Use the iteration technique to find a Big-Oh bound for the recurrence relation below. Note you may find the following mathematical result helpful: $2^{log_3 n} = n^{log_3 2}$, $\sum_{i=0}^{\infty}(2/3)^i = 3$

$$T(n) = 2T\left(\frac{n}{3}\right) + cn$$
$$T(n) = 2(2T\left(\frac{n}{9}\right) + c\left(\frac{n}{3}\right)) + cn$$
$$T(n) = 4T\left(\frac{n}{9}\right) + c(\left(\frac{2n}{3}\right) + n)$$
$$T(n) = 4(2T\left(\frac{n}{27}\right) + c\left(\frac{n}{9}\right)) + c(\left(\frac{2n}{3}\right) + n)$$
$$T(n) = 8T\left(\frac{n}{27}\right) + c\left(\left(\frac{4n}{9}\right) + \left(\frac{2n}{3}\right) + n\right)$$
$$T(n) = 8T\left(\frac{n}{27}\right) + cn\left(\left(\frac{4}{9}\right) + \left(\frac{2}{3}\right) + 1\right)$$

Now that we've done three iterations, we can guess the form of the recurrence after k iterations:

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

We want to plug in a value of k to this formula such that $\frac{n}{3^k} = 1$, which occurs when $n = 3^k$. By definition of log, we have that $k = log_3 n$. We will bound the summation by taking it to infinity instead of k-1:

$$T(n) \leq 2^{log_3 n} T(1) + cn(\sum_{i=0}^{\infty}(\frac{2}{3})^i)$$

Now, we can use both given hints to arrive at:

$$T(n) \leq n^{log_3 2} + 3cn = O(n)$$

Note that $\log_3 3 = 1$, so if follows that $\log_3 2 < 1$. Thus, the dominant term is 3cn, which is O(n).

2. With proof, determine the Big-Oh run time of the function, f, below, in terms of the input parameter n. (Note: You may use results from algorithms studied previously in COP 3502 without restating the full proof of run time.)

```
int f(int array[], int n) {
    return frec(array, 0, n-1);
}
int frec(int array[], int lo, int hi) {
    if (lo == hi) return array[lo];
    int left = frec(array, lo, (lo+hi)/2);
    int right = frec(array, (lo+hi)/2+1, hi);
    int i, lCnt = 0, rCnt = 0;
    for (i=lo; i<=hi; i++) {
        if (abs(array[i]-left) < abs(array[i]-right))
            lCnt++;

else rCnt++;

    }
    if (lCnt > rCnt) return lCnt;
    return rCnt;

}
```

ANS: Solution: The function f is a wrapper function that calls the recursive function frec. f takes in an array of size n while frec takes in a subsection of an array of size hi-lo+1. Let T(n) be the run time of the function frec where hi-lo+1 = n.

To determine what T(n) equals, first note that two recursive calls are made, each to arrays of size n/2. Each of these recursive calls, by definition, takes T(n/2) time. This is follows by a for loop that runs n times, inside of which there are only a few O(1) operations. Thus, we add O(n) to the runtime of the function for the second portion of the code. Thus, our total tally is:

T(n) = T(n/2) + T(n/2) + O(n) T(n) = 2T(n/2) + O(n)

If one recognizes this as the recurrence of Merge Sort solved in COP 3502, one can state that the result of solving this recurrence relation is **T(n) = O(nlgn).** Alternatively, either the Master Theorem or Iteration Technique can be used to arrive at the final solution for the recurrence.

3. Use the iteration technique to solve the following recurrence relation in terms of n: $(n) = 3T(n - 1) + 1$, $for\ all\ integers\ n > 1\ T(1) = 1$

Please give an exact closed-form answer in terms of n, instead of a Big-Oh answer. (Note: A useful summation formula to solve this question is $\sum_{i=1}^{n}(x^i) = \frac{x^{n+1}-1}{x-1}$

$(n) = 3T(n-1) + 1 = 3(3T(n-2) + 1) + 1 = 9T(n-2) + 3 + 1 = 9(3T(n-3) + 1) + 3 + 1 = 27T(n-3) + 9 + 3 + 1$

After k steps, we have:

$= 3^k(n-k) + \sum_{i=0}^{k-1}(3^i)$

Let k = n−1, then we have that

$(n) = 3^{n-1}T(n-(n-1)) + \sum_{i=0}^{n-2}(3^i)$

$= 3^{n-1}(1) + \sum_{i=0}^{n-2}(3^i)$

$= 3^{n-1} + \sum_{i=0}^{n-2}(3^i)$

$= \sum_{i=0}^{n-1}(3^i)$

$= (3^n-1)/(3-1) = 3^n-1/2$

4. Using the iteration technique, find a tight Big-Oh bound for the recurrence relation defined below:

$$(n) = 3T(n/2) + n^2, \text{ for } n > 1\ T(1) = 1$$

Hint: You may use the fact that $\sum_{i=0}^{\infty}(3/4) = 4$ and that $3^{log_2 n} = n^{log_2 3}$, and that $log_2 3 < 2$.

