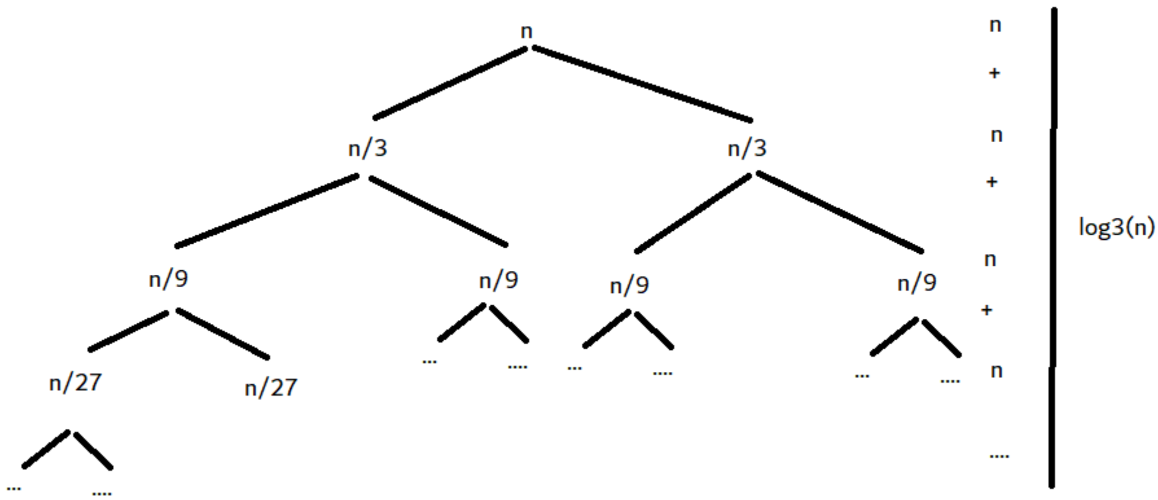
// NOME: Mateus Kenzo Iochimoto TIA: 32216289

2. Dado os algoritmos recursivos abaixo, apresente suas funções de complexidade de tempo.

a)

```
void f(int n)
{
 if (n > 1){
          for (int i = 1; i <= n; i++)
                   {
                   printf("%d\n",n);
                   }
                   f(n/3);
          }
 }
```

T(n) = T(n/3) + n



$T(n) = T(n/3) + n$

$T(n) = [T(n/3) + n/3] + n$

$T(n) = T(n/3^2) + n + n$

$T(n) = [T(n/3^2) + n/3^2] + n + n$

$T(n) = T(n/3^3) + 3n$

⋮

$T(n) = T(n/3^k) + kn$

Quando $n/3^k = 1$, temos $n = 3^k$ e $k = log_3 n$

$T(n) = T(n/3^k) + kn$

$= 3^{log_3 n} \cdot T(1) + n\,log_3 n$

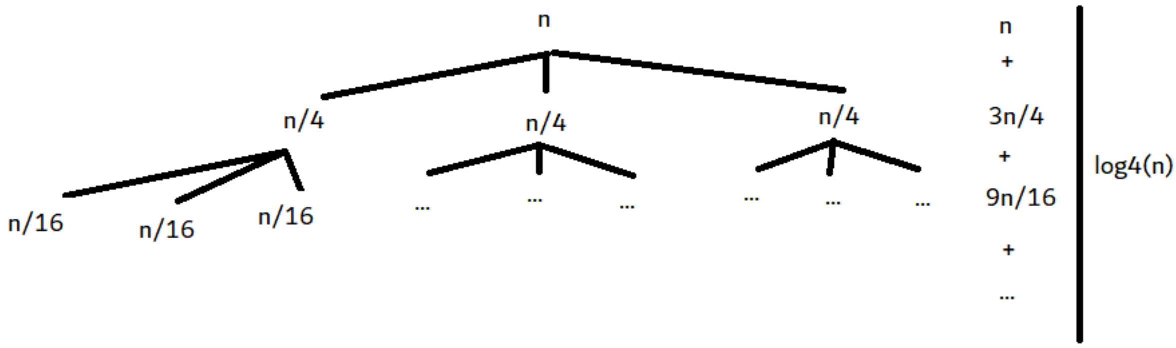
$= n + nlog_3 n$

$\therefore T(n) = O(nlog_3 n)$

b)

```
void f(int n)
{
        if (n > 1){
        for (int i = 1; i <= n; i++)
        {
                printf("%d\n",n);
        }
        f(n/4);
        f(n/4);
        f(n/4);
        }
}
```

