

## CSE 332 - Section 2 Worksheet

### 0. Recurrence Relations

a) Find a recurrence  $T(n)$  modeling the worst-case runtime complexity of  $f(n)$

```
1 f(n) {
2   if (n <= 0) {
3     return 1
4   }
5   return 2 * f(n - 1) + 1
6 }
```

$$T(n) = \begin{cases} c_0 & \text{if } n \leq 0 \\ T(n-1) + c_1 & \text{otherwise} \end{cases}$$

b) Find a recurrence  $T(n)$  modeling the worst-case runtime complexity of  $g(n)$

```
1 g(n) {
2   if (n <= 1) {
3     return 1000
4   }
5   if (g(n/3) > 5) {
6     for (int i = 0; i < n; i++) {
7       println("Yay")
8     }
9     return 5 * g(n/3)
10  } else {
11    for (int i = 0; i < n * n; i++) {
12      println("Yay")
13    }
14    return 4 * g(n/3)
15  }
```

$$T(n) = \begin{cases} c_0 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{3}\right) + c_1n + c_2 & \text{otherwise} \end{cases}$$

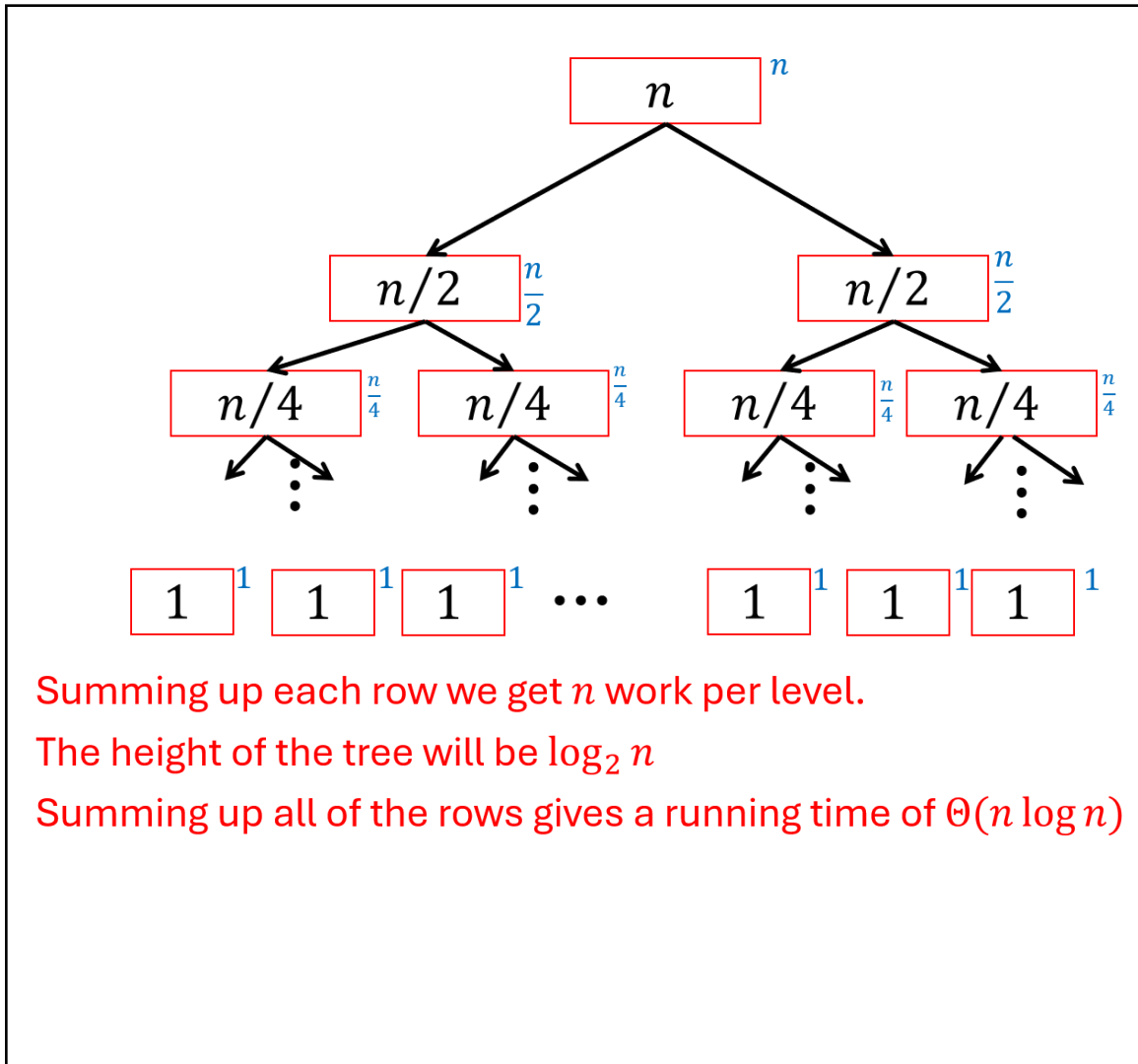
## CSE 332 - Section 2 Worksheet

### 1. Tree Method

For each of the following recurrence relations, use the tree method to convert it to closed form:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n & \text{otherwise} \end{cases}$$

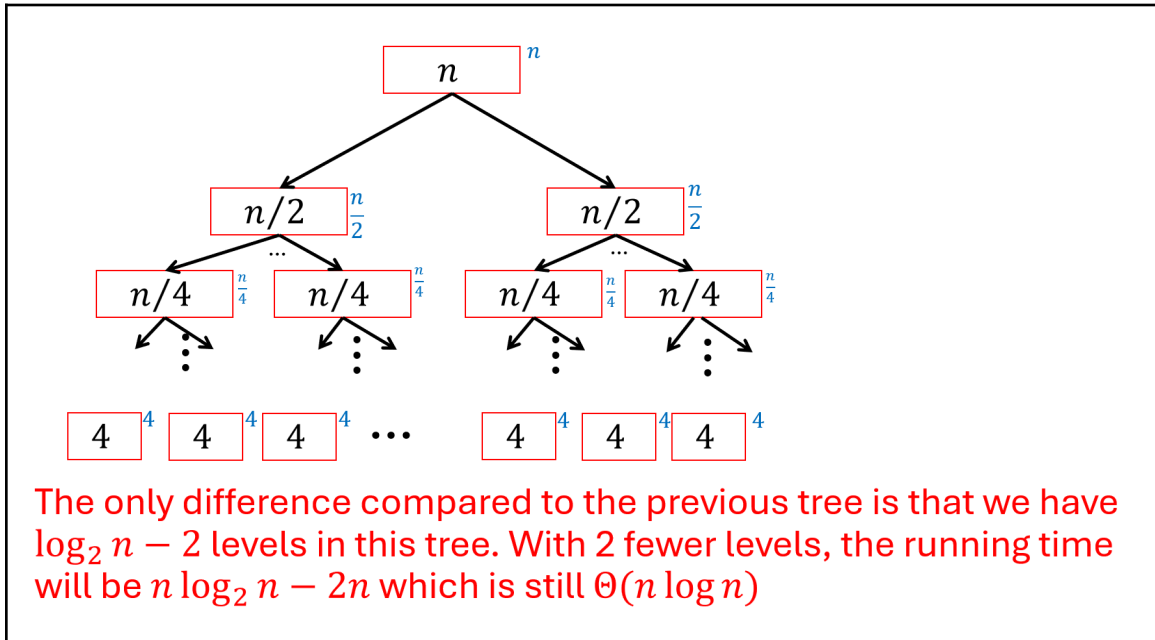
a)



