1. This question will give you further practice with the Master Method. Suppose the running time of an algorithm is governed by the recurrence $T(n) = 7 * T(n/3) + n^2$. What's the overall asymptotic running time (i.e., the value of T(n))?

With the Master Method, a = 7, b = 3, d = 2. a < b^d so this is case 2. Root work dominates the run time → $\theta(n^d)$

$\theta(n^2)$

2. This question will give you further practice with the Master Method. Suppose the running time of an algorithm is governed by the recurrence $T(n) = 9 * T(n/3) + n^2$. What's the overall asymptotic running time (i.e., the value of T(n))?

With the Master Method, a = 9, b = 3, d = 2. a = b^d so this is case 1. Work at every level is the same. → $\theta(n^d \, logn)$

$\theta(n^2 \, logn)$

3. This question will give you further practice with the Master Method. Suppose the running time of an algorithm is governed by the recurrence $T(n) = 5 * T(n/3) + 4n$. What's the overall asymptotic running time (i.e., the value of T(n))?

With the Master Method, a = 5, b = 3, d = 1. a > b^d so this is case 3. Work done = number of leaves. → $\theta(n^{log_b a})$

$\theta(n^{log_3 5})$

4. Consider the following pseudocode for calculating a^b(where a and b are positive integers)

```
FastPower(a,b) :
  if b = 1
    return a
  else
    c := a*a
    ans := FastPower(c,[b/2])
  if b is odd
    return a*ans
  else return ans
end
```

Here [x] denotes the floor function, that is, the largest integer less than or equal to x.

Now assuming that you use a calculator that supports multiplication and division (i.e., you can do multiplications and divisions in constant time), what would be the overall asymptotic running time of the above algorithm (as a function of b)?

The function work (besides recursion) is constant, so a = 1. Each time the input is half of the previous input, so b = 2. d = 0. a = b^d, so this is case 1. Work at every level is the same. → $\theta(n^d \, logn)$

$\theta(log b)$ (input size is b)

5. Choose the smallest correct upper bound on the solution to the following recurrence: $T(1) = 1$ and $T(n) \leq T(\lfloor \sqrt{n} \rfloor) + 1$ for $n > 1$. Here [x] denotes the "floor" function, which rounds down to the nearest integer. (Note that the Master Method does not apply.)

Substitute m = logn to get n = 2^m

$T(2^m) < T(\left\lfloor \sqrt{2^m} \right\rfloor) + 1 \rightarrow S(m) < S(m/2) + 1$

$S(k) = T(2^k) \leq S(m/2) + 1$

a = 1, b = 2, d = 0, a = b^d, so this is case 1. Work at every level is the same. $\rightarrow \theta(n^d \, logn)$

$\theta(logm)$

$\theta(loglogn)$