Due Fri Sep 10 at 11:59PM

1. (20 points) Define the Foonacci sequence as follows: $f(0) = -1, f(1) = -5, f(2) = 1$, and $f(n) = 3f(n-1) + 4f(n-2) - 12f(n-3)$ for $n > 2$.

   Prove: Using induction (weak or strong), show that $f(n) = 3^n - 3(2^n) + (-2)^n$ for all $n \geq 0$.

   Answer:

   We proceed by strong induction on n.

   Basis step: we consider $n = 0$, $n = 1$, and $n = 2$.
   $f(0) = -1 = 1 - 3(1) + 1$.
   $f(1) = -5 = 3 - 3(2) + (-2)$.
   $f(2) = 1 = 9 - 3(4) + 4$.
   This completes the basis step.

   Inductive step: the inductive hypothesis is "$f(k) = 3^k - 3(2^k) + (-2)^k$ for $0 \leq k \leq n$" for some $n \geq 2$. By using this, we can show the relation for $n+1$:

$$\begin{align}
f(n+1) &= 3f(n) + 4f(n-1) - 12f(n-2) \tag{1}\\
&= 3\big(3^n - 3(2^n) + (-2)^n\big) + 4\big(3^{n-1} - 3(2^{n-1}) + (-2)^{n-1}\big) - 12\big(3^{n-2} - 3(2^{n-2}) + (-2)^{n-2}\big) \tag{2}\\
&= 3(3^n) - 9(2^n) + 6(2^{n-1}) + 3(-2)^n + 10(-2)^{n-1} \tag{3}\\
&= 3(3^n) - 6(2^n) - 2(-2)^n \tag{4}\\
&= 3^{n+1} - 3(2^{n+1}) + (-2)^{n+1} \tag{5}
\end{align}$$

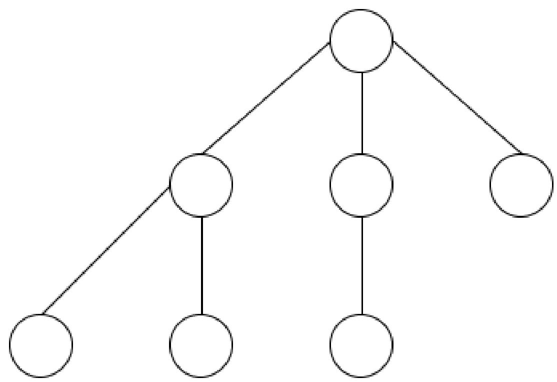
   This completes the inductive step.

   Therefore $f(n) = 3^n - 3(2^n) + (-2)^n$ for all $n \geq 0$.

2. (20 points) Let T be a tree with $n > 1$ vertices where every vertex has degree at most 3 (the degree of a vertex is the number of edges incident to it). Let $S_d(T)$ be the set of vertices in T that have degree exactly d.

   Prove: Using induction (weak or strong), show that for any such T, $|S_1(T)| - |S_3(T)| = 2$ holds.

   For example, in the example below there are 4 vertices of degree 1, 1 vertex of degree 2, and 2 vertices of degree 3. $|S_1(T)| - |S_3(T)| = 4 - 2 = 2$.



Answer:

We proceed by weak induction on n.

Basis step: we consider $n = 2$.
T is a single edge. $|S_1(T)| - |S_3(T)| = 2 - 0 = 2$. This completes the basis step.

Inductive step: the inductive hypothesis is "$|S_1(T')| - |S_3(T')| = 2$ for any such T' of size n" for some $n > 1$. Now we consider such a tree T of size $n+1$. Since $n > 1$, T contains an vertex v of degree 1. If we were to remove v and its edge, we would get a tree T' which satisfies the conditions of the inductive hypothesis. Let v's neighbor be u. u has degree at least 2 since $n > 1$. From here we break into cases based on the degree of u. Since $n > 1$, we know u does not have degree 1.

Case 1: u has degree 3.
Then in T', u has degree 2 and all other vertices' degrees are unchanged. So $|S_1(T)| = |S_1(T')| + 1$ and $|S_3(T)| = |S_3(T')| + 1$, meaning that $|S_1(T)| - |S_3(T)| = |S_1(T')| - |S_3(T')| = 2$ by the inductive hypothesis.

