points)

Give a modified version of the mergeSort algorithm so that given an array A of n numbers, it outputs the number of pairs $i < j$ such that $A[i] > A[j]$. The running time of your algorithm must still be $T(n) = 2T(n/2) + C_1 n + C_2$ where C_1 and C_2 are constants. For example, on given array A = [5 2 6 7 1], the algorithm should return 5, because $5 > 2, 5 > 1, 6 > 1, 7 > 1, 2 > 1$.
Solution sketch: We modify mergeSort$(A, \text{left}, \text{right})$ algorithm so that it sorts the array between *left* and *right*, furthermore returns the number of such pairs in that range. It returns $A + B + C$, where

$$A = \text{mergeSort}(A, \text{left}, \text{mid}),$$

$$B = \text{mergeSort}(A, \text{mid} + 1, \text{right}),$$

$$C = \text{merge}(A, \text{left}, \text{mid}, \text{right}),$$

where now merge$(A, \text{left}, \text{mid}, \text{right})$ merges the two sorted parts of the array and furthermore counts the number of pairs left $\leq i \leq$ mid and mid $+ 1 \leq j \leq$ right such that $A[i] > A[j]$.

It remains to modify merge$(A, \text{left}, \text{mid}, \text{right})$ to achieve this task: Now in merge$(A, \text{left}, \text{mid}, \text{right})$ we have a variable *count* which is initiated to count $\leftarrow$ 0, and whenever we make a *right selection* we increase count by the number

of elements that are remained in the left part of the array (we set count $\leftarrow$ count $+ (\text{mid} - \text{indexLeft} + 1)$). This is because the selected element in the right part of the array is smaller than all the elements that are remained in the left part of the array.

Question 9 (15 points)

The algorithm below, called insertionSort, sorts in increasing order the elements of an array.
Algorithm insertionSort(A,n)
Input: An array $A[0...n-1]$ of n elements
Output: The array A is sorted
for $i \leftarrow 1$ **to** $n-1$ **do**
. $x \leftarrow A[i]$
. $j \leftarrow i-1$
. **while** ($j \geq 0$ **and** $x < A[j]$) **do**
. $A[j+1] \leftarrow A[j]$
. $j \leftarrow j-1$
. $A[j+1] \leftarrow x$

a) (5 points) What kind of input will yield the worst possible running time, for any fixed size n? Why?
Solution:
When A is sorted in decreasing order, the inner loop runs until $j = 0$.
b) (10 points) Let $T(n)$ be the number of primitive operations performed in this worst case. Write, for each line of the pseudocode, the number of primitive operations executed. Sum up everything to obtain $T(n)$. If $T(n)$ contains summations, replace them by their explicit value. Show your work step by step like we did in class.
Solution:
for $i \leftarrow 1$ **to** $n-1$ **do** 1 op + 2 op inside loop
. $x \leftarrow A[i]$ 2 ops
. $j \leftarrow i-1$ 2 ops
. **while** ($j \geq 0$ **and** $x < A[j]$) **do** 4 ops
. $A[j+1] \leftarrow A[j]$ 4 ops
. $j \leftarrow j-1$ 2 ops
. $A[j+1] \leftarrow x$ 3 ops, + 1 op for i++

In the worse case, the while loop is executed i times, so at the i-th iteration, the total running time of the while loop is $10i$, and the total running time of the i-th iteration is $10 + 10i$. To get the total running time of the algorithm, we need to sum the running time of each iteration. Thus, we get:

$$\begin{aligned} T(n) &= 1 + \sum_{i=1}^{n-1} 10 + 10i \\ &= 1 + 10(n-1) + 10\sum_{i=1}^{n-1} i \\ &= 1 + 10(n-1) + 10(n-1)n/2 \\ &= 5n^2 + 5n - 9 \end{aligned}$$

Thus, $T(n) = 5n^2 + 5n - 8$.

Question 10 Bonus (10 points)

Consider $2n$ distinct points in the plane. Prove by induction that we can partition the points into n pairs such that the line segments between no two pairs cross.

Solution: *Base case:* Take $n = 1$, then there is only one line segment and the statement is trivial.

Induction hypothesis: Suppose that for a fixed n, we can always partition a set if $2n$ points into n pairs such that the line segments between no two pairs cross.

Induction Step: Consider $2n$ points. Sort them so that if you list their coordinates $(x_1, y_1), \ldots, (x_{2n+2}, y_{2n+2})$, then $x_1 \leq x_2 \leq \ldots \leq x_{2n+2}$ and furthermore for $i = 1 \ldots, 2n+1$, if $x_i = x_{i+1}$ then $y_i \leq y_{i+1}$ (we are breaking the ties by looking at the y coordinates). Now pair the last two points (x_{2n+1}, y_{2n+1}) and (x_{2n+2}, y_{2n+2}). Note that no line segment between no other pair of points can intersect the line segment between these two. Moreover, by induction hypothesis we can pair the other $2n$ points $(x_1, y_1), \ldots, (x_{2n}, y_{2n})$ so that line segments between no two pairs of them cross, and since they can't also cross the line segment between (x_{2n+1}, y_{2n+1}) and (x_{2n+2}, y_{2n+2}), we are done.