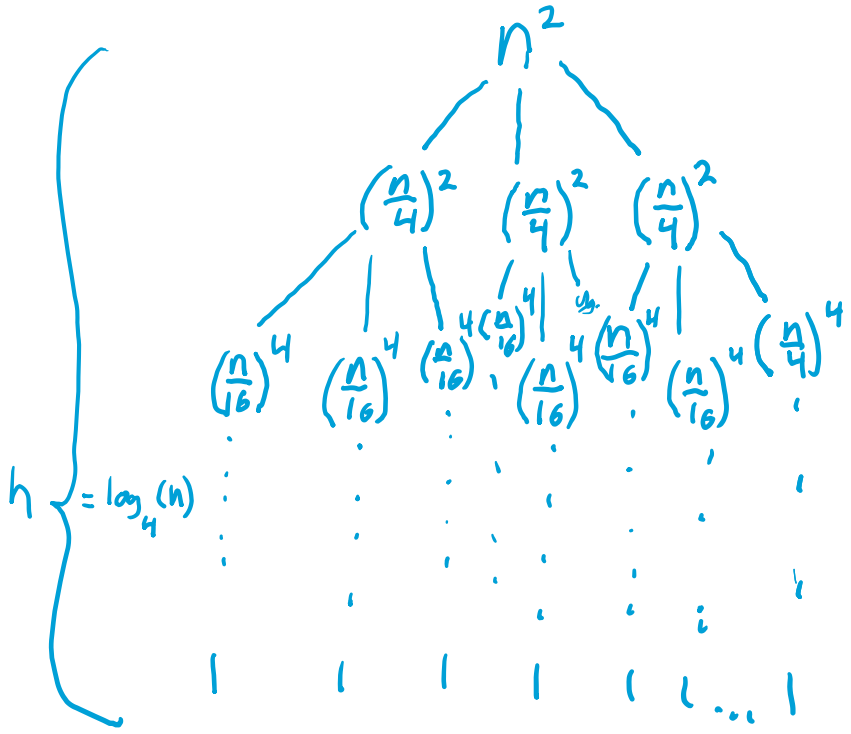
Riley Campbell
CSC 372
Project 1b

1) [26 pt] Determine the run time (big-O) for the following recurrence formula using the tree or substitution method. You may use the master method only to check your answer.

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T(n/4) + n^2 & n > 1 \end{cases}$$

master check
$a = 3$
$b = 4$
$k = 2$
$= O(n^k) = O(n^2)$

Level	# nodes	Cost
0	1 = 3^0	n^2
1	3 = 3^1	3(n/4)^2
2	9 = 3^2	9(n/16)^2
i	3^i	3^i(n/4^i)^2



$$\sum_{i=0}^{\log_3(n)} a^i \cdot f(n/b^i)$$

$$\sum_{i=0}^{\log_3 n} 3^i (n/4^i)^2$$

$$\sum_{i=0}^{\log_3 n} n^2 3^i \left(\frac{1}{4}i\right)^2$$

$$n^2 \sum_{i=0}^{\log_3 n} 3^i \left(\frac{1}{4}i\right)^2$$

$$= n^2$$

$O(n^2)$

2) [26 pt] Determine the run time (big-O) for the following recurrence formula using the tree or substitution method. You may use the master method only to check your answer.

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\frac{n}{3}\right) + n^3 & n > 1 \end{cases}$$

master check
$a = 2$
$b = 3$
$k = 3$
$O(n^k) = O(n^3)$

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

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

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

$= 2^3\left(2T\left(\frac{n}{81}\right) + \left(\frac{n}{27}\right)^3\right) + 2^2\left(\frac{n}{9}\right)^3 + 2^1\left(\frac{n}{3}\right)^3 + n^3 = 2^4 T\left(\frac{n}{81}\right) + 2^3\left(\frac{n}{27}\right)^3 + 2^2\left(\frac{n}{9}\right)^3 + 2^1\left(\frac{n}{3}\right)^3$

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

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

let $k = h = \log_3 n$

$= 2^k T\left(n/3^k\right) + n^3 \Theta(1)$

$= 2^k T(1) + n^3$

$= 2^{\log_3 n} + n^3$

$= n^{\log_3 2} + n^3$

$= n^3$

$O(n^3)$

3) (12pt) Determine which case of the Master Theorem applies for the following recurrences Include the values of a, b, and k (and ideally b k) as proof of your selection. Also, include the final big-theta formula. You also have the option of a recurrence relation that cannot use the master method as described in class, in which case, state it "fails."

a. $T(n) = 2T\left(\frac{n}{2}\right) + \sqrt{n}$

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

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

d. $T(n) = 16T\left(\frac{n}{2}\right) + n^4$

Question #	A	B	K	B^k	Master Theorem case	Big Θ
A	2	2	½	$\sqrt{2}$	If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$	$\Theta(n^{\log_2 2}) = \Theta(n)$
B	3	4	2	16	If $a < b^k$, then $T(n) = \Theta(n^k)$	$\Theta(n^2)$
C	2	n/a	3	n/a	fails	fails
D	16	2	4	16	If $a = b_k$, then $T(n) = \Theta(n^k \log_b n)$	$\Theta(n^4 \log_2 n)$

4) (10 pt) Prove the correctness of the outer loop of the following 2D array summation using the loop invariant technique.

```
sum_array(A[][])
    sum = 0
    for each row R
        for each element j in R
            sum = sum + j
    return sum
```

1. Prove the base case
   For an array A of size 1x1, sum will be equal to A[0][0] since sum is initially equal to 0.
2. What is the loop invariant?
   At the end of each of the outer loops, the current sum will be equal to summed values in each row up to the current row via sum = sum + j in the inner loop.
3. Prove the final step
   At the end, we will have the sum from A[0][0] to A[m][n] because we are adding every element in every row in a running tally

5) (8 pt) For the following problem, what is the best sort among the given list. This is a real-world problem, so some ambiguity exists. Therefore, you must also list any assumptions, and explain why you removed the other sort options. Your options are insertion, selection, bubble, merge, quick, and heap. Consider stability, in-place, data structure, etc. in your answer

Problem: You want to sort the number of steps taken per day in the last 2 years on a smart watch.

Assumptions

- The data is stored on the smart watch
- The sorting will take place on the smart watch
- The person wearing it is training for long distance hiking

I would use Insertion Sort. My reasons are the low memory cost on the already low memory of the watch, the fact that since the user is training so the list should already be mostly sorted, and the speed on the sort. I wouldn't use the other sorts for their memory overhead, runtime, complex coding/debugging, and personal preference.

6) (18 pt) Write the resulting recurrence relation (the T(n) piece-wise function) for the following pseudocode where A is an arrays of integers:

```
FUNC(A, s, e)
    if s >= e
        print s, e, and all of A[1..n]
        return
    cut1 = (e - s) / 2
    cut2 = (e - s) / 4
    FUNC(A, s, s + cut1)
    FUNC(A,  s + cut1 + 1, s + cut1 + cut2)
    FUNC(A, s + cut1 + cut2 + 1, e)
```

$$T(n) = T(n/2) + 2T(n/4) + n$$
$$T(1) = 1$$
$$T(0) = 1$$