Due Date:

In problem 1 to 2, solve the following recurrence with different methods that we learned in class.

$$T(n) = \begin{cases} \Theta(1) & n \le k \\ 6T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{2}\right) + 9^{(\log_3 n)} & n > k \end{cases}$$

where n = 2^j for a positive integer j, and k is a small positive integer. That is, find a function g(n) such that T(n) ∈Θ(g(n)). The Θ(1) terminating condition is intended to represent some small constant.

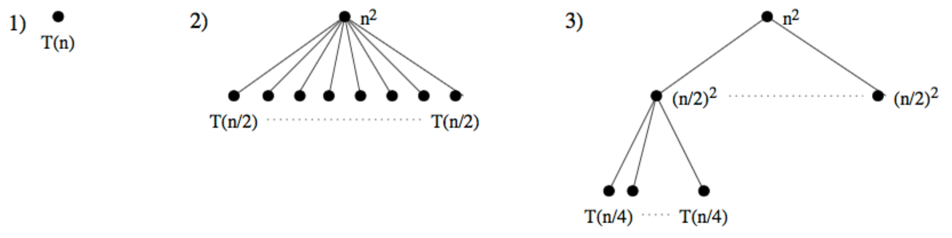
1. (10 points) Use the **Master Theorem** to solve this recurrence.

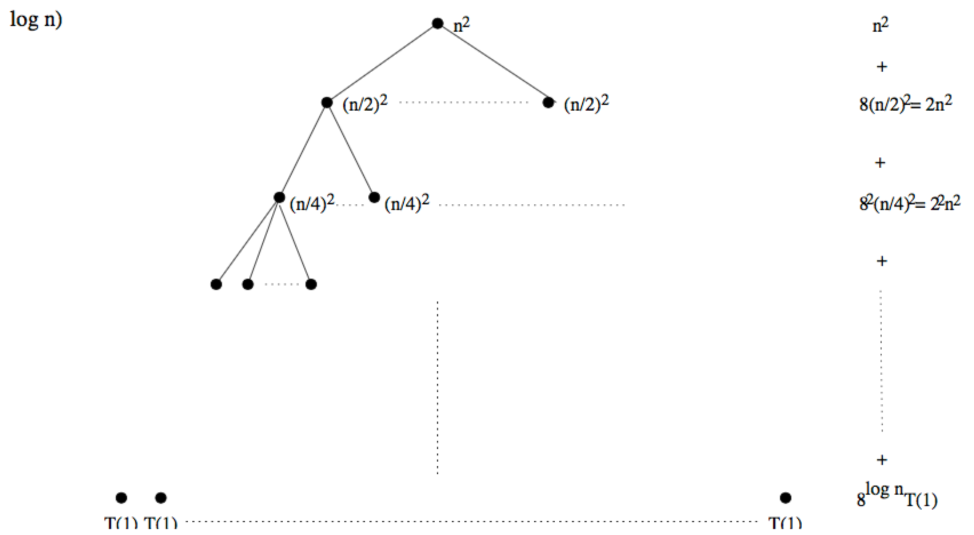
   $9^{(\log_3 n)} = n^{(\log_3 9)} = n^2$

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

   a = 8, b = 2, , compare n^3 vs. n^2, case 1 of Master Method, T(n) = $\Theta(n^3)$

2. (20 points) Use the **recursion tree** to solve the recurrence.





- Recursion depth: How long (how many iterations) it takes until the subproblem has constant size? i times where $\frac{n}{2^i} = 1 \Rightarrow i = \log n$
- What is the last term? $8^i T(1) = 8^{\log n}$

$$\begin{aligned} T(n) &= n^2 + 2n^2 + 2^2n^2 + 2^3n^2 + 2^4n^2 + \ldots + 2^{\log n - 1}n^2 + 8^{\log n} \\ &= \sum_{k=0}^{\log n - 1} 2^k n^2 + 8^{\log n} \\ &= n^2 \sum_{k=0}^{\log n - 1} 2^k + (2^3)^{\log n} \end{aligned}$$

- Now $\sum_{k=0}^{\log n-1} 2^k$ is a geometric sum so we have $\sum_{k=0}^{\log n-1} 2^k = \Theta(2^{\log n - 1}) = \Theta(n)$
- $(2^3)^{\log n} = (2^{\log n})^3 = n^3$

$$\begin{aligned} T(n) &= n^2 \cdot \Theta(n) + n^3 \\ &= \Theta(n^3) \end{aligned}$$

3. Resolve Recurrence

(1) (10 points) Find a tight upper and lower bound solution for the following recurrence:

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

That is, find a function g(n) such that T(n) ∊ Θ(g(n)). You may assume that n=3^k for some positive integer k. Justify your answer.

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

a = 4, b = 3, $\log_b a = \log_3 4$, compare $n^{\log_3 4}$ vs. n, case 1 of Master Method,

$T(n) = \Theta(n^{\log_3 4})$

(2) (20 points) prove your answer in (1) using the **substitution method**.