T(n) = 3T(n/4) + n



$T(n) = 3T(n/4) + n$

$T(n) = 3[T(n/4) + n/4] + n$

$T(n) = 3^2T(n/4^2) + n + n$

$T(n) = 3^2[T(n/4^2) + n/4^2] + n + n$

$T(n) = 3^3T(n/4^3) + 3n$

$\vdots$

$T(n) = 3^kT(n/4^k) + kn$

Quando $n/4^k = 1$, temos $n = 4^k$ e $k = log_4 n$

$T(n) = 3^kT(n/4^k) + kn$

$= 3^{log_4 n} \cdot T(1) + n\, log_4 n$

$= n + nlog_4 n$

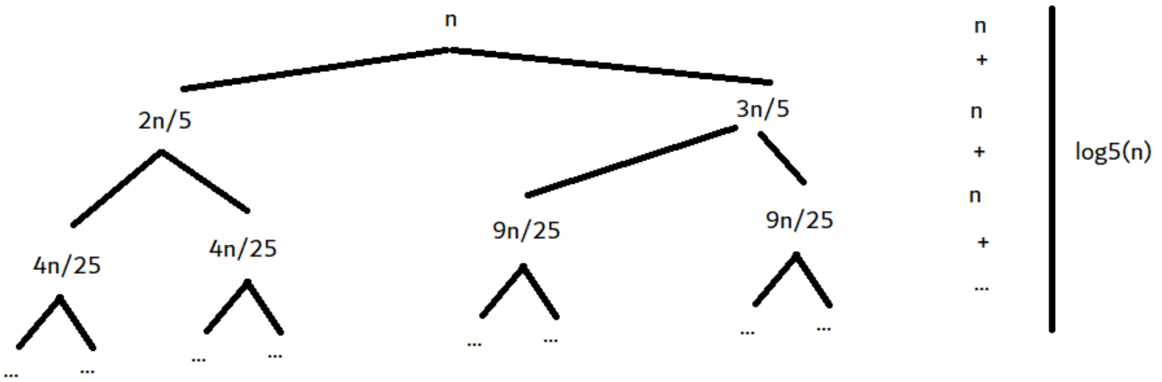
$\therefore T(n) = O(nlog_4 n)$

c)

```
void f(int n)
{
        if (n > 1)
        {
```

```
        for (int i = 1; i <= n; i++)
        {
                printf("%d\n",n);
        }
        f(2*n/5);
        f(3*n/5);
        }
}
```

T(n) = T(2n/5) + T(3n/5) + n



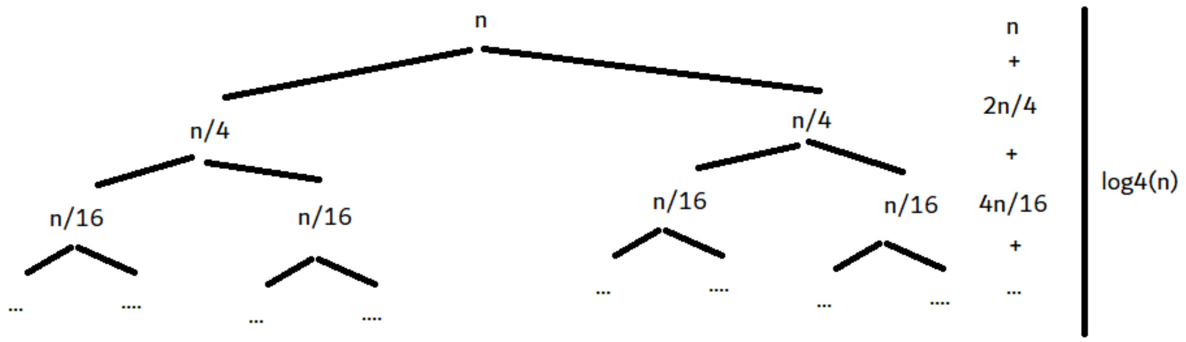
$T(n) = T(2n/5) + T(3n/5) + n$
$T(n) = [T(2n/5) + 2n/5] + [T(3n/5) + 3n/5] + n$
$T(n) = T(2^2 n/5^2) + T(3^2 n/5^2) + n + n$
$T(n) = [T(2^2 n/5^2) + 2^2 n/5^2] + [T(3^2 n/5^2) + 3^2 n/5^2] + n$
$T(n) = T(2^3 n/5^3) + T(3^3 n/5^3) + 3n$
$\vdots$
$T(n) = T(2^k n/5^k) + T(3^k n/5^k) + kn$
Quando $(2n/5)^k = 1$, temos $n = 5^k/2^k$ e $k = log_{5/2} n$
Quando $(3n/5)^k = 1$, temos $n = 5^k/3^k$ e $k = log_{5/3} n$
$T(n) = T(3n/5)^k + T(2/5)^k + kn$
$= 5^{log_{5/2} n} \cdot T(1) + 5^{log_{5/3} n} \cdot T(1) + n\, log_5 n$
$= n + nlog_5 n$
$\therefore T(n) = O(nlog_5 n)$

