```
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 melhor caso:

No melhor caso, a chave key encontra-se na posição $mid = \lfloor (high + low)/2 \rfloor$. São realizadas duas comparações (linhas 5 e 9) independentemente do tamanho do vetor $A[1..n]$.

Portanto, o custo no melhor caso é constante, i.e. $T_b(n) = \Theta(1)$.