## CSE 332 - Section 2 Worksheet

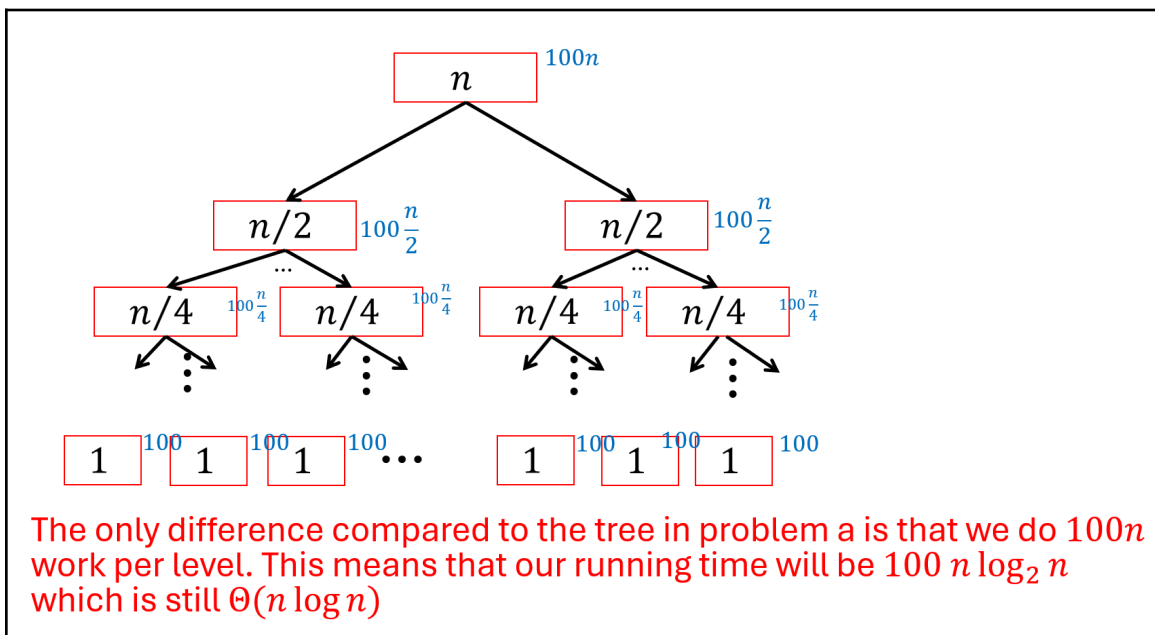
$$T(n) = \begin{cases} 4 & \text{if } n \leq 4 \\ 2T\left(\frac{n}{2}\right) + n & \text{otherwise} \end{cases}$$

b)



$$T(n) = \begin{cases} 100 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + 100n & \text{otherwise} \end{cases}$$

c)



## CSE 332 - Section 2 Worksheet

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 3T\left(\frac{n}{2}\right) + n & \text{otherwise} \end{cases}$$

d)

At each level the number of nodes in the tree will be triple the number of nodes at the previous level. If we consider the root to be at level 0 then the number of nodes on level  $i$  is  $3^i$ . Additionally, the size of each subproblem will be half that of its parent, and so the size of each node on level  $i$  is  $\frac{n}{2^i}$ , which means there is  $\frac{n}{2^i}$  non-recursive work.

The total work done on level  $i$  is therefore  $\left(\frac{3}{2}\right)^i n$ . Because there are  $\log_2 n$  levels the solution is given by the sum

