1. Considere o algoritmo `ternarySearch`. Indique a complexidade do mesmo, no pior caso, em função de $n = r - l + 1$. Indique a equação de recorrência e resolva-a de duas formas, nomeadamente, usando o método das substituições sucessivas e o teorema mestre.

```
fun ternarySearch( a: IntArray, value: Int, l: Int, r: Int ): Int {
    if (l > r) return -1
    val mid1 = l + (r - l) / 3
    val mid2 = l + 2 * (r - l) / 3
    return if (a[mid1] == value) mid1
    else if (a[mid2] == value) mid2
    else if (value < a[mid1]) ternarySearch(a, value, l, mid1 - 1)
    else if (value > a[mid2]) ternarySearch(a, value, mid2 + 1, r)
    else ternarySearch(a, value, mid1, mid2)
}
```

$$T(n) = T(n/3) + O(1)$$

Método de substituições

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

$= T\left(\frac{n}{9}\right) + 2 =$

$= T\left(\frac{n}{27}\right) + 3 =$

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

$= T(1) + log3(n), \qquad k = log3(n)$

$= log3(n)$

A complexidade temporal é de $O(log3(n))$

Teorema mestre

$a = 1, \qquad b = 3, \qquad f(n) = O(1)$

$f(n) = O\left(n^{log3(1)}\right) = O(1) \rightarrow Caso\ 2$

$f(n) = O\left(n^{log3(1)} \log(n)\right) = O(\log(n))$

A complexidade temporal é de $O(log(n))$