Name Caitlin Walker

CSC 325, Assignment #2 – Master Theorem

```
def           (a):
    n = len(a)
    if n <= 1:
        return a

    mid1 = 1 * n // 3
    mid2 = 2 * n // 3

    left   = mergesort3(a[    0:mid1 ])
    middle = mergesort3(a[ mid1:mid2 ])
    right  = mergesort3(a[ mid2:n    ])

    # merge3: T(m) <= 12n + 8
    return merge3(left, middle, right)
```

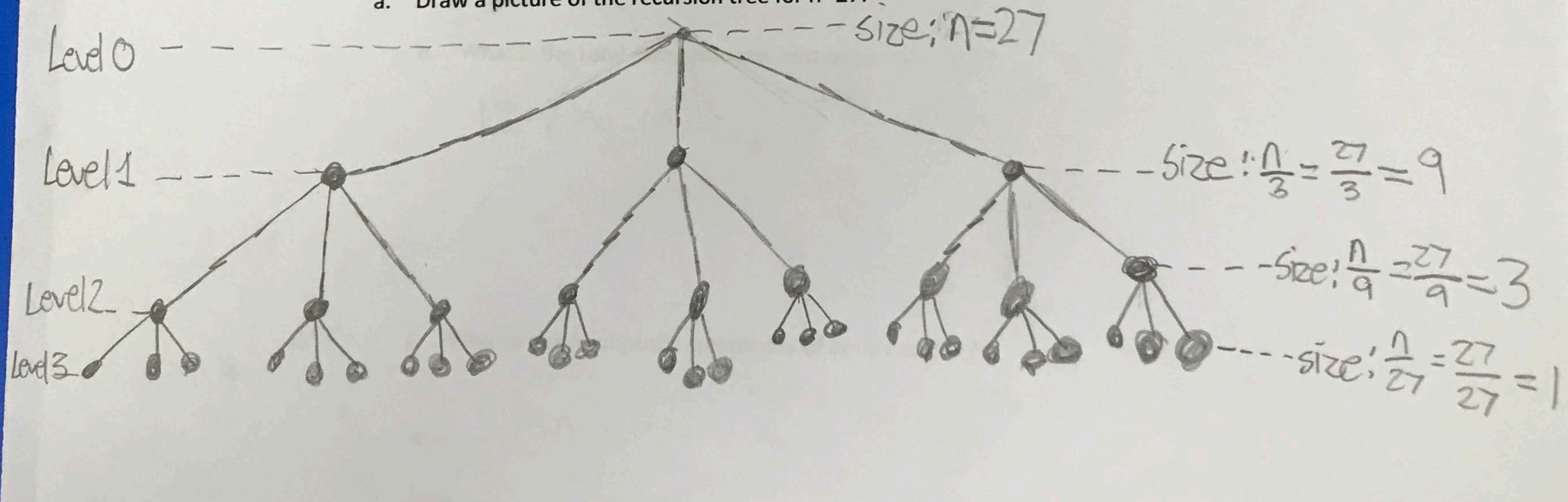
1. The easy way: use the master theorem to get the asymptotic running time of . (What are the values of , , , and what case does that lead to?)

$a = 3$
$b = 3$
$d = 1$

$a = b^d$
$3 = 3^1$

asymptotic Rt = $O(n^1 \lg n)$
case 1

2. The hard way (this question continues on the next page):
   a. Draw a picture of the recursion tree for n=27.



Name _____

_____

_____

b. How many sub-problems are there at any given level (use L in your equation)?
Note1: root is **Level 0**, the second level is **Level 1**, and leaves are at **Level $\log_2(n)$**.

3^L

1

3

Levels	Sub P's
0	1
1	3
2	9
3	27
4	

c. How many elements are there for a given sub-problem found in level _?

$\frac{n}{3^L}$

n=27

Levels	Elements
0	27
1	$27/3$ or $27/3^1$
2	$\frac{27}{9}$ or $27/3^2$
3	$\frac{27}{27}$ or $27/3^3$

d. How much work is performed at a given level _?

$3^L \cdot 12(\frac{n}{3^L}) + 6 \Rightarrow 12n$

cancel out

Const

e. What is the total computational cost of `mergesort3`?

$12n(\log_2(n)+1)$

f. What is the asymptotic complexity of `mergesort3`?

$n \lg n$