Case 2: u has degree 2.
Then in T', u has degree 1 and all other vertices' degrees are unchanged. So $|S_1(T)| = |S_1(T')|$ and $|S_3(T)| = |S_3(T')|$, meaning that $|S_1(T)| - |S_3(T)| = |S_1(T')| - |S_3(T')| = 2$ by the inductive hypothesis.

So in either case we get $|S_1(T)| - |S_3(T)| = 2$. This completes the inductive step.

Therefore $|S_1(T)| - |S_3(T)| = 2$ for all trees T of size $n > 1$ where every vertex has degree at most 3.

3. (20 points) For the following pairs of functions, relate one function to the other with a big O bound, or with a big Theta bound if applicable. For each pair of functions, state **and prove** whether the first function is big O, Theta, or big Omega of the second function. (If Theta is possible, you must prove Theta).

   - n^3 and $n^3 + 6n^2$
   - $n!$ and n^n
   - $8^{\log_2(n)}$ and n^3
   - $n + \ln(n)$ and $\ln(n^n)$
   - $\log_2(n)$ and $\log_{10}(100n)$
   - $\ln(n^2) - \ln(2n)$ and $\log_2(16^n)/\sqrt{n}$

Answer:

- $n^3 + 6n^2 = \Theta(n^3)$
  $c_1 = 1, c_2 = 7, n_0 = 1$. For $n > 1$, we have $n^3 + 6n^2 > n^3$. For $n > 1$, we have $n^2 < n^3$ so $n^3 + 6n^2 < 7n^3$.

- $n! = O(n^n)$
  $c = 1, n_0 = 1$. $n! = 1 \cdot 2 \cdot 3 \ldots n \leq n \cdot n \cdot n \ldots n = n^n$.
- $8^{\log_2(n)} = n^3 = \Theta(n^3)$
  $c_1 = 1, c_2 = 1, n_0 = 1$. $n^3 \leq n^3 \leq n^3$.
- $\ln(n^n) = n * \ln(n)$, $n + \ln(n) = O(n * \ln(n))$
  $c = 2, n_0 = e$. For $n > e$, we have $\ln(n) > 1$. So $n + \ln(n) \leq n\ln(n) + n\ln(n) = 2n\ln(n)$.
- $\log_{10}(100n) = \log_2(100n)/\log_2(10) = \frac{\log_2(100)+\log_2(n)}{\log_2(10)}, \log_2(n) = \Theta(\frac{\log_2(100)+\log_2(n)}{\log_2(10)})$
  $c_1 = \log_2(10)/2, c_2 = \log_2(10), n_0 = 100$. For $n > 100$ we have $\log_2(n) > \log_2(100)$. So $(\log_2(100) + \log_2(n))/2 \leq 2\log_2(n)/2 = \log_2(n) \leq \log_2(100) + \log_2(n)$.
- $\ln(n^2) - \ln(2n) = \ln(n/2)$ and $\log_2(16^n)/\sqrt{n} = 4\sqrt{n}$, $\ln(n/2) = O(4 * \sqrt{n})$
  $\lim_{n\to\infty} \frac{\ln(n/2)}{4\sqrt{n}} = \lim_{n\to\infty} \frac{1/n}{2/\sqrt{n}} = \lim_{n\to\infty} \frac{1}{2\sqrt{n}} = 0$ by L'Hopital's rule.

4. (20 points) For the following code segment, provide a tight big O bound on the number of times "foo()" is called **with a proof** for why this bound is tight:

```
while n > 1 do
    for i = 1 to n
        k = n;
        while k > 1 do
            foo();
            k = k / 3;
    n = n / 9
```

Answer: This code runs foo() $\sum_{i=0}^{\log_9 n} n/9^i * \log_3(n/9^i)$ times
$n * \sum_{i=0}^{\log_9 n} [\log_3(n) - \log_3(9^i)]/9^i$ times
$n * \sum_{i=0}^{\log_9 n} [\log_3(n) - 2i]/9^i$ times
$n * [\log_3(n) * \sum_{i=0}^{\log_9 n} 1/9^i - \sum_{i=0}^{\log_9 n} 2i/9^i]$ times
$\log_3(n)$ sum term dominates other and $\sum_{i=0}^{\log_9 n} 1/9^i < (9/8)$ therefore:
$O(n * \log(n))$

5. (20 points) Let $T(n)$ satisfy the following recurrence

$$T(n) = T(n/2) + 1 \tag{6}$$

Prove: Assuming $T(1) = 0$ and $n = 2^m$ where m is a natural number bigger than zero, show using the telescoping method (or using a recursion tree) that $T(n) = m$. Do not use the Master Theorem.