$$n \sum_{i=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^i$$

Applying the geometric series formula we get

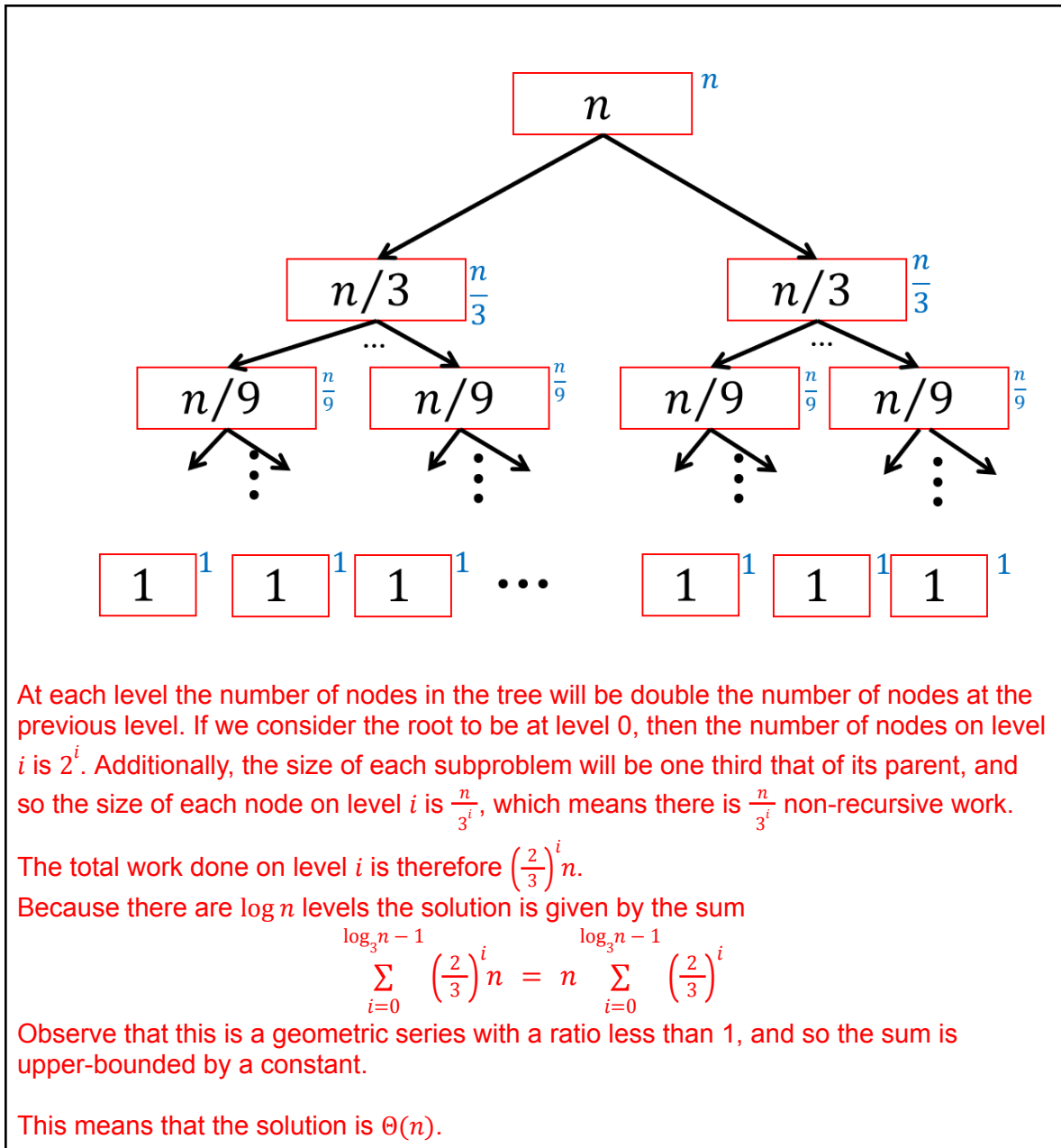
$$n \left( \frac{1 - \left(\frac{3}{2}\right)^{\log_2 n}}{1 - \frac{3}{2}} \right) = 2n \left( \frac{3^{\log_2 n}}{2^{\log_2 n}} - 1 \right) = 2n \left( \frac{n^{\log_2 3}}{n} - 1 \right) = 2n^{\log_2 3} - 2n$$

This means that the solution is  $\Theta(n^{\log_2 3})$ .

## CSE 332 - Section 2 Worksheet

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T\left(\frac{n}{3}\right) + n & \text{otherwise} \end{cases}$$

e)



## 2. Putting It All Together

Consider the function  $f(n)$ . Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

## CSE 332 - Section 2 Worksheet

```
1 f(n) {
2     if (n <= 1) {
3         return 0
4     }
5     int result = f(n/2)
6     for (int i = 0; i < n; i++) {
7         result *= 4
8     }
9     return result + f(n/2)
10 }
```

a) Find a recurrence  $T(n)$  modeling the *worst-case runtime complexity* of  $f(n)$

We look at the three separate components (base case, non-recursive work, recursive work). The base case is a constant amount of work, because we only do a return statement. We'll label it  $c_0$ . The non-recursive work is a constant amount of work (we'll call it  $c_1$ ) for the assignments and `if` tests and a constant (we'll call  $c_2$ ) multiple of  $n$  for the loops. The recursive work is  $2T\left(\frac{n}{2}\right)$ .

Putting these together, we get:

$$T(n) = c_0, \text{ if } 1$$
$$T(n) = 2T\left(\frac{n}{2}\right) + c_2n + c_1, \text{ otherwise}$$

b) Use your answer in part (a) to find a closed form for  $T(n)$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_2(n)-1} 2^i \left( c_2 \left( \frac{n}{2^i} \right) + c_1 \right) \\ &= \sum_{i=0}^{\log_2(n)-1} (c_2n + 2^i \cdot c_1) \\ &= c_2n \log_2(n) + c_1 \sum_{i=0}^{\log_2(n)-1} 2^i \\ &= c_2n \log_2(n) + c_1 \frac{1 - 2^{\log_2(n)}}{1 - 2} \\ &= c_2n \log_2(n) + c_1 (n - 1) \\ &\in \Theta(n \log n) \end{aligned}$$