

## Binary

É um algoritmo eficiente para encontrar um elemento em um array ordenado, comparando o elemento do meio com o elemento de busca e repetindo a busca na metade esquerda ou direita do array, reduzindo o espaço de busca pela metade a cada iteração até encontrar o elemento desejado ou determinar que ele não está presente.

Pseudocódigo

```
ALGORITHM BinarySearch(A[0..n - 1], K)
//Implements nonrecursive binary search
//Input: An array A[0..n - 1] sorted in ascending order and
// a search key K
//Output: An index of the array's element that is equal to K
// or -1 if there is no such element
l ← 0; r ← n - 1
while l ≤ r do
    m ← ⌊(l + r)/2⌋
    if K = A[m] return m
    else if K < A[m] r ← m - 1
    else l ← m + 1
return -1
```

Pior Caso

$$C_{worst}(n) = \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n + 1) \rceil.$$

BINARY SEARCH					
Search 78 Divide from middle check mid	21	34	mid 43	57	66 78
78 > 43 look for 78 in right half	21	34	43	57	66 78
new mid=66 check mid	21	34	43	57	mid 66 78
78 > 66 look for 78 on right half	21	34	43	57	66 78
new mid=78 element found at mid	21	34	43	57	66 mid 78

# Search

Algoritmos de busca

## Sequential

Algoritmo simples de força bruta que compara cada elemento de uma lista com o item de busca até encontrar uma correspondência, ou até percorrer toda a lista sem encontrar uma correspondência.

Pseudocódigo

```
ALGORITHM SequentialSearch2(A[0..n], K)
//Implements sequential search with a search key as a sentinel
//Input: An array A of n elements and a search key K
//Output: The index of the first element in A[0..n - 1] whose
value is
// equal to K or -1 if no such element is found
A[n] ← K
i ← 0
while A[i] = K do
    i ← i + 1
if i < n return i
else return -1
```