Substitution Proof

$T(n) = 4T(\frac{n}{3}) + n = \Theta(n^{\log_3 4})$

Upper Bound :

$T(n) \leq 4T(\frac{n}{3}) + cn$ for some positive constant $\overset{*}{c}$

Guess $T(n) = O(n^{\log_3 4})$ or $T(n) \leq d \cdot n^{\log_3 4}$ for some positive constant d

Substitution :

$T(n) \leq 4T(\frac{n}{3}) + cn$

$= 4d(\frac{n}{3})^{\log_3 4} + cn$

$= 4d \frac{n^{\log_3 4}}{3^{\log_3 4}} + cn = 4d \frac{n^{\log_3 4}}{4} + cn$

$= d\, n^{\log_3 4} + cn$

$\not\leq d n^{\log_3 4}$ (because cn can't ≤ 0)

$\boxed{\text{Make a new Guess}}$ by subtracting off a lower-order term

$T(n) \leq d n^{\log_3 4} - d'n$, for some positive d and d'

Substitution :

$T(n) \leq 4T(\frac{n}{3}) + cn$

$= 4\left(d(\frac{n}{3})^{\log_3 4} - d'\frac{n}{3}\right) + cn$

$= dn^{\log_3 4} - \frac{4}{3}d'n + cn$

$= dn^{\log_3 4} - d'n - \frac{1}{3}d'n + cn$

$\leq dn^{\log_3 4} - d'n$ if $(-\frac{1}{3}d'n + cn) \leq 0$

$d' \geq 3c$

Therefore, $T(n) = O(n^{\log_3 4})$

(* It is important have c, d, d')

Lower Bound :

$T(n) \geq 4T(\frac{n}{3}) + cn$, for some positive constant c

Guess $T(n) = \Omega(n^{\log_3 4})$ or

$T(n) \geq d \cdot n^{\log_3 4}$ for some positive constant d

Substitution :

$T(n) \geq 4T(\frac{n}{3}) + cn$

$= 4(d(\frac{n}{3})^{\log_3 4}) + cn$

$= d \cdot n^{\log_3 4} + cn$

$\geq dn^{\log_3 4}$ (for any n)

Therefore , $T(n) = \Omega(n^{\log_3 4})$

$\therefore$ We can say that $T(n) = \Theta(n^{\log_3 4})$

4. (10 points) For each of the following recurrences, give an expression for the runtime T(n) if the recurrence can be solved with the Master Theorem. Otherwise, explain why the Master Theorem does not apply. Justify your answer.

(1) $T(n) = 3^n T(\frac{n}{3}) + n^3$

$a = 3^n$ is not a constant, so Mater Theorem doesn't apply.

(2) $T(n) = 5T(\frac{n}{2}) + \sqrt{10}n^3$

$a = 5$, $b = 2$, $\log_b a = \log_2 5$, $f(n) = \sqrt{10}n^3$
so compare $n^{\log_2 5}$ vs. n^3, so case 3, $T(n) = \Theta(n^3)$

(3) $T(n) = \frac{1}{4} T(\frac{n}{4}) + n \lg n$

$a = \frac{1}{4}$, less than 1, so Mater Theorem doesn't apply

(4) $T(n) = T(n-1) + 2n$

not in the form of $aT(n/b) + f(n)$, (where a and b are constants > 1), so Mater Theorem doesn't apply

(5) $T(n) = 16T(\frac{n}{4}) + n^2$

$a = 16$, $b = 4$, $\log_b a = \log_4 16$, $f(n) = n^2$
compare n^2 vs. n^2, so case 2, $T(n) = \Theta(n^2 \lg n)$

5. **Design an Algorithm** (30 points)

You are given a sorted array *A*, which stores *n* integers, and a value *key*. Design an efficient **divide-and-conquer** algorithm that returns the index of the value *key* if it can be found in array *A*. *Otherwise, the algorithm returns 0*.

(1) (10 points) Pseudocode *(please use the textbook conventions)*

```
  DC_Search(A, n, key)
1         Return BinarySearch (A, 1, n, key)
```

```
  BinarySearch (A, low, high, key)
1         if (low > high)
2                 return 0
3         else
4                 mid = ⌊(low + high) /2⌋
5         if (key == A[mid])
6                 return mid
7         else  if (key < A[mid])
8                 return BinarySearch(A, low, mid-1, key)
9         else return BinarySearch(A, mid+1, high, key)
```

(2) (10 points) Analysis: Provide a tight lower and upper bound on the worst-case asymptotic running time of your pseudocode (provide the recurrence and resolve it). Justify your answer (i.e., list the cost for executing each line of code and how you calculate the total running time).

Line 1~2: base case, $\Theta(1)$

or Line 4: Divide, $\Theta(1)$

Line 5~6 or Line 7~8, or Line 9

Line 5~6: $\Theta(1)$

Line 8 or Line 9: Conquer: $T(n/2)$

Worst-case:

$T(n) = T(n/2) + \Theta(1) = \Theta(\lg n)$ by using the Mater Method, case 2

Students should justify how they calculate the running time

(3) (10 points) Prove your answer using the **substitution method**

$T(n) = T(\frac{n}{2}) + \Theta(1)$

Upper Bound:

$T(n) \le T(\frac{n}{2}) + c$, for some positive constant c

Guess: $T(n) \le d\lg n$ for some positive constant d

Substitution:

$$\begin{aligned} T(n) &\le T(\tfrac{n}{2}) + c \\ &= d\lg\tfrac{n}{2} + c \\ &= d\lg n - d\lg 2 + c = d\lg n - d + c \\ &\le d\lg n \qquad \text{if } (-d+c) \le 0 \\ & \qquad\qquad\quad d \ge c \end{aligned}$$

Therefore, $T(n) = O(\lg n)$

Lower Bound:

$T(n) \ge T(\frac{n}{2}) + c$, for some positive constant c

Guess: $T(n) \ge d\lg n$ for some positive constant d

Substitution:

$T(n) \quad \ge T(\frac{n}{2}) + c$

$= d \lg \frac{n}{2} + c$

$= d \lg n - d + c$

$\geq d \lg n \quad \text{if } (-d+c) \geq 0$

$\quad d \leq c$

Therefore, $T(n) = \Omega(\lg n)$

We can conclude that, $T(n) = \Theta(\lg n)$