```
1  if high < low then
2  |  return -1;
3  end
4  mid = ⌊(high + low)/2⌋;
5  if key > A[mid] then
6  |  return BinarySearch(A, mid + 1, high, key);
7  end
8  else
9  |  if key < A[mid] then
10 |  |  return BinarySearch(A, low, mid − 1, key);
11 |  end
12 |  else
13 |  |  return mid;
14 |  end
15 end
```

Algorithm 1: BinarySearch($A[1..n]$, low, $high$, key)

Análise do pior caso:

O pior caso ocorre quando a chave key não ocorre no vetor $A[1..n]$. Note que cada chamada recursiva (linhas 6 e 10) é feita sobre um subvetor que tem a metade do tamanho do vetor atual. Podemos então caracterizar esta situação por meio da seguinte recorrência:

$$T_w(n) = T_w(n/2) + \Theta(1)$$

o custo constante se justifica pelas comparações realizadas nas linhas 1, 5 e 9.

Podemos resolver esta recorrência pelo teorema mestre:

$a=1, b=2$ e $d=0 \Rightarrow a = 1 = b^d$ temos $T_w(n) = \Theta(n^0 . \lg n) = \Theta(\lg n)$.

Alternativamente, podemos resolver esta recorrência pelo método da substituição. Sem perda de generalidade vamos assumir que n é potência de 2, ou seja, $n = 2^k$. Vamos então resolver a seguinte recorrência:

$$T_w(2^k) = T_w(2^{k-1}) + c \text{ ; onde } c = \Theta(1).$$

Aplicando a definição, temos

$$\begin{aligned} T_w(2^k) &= \underline{T_w(2^{k-1})} + c \quad (*) \\ &= (T_w(2^{k-2}) + c) + c \\ &= \underline{T_w(2^{k-2})} + 2.c \\ &= (T_w(2^{k-3}) + c) + 2c \\ &= \cdots = T_w(2^{k-k}) + k.c, \end{aligned}$$

e digamos que $T_w(1) = 0$.

Logo $T_w(2^k) = k.c$ e para verificarmos que $k.c$ é, de fato, solução da recorrência original (*) utilizamos indução em k:

- Base da indução ($k=0$): $T_w(2^0) = 0.c = 0$ ✓
- Passo indutivo ($k>0$): $T_w(2^k) = T_w(2^{k-1}) + c \overset{(h.i.)}{=} (k-1).c + c = k.c$ como queria

mos. Logo $T_w(2^k) = k.c$. Agora podemos reescrever esta solução em função de n já que $n = 2^k$, i.e. $T_w(n) = c.\lg n = \Theta(\lg n)$. Por fim, note que a regra da suavização nos permite concluir que se $T_w(n) = \Theta(\lg n)$ quando n é potência de 2 então $T_w(n) = \Theta(\lg n)$ para qualquer valor de n já que $\lg n$ é uma função suave. $\blacksquare$