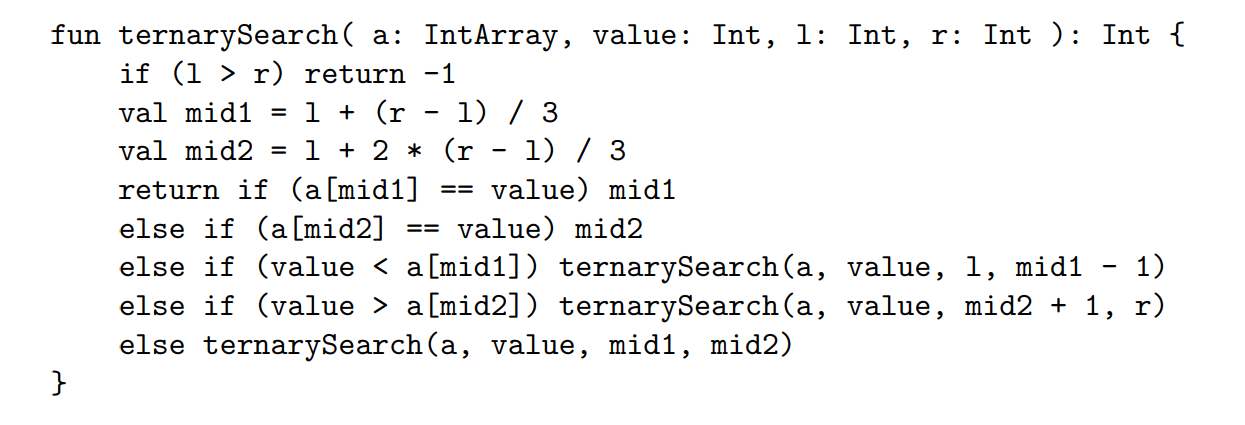
2-Análise de Desempenho:

If (l > r) return –1 = **O(1)**

Val mid1 = l + (r - l)/3 = **O(1)**

Val mid2 = l+2\*(r - l)/3 = **O(1)**

Return if (a[mid1] == value) mid1 = **O(1)**

Else if (value < a[mid1]) ternarySearch(…) = **T(n/3)**

Else if (value > a[mid2]) ternarySearch(...) = **T(n/3)**

Else if(value > a[mid2]) ternarySearch(…) = **T(n/3)**

Else ternarySearch(…) = **T(n/3)**

{T (0) = O (1)

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

Substituição:

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

= [T(n/9) + 1] + 1

= T(n/9) +2

= [T(n/27) + 1] +2

= T(n/27) + 3

Chegamos a conclusão que:

T(n/3k) + k T(n/3log3(n)) + log3(n)

n/3k = 1 T (1) + log3(n)

n = 3k O (1) + O(log3(n)) = O(log(n))

k = log3n

Teorema Mestre:

T(n/3) + 1;

A = 1, B = 3, f(n) = 1;

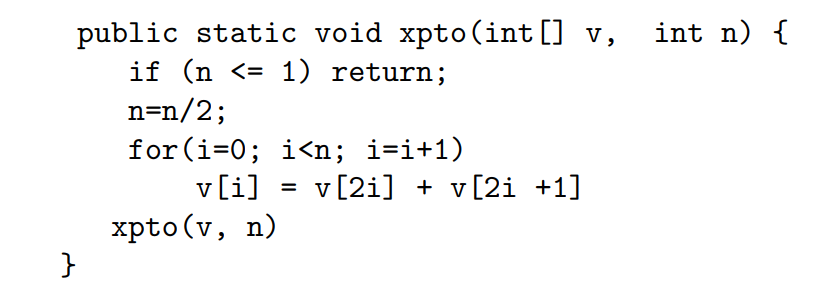
A >= 1, B>1;

nlog3(1) = n0<=> nlog3(1) = 1;

nlog3(1) = f(n);

Logo é o segundo caso do teorema mestre.

O(n0\*log2(n)) = O(log2(n)).



If (n <= 1) return; = O (1)

N = n/2; = O (1)

For (…) = O (n)

Xpto(…) = T (n/2)

{T (0) = T (1) = O (1)

{T (n) = T (n/2) + O (n) + O (1) = T (n/2) + n + 1

[T(n/4) + n/2 + 1] + n + 1

T(n/4) + 3n/2 + 2

[T(n/8) + n/4 + 1] +3n/2 +2

T(n/8) + 7n/4 +3

K + somatório (2^k-1) \*n/2^k-1 + T(n/2^k)

K = log2(n)

log2(n) + 2n-2 + 1

O (log2(n)) + O (2n) +O (-1) +O (1) = O (n)