Answer: For $n > 1$:

$$T(n) = T(n/2) + 1 \tag{7}$$
$$= (T(n/4) + 1) + 1 \tag{8}$$
$$= T(n/4) + 2 \tag{9}$$
$$\ldots \tag{10}$$
$$= T(n/n) + m \tag{11}$$
$$= 0 + m \tag{12}$$
$$= m \tag{13}$$

Each iteration we are dividing the argument of T by 2 and adding 1. When we have divided by n, it is because two to the power of the number of iterations is equal to n. I.e. the number of iterations equals $\log(n) = m$. Thus when we have divided by n, we have added m.

6. (20 points) Consider the following pseudo code which presents a variant of the merge sort algorithm:

```
variantsort(Array A){
    n = size_of(A);

    if (n == 1){
        return;
    }

    j = 1;
    A1,A2,A3 = [];
    for i = 1 to n/3:{
        A1[j] = A[i];
        j = j+1;
    }

    j = 1;
    for i = n/3+1 to 2n/3:{
        A2[j] = A[i];
        j = j+1;
    }

    j = 1;
    for i = 2n/3 + 1 to n:{
        A3[j] = A[i];
        j = j+1;
    }

    variantsort(A1);
    variantsort(A2);
    variantsort(A3);

    A4 = merge(A1,A2);
    A5 = merge(A4,A3);

    return A5;
}
```

Assume that the function C = merge(A,B) takes two sorted arrays A (of size n_1) and B (of size n_2) and combines and returns them as one big sorted array C (of size $n_1 + n_2$) in $n_1 + n_2$ steps.

Solve:

(a) State the recurrence relation for the running time of the above pseudocode for an input of size n (assume n is a power of 3). (5 points)

(b) Solve the recurrence relation you obtained in the previous step. Do not use the master theorem. (15 points)

You can present the recurrence relation and the final solution in big O notation wherever appropriate. Note that you have to use the best possible bound for the big O notation i.e., you have to use O(n^3), not O(n^4) if the algorithm runs in $n^3 + 2n^2$ steps.

Answer:

(a)

$$\begin{aligned} T(n) &= \frac{n}{3} + \frac{n}{3} + \frac{n}{3} + T\left(\frac{n}{3}\right) + T\left(\frac{n}{3}\right) + T\left(\frac{n}{3}\right) + \left(\frac{n}{3} + \frac{n}{3}\right) + \left(\frac{n}{3} + \frac{2n}{3}\right) && (14) \\ &= n + 3T\left(\frac{n}{3}\right) + \frac{5n}{3} && (15) \\ &= \frac{8n}{3} + 3T\left(\frac{n}{3}\right) && (16) \\ &= O(n) + 3T(\frac{n}{3}) && (17) \end{aligned}$$

(b) For $n > 1$:

$$\begin{aligned} \frac{T(n)}{n} &= \frac{8}{3} + \frac{3}{n}T\left(\frac{n}{3}\right) && (18) \\ & && (19) \\ &= \frac{3}{n} \cdot \left(\frac{8n}{9} + 3T\left(\frac{n}{9}\right)\right) + \frac{8}{3} && (20) \\ &= \frac{9}{n}T\left(\frac{n}{9}\right) + \frac{8}{3} + \frac{8}{3} && (21) \\ \dots & && (22) \\ &= \frac{n}{n}T(n/n) + \frac{8}{3} + \frac{8}{3} + \ldots + \frac{8}{3} && (23) \\ &= 1 + \frac{8}{3}\log_3(n) && (24) \end{aligned}$$

Thus, we have $T(n) = n \cdot (1 + \frac{8}{3}\log_3(n)) = O(n\log(n))$

Each iteration we are dividing the argument of T by 3, multiplying its coefficient by 3, and adding $\frac{8}{3}$. When we have divided by n, it is because three to the power of the number of iterations is equal to n. I.e. the number of iterations equals $\log_3(n)$. Thus when we have divided by n, the coefficient of $T(n/n)$ has been multiplied by $3^{\log_3(n)} = n$ and we have added $\frac{8}{3}\log_3(n)$.

Note that we have assumed $T(1) = 1$ here. But, one could also assume it to be $T(1) = 0$ and arrive at $T(n) = n \cdot \frac{8}{3}\log_3(n)$, which is still $O(n\log(n))$.