PRIMEIRO TRABALHO
ALGORITMOS DE ORDENAÇÃO

Grupo:
João Marcos R.
Raildom da Rocha

ALGORITMOS APRESENTADOS

- INSERTION SORT

- BINARY INSERTION SORT

INSERTION SORT

ASPECTOS HISTÓRICOS

ASPECTOS HISTÓRICO

O Insertion Sort não tem autor ou data específica de criação, pois surgiu naturalmente da forma intuitiva como pessoas organizam itens, como cartas, inserindo cada novo elemento na posição correta. Formalizado nas décadas de 1940 e 1950 com os primeiros computadores.

DEFINIÇÃO

INSERTION SORT

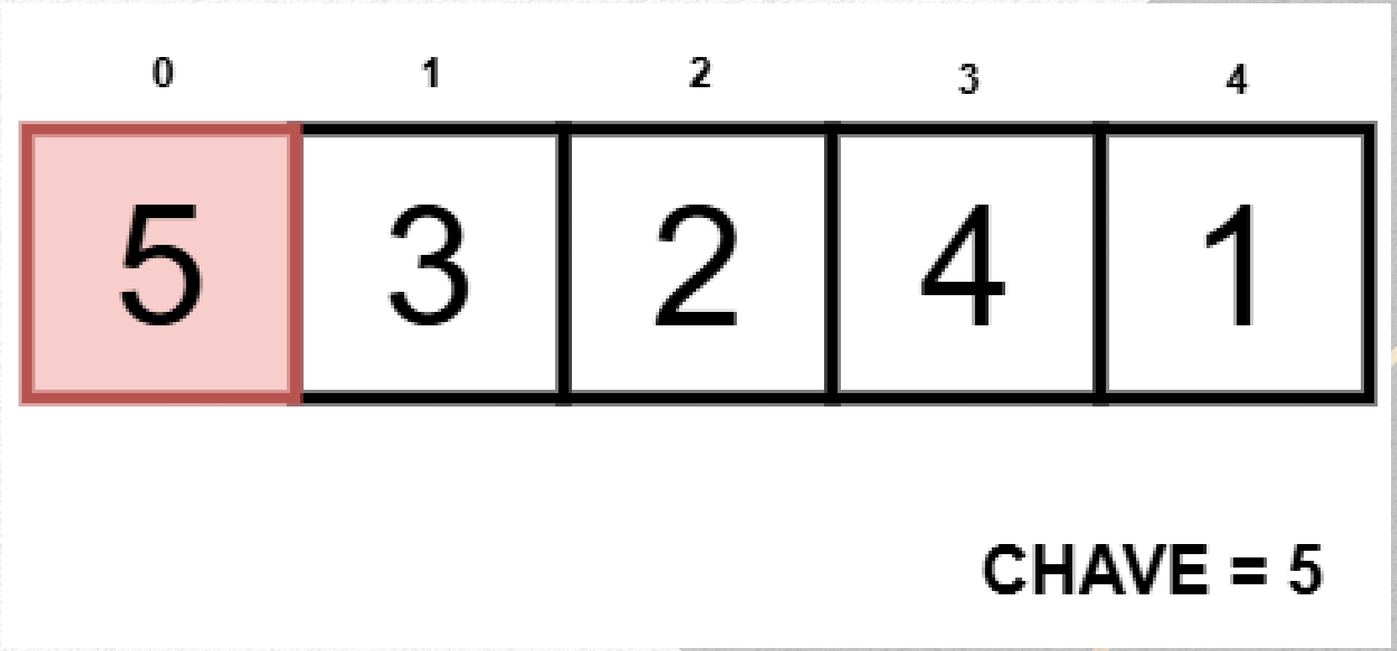
O Insertion Sort é um algoritmo de ordenação simples, in-place e estável que constrói a sequência ordenada um elemento por vez, inserindo cada novo elemento na posição correta, semelhante à organização de cartas na mão.

CÓDIGO

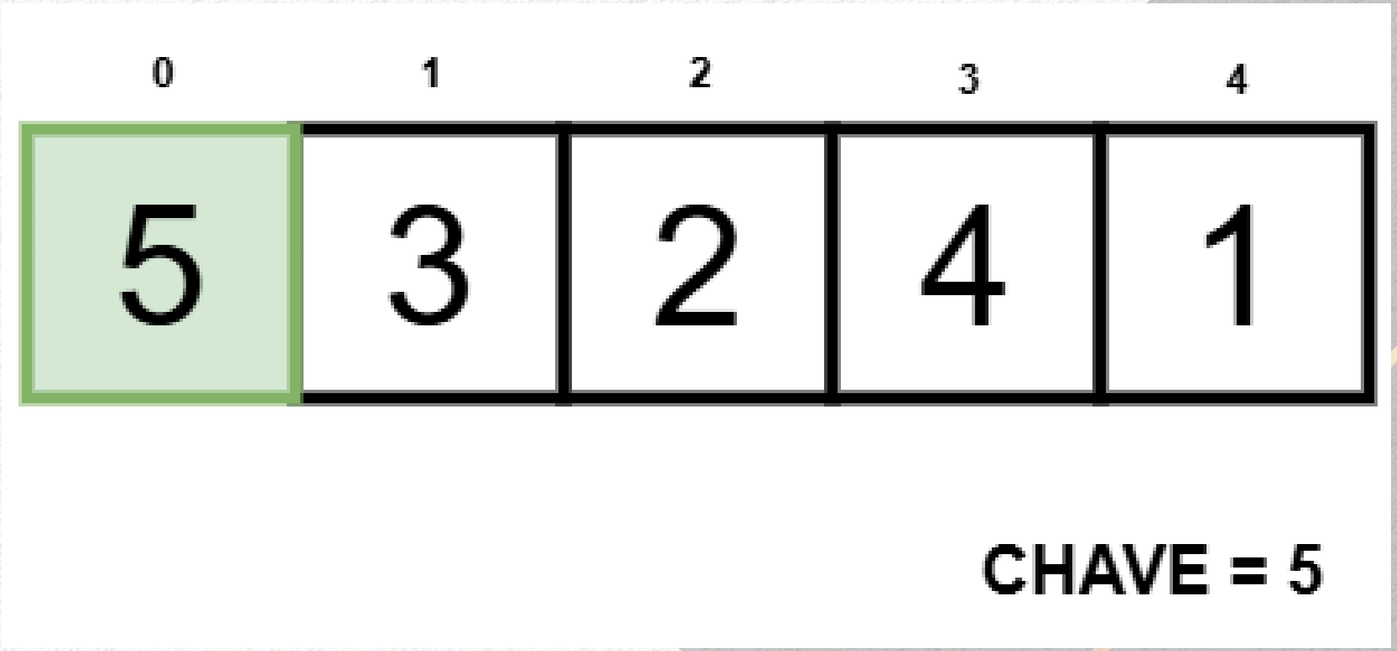
```python
def insertion_sort(arr, tam):
    for i in range(1, tam):
        chave, j = arr[i], i - 1
        while j >= 0 and arr[j] > chave:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = chave
    return arr
```

EXEMPLO DE FUNCIONAMENTO

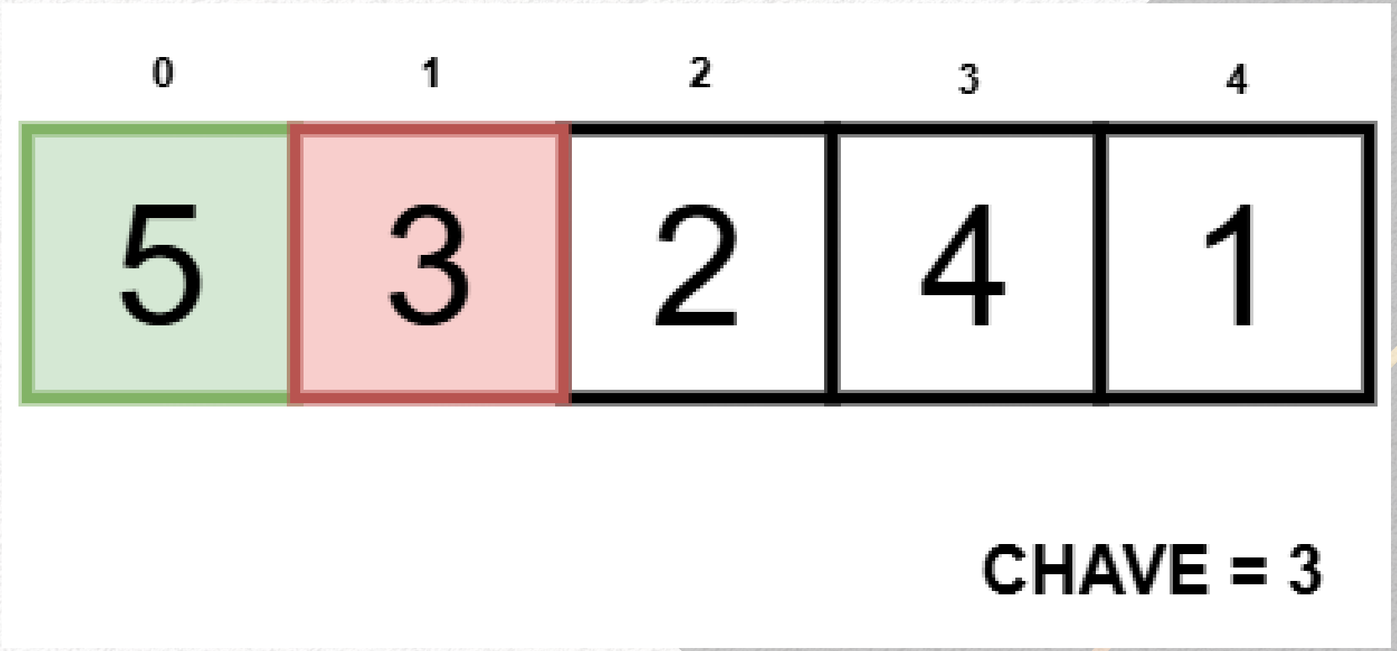
EXEMPLO DE FUNCIONAMENTO



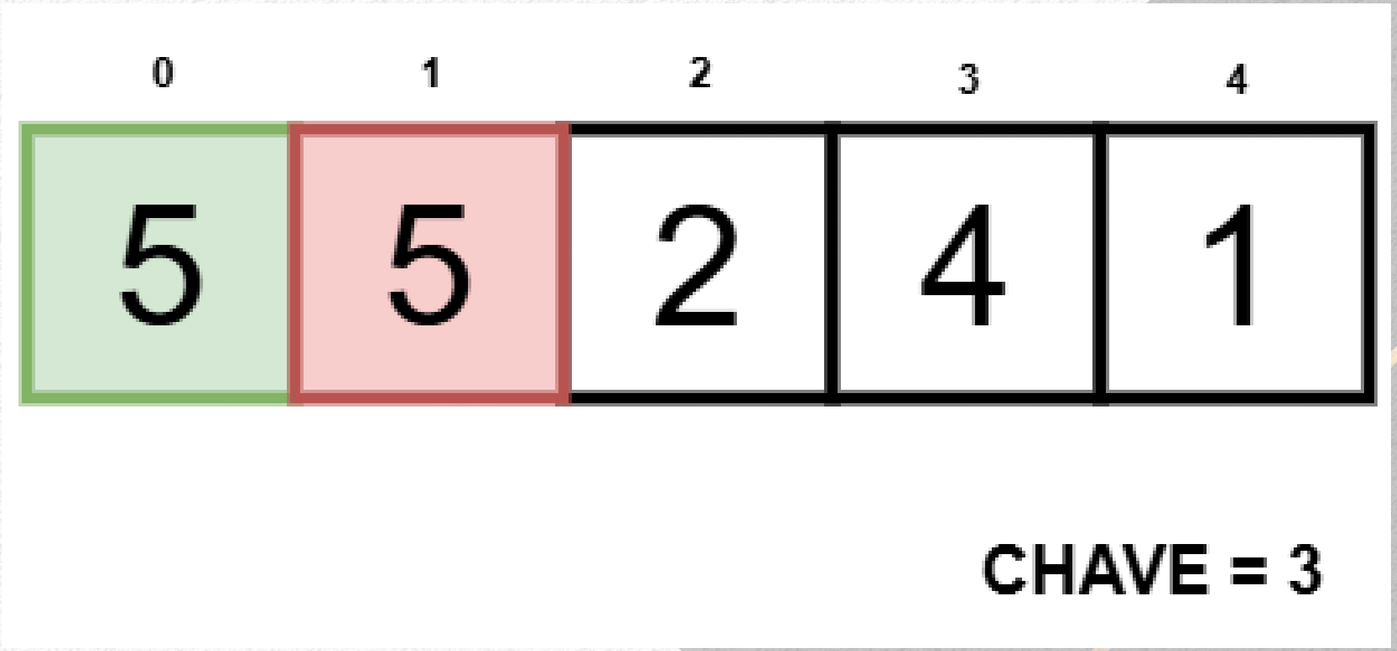
EXEMPLO DE FUNCIONAMENTO