ANS: Iterate the given recurrence two more times:

$$T(n) = 3T\left(\frac{n}{2}\right) + n^2$$

$$T(n) = 3(3T\left(\frac{n}{4}\right) + (\frac{n}{2})^2) + n^2$$

$$T(n) = 9T\left(\frac{n}{4}\right) + \frac{3n^2}{4} + n^2$$

$$T(n) = 9T\left(\frac{n}{4}\right) + n^2(1 + \frac{3}{4})$$

$$T(n) = 9(3T\left(\frac{n}{8}\right) + (\frac{n}{4})^2) + n^2(1 + \frac{3}{4})$$

$$T(n) = 27T\left(\frac{n}{8}\right) + \frac{9n^2}{16} + n^2(1 + \frac{3}{4})$$

$$T(n) = 27T\left(\frac{n}{8}\right) + n^2(1 + \frac{3}{4} + \frac{9}{16})$$

In general, after the k^{th} iteration, we get the recurrence

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

To solve the recurrence, find k such that $n/2^k = 1$. This occurs when $n = 2^k$ and $k = log\ n$. Plug into the equation above for this value of k to get:

$$T(n) = 3^{logn}T(1) + n^2\left(\sum_{i=0}^{k-1}\left(\frac{3}{4}\right)^i\right) \leq 3^{logn} + n^2\left(\sum_{i=0}^{\infty}\left(\frac{3}{4}\right)^i\right) = n^{log_2 3} + 4n^2 = O(n^2)$$

5. Find the Big-Oh solution to the following recurrence relation using the iteration technique. Please show all of your work, including 3 iterations, followed by guessing the general form of an iteration and completing the solution. Full credit will only be given if all of the work is accurate (and not just for arriving at the correct answer.)

$$T(\ ) = 2\,T\,(n/2\,) + n^{\,2}\,,\,T\,(\,1\,) = 1$$

Answer:

First, iterate three times:

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + n^2 \\
&= 2\left[2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2\right] + n^2 \\
&= 2\left[2T\left(\frac{n}{4}\right) + \frac{n^2}{4}\right] + n^2 \\
&= 4T\left(\frac{n}{4}\right) + \frac{n^2}{2} + n^2 \\
&= 4T\left(\frac{n}{4}\right) + \frac{3n^2}{2} \\
&= 4[2T\left(\frac{n}{8}\right) + (\frac{n}{4})^2] + \frac{3n^2}{2} \\
&= 4[2T\left(\frac{n}{8}\right) + \frac{n^2}{16}] + \frac{3n^2}{2} \\
&= 8T\left(\frac{n}{8}\right) + \frac{n^2}{4} + \frac{3n^2}{2} \\
&= 8T\left(\frac{n}{8}\right) + \frac{7n^2}{4}
\end{aligned}$$

In general, after k iterations we will have $T(n) = 2^k T\left(\frac{n}{2^k}\right) + \frac{(2^k-1)n^2}{2^{k-1}}$. We want to plug in the value of k for which $\frac{n}{2^k} = 1$,which is when $n = 2^k$. Note that for this value of k, $2^{k-1} = n/2$, since $2 \times 2^{k-1} = 2^k$:

$$T(n) = nT(1) + \frac{(n-1)n^2}{\frac{n}{2}} = n(1) + 2n(n-1) = 2n^2 - n = O(n^2)$$

6. Find the Big-Oh solution to the following recurrence relation using the iteration technique. Please show all of your work, including 3 iterations, followed by guessing the general form of an iteration and completing the solution. Full credit will only be given if all of the work is accurate (and not just for arriving at the correct answer.)

$$T(\ ) = 4\,T(\,n/2\,) + n\,,\ T(\,1\,) = 1$$

Answer::

First, iterate three times:

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + n \\ &= 4\left[4T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n \\ &= 16T\left(\frac{n}{4}\right) + 2n + n \\ &= 16T\left(\frac{n}{4}\right) + 3n \\ &= 16\left[4T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 3n \\ &= 64T\left(\frac{n}{8}\right) + 4n + 3n \\ &= 64T\left(\frac{n}{8}\right) + 7n \end{aligned}$$

In general, after k iterations, we will have $T(n) = 4^k T\left(\frac{n}{2^k}\right) + (2^k - 1)n$. We want to plug in the value of k for which $\frac{n}{2^k} = 1$,which is when $n = 2^k$. In this case, note that $n^2 = (2^k)^2 = 2^{2k} = 4^k$. Plugging in, we find:

$$\begin{aligned} T(n) &= n^2 T(1) + (n-1)n \\ &= n^2 + n^2 - n \\ &= 2n^2 - n \\ &= O(n^2) \end{aligned}$$