d)

```
void f(int n)
{
if (n > 1)
        {
        for (int i = 1; i*i <= n; i++)
        {
        printf("%d\n",n);
```

```
        }
        f(n/4);
        f(n/4);
         }
}
```

T(n) = 2(n/4) + n



$T(n) \ = 2T(n/4) \ + \ n$

$T(n) \ = 2\,[T(n/4) \ + n/4] \ + \ n$

$T(n) \ = 2^2T(n/4^2) \ + \ n \ + \ n$

$T(n) \ = 2^3\,[T(n/4^3) \ + n/4^3] \ + \ n + n$

$T(n) \ = 2^3T(n/4^3) \ + \ 3n$

$\vdots$

$T(n) \ = \ 2^kT(n/4^k) \ + \ kn$

Quando $n/4^k = 1$, temos $n = 4^k$ e $k \ = \ log_4n$

$T(n) \ = 2^kT(n/4^k) \ + \ kn$

$= \ 2^{log_4n} \cdot T(1) \ + \ n\,log_4n$

$= \ n \ + \ nlog_4n$

$\therefore T(n) \ = \ O(nlog_4n)$

3. Resolva as recorrências.
(a) T(n) = T(n − 1) + 1
a = 1
caso 2 do teorema mestre
T(n) = O(n * 1) = O(n)

(b) T(n) = T(n − 10) + 1
a = 1
caso 2 do teorema mestre
T(n) = O(n * 1) = O(n)

(c) T(n) = T(n − 1) + n
a = 1
caso 2 do teorema mestre

T(n) = O(n * n) = O(n^2)

(d) T(n) = 2T(n − 1) + 1
a = 2
caso 3 do teorema mestre
T(n) = O(2^n/-1 * 1) = O(2^n)

(e) T(n) = 2T(n − 1) + n
a = 2
caso 3 do teorema mestre
T(n) = O(2^n/-1 * n) = O(n2^n)

(f) T(n) = 2T(n − 1) + n^2
a = 2
caso 3 do teorema mestre
T(n) = O(2^n/1 * n^2) = O(n^2 * 2^n)

(g) T(n) = T(n/2) + 1
a = 1, b = 2, k = 0
caso 2 do teorema mestre
T(n) = 0(n^0 * log2n) = 0(logn)

(h) T(n) = T(n/2) + n
a = 1, b = 2, k = 1
caso 3 do teorema mestre
T(n) - 0(n)

(i) T(n) = T(n/2) + n^2
a = 1, b = 2, k = 2
caso 3 do teorema mestre
T(n) = 0(n^2)

(j) T(n) = 2T(n/2) + 1
a = 2, b = 2, k = 0
caso 1 do teorema mestre
T(n) = 0(n^log2)

(k) T(n) = 2T(n/2) + n
a = 2, b = 2, k = 1
caso 2 do teorema mestre
T(n) = 0(n^1 * logn) = 0(nlogn)

(l) T(n) = 2T(n/2) + n^2
a = 2, b = 2, k = 2
caso 3 do teorema mestre
T(n) = 0(n^2)

(m) T(n) = 4T(n/2) + n

a = 4, b = 2, k = 1
caso 1 do teorema mestre
T(n) = 0(n^log4) = 0(n^2)

(n) T(n) = 3T(n/2) + n
a = 3, b = 2, k = 1
caso 1 do teorema mestre
T(n) = 0(n^log3)

(o) T(n) = 3T(n/4) + n
a = 3, b = 4, k = 1
caso 3 do teorema mestre
T(n) = 0(n)

(p) T(n) = 7T(n/4) + log n
a = 7, b = 4, k = 0
caso 1 do teorema mestre
T(n) = 0(n^log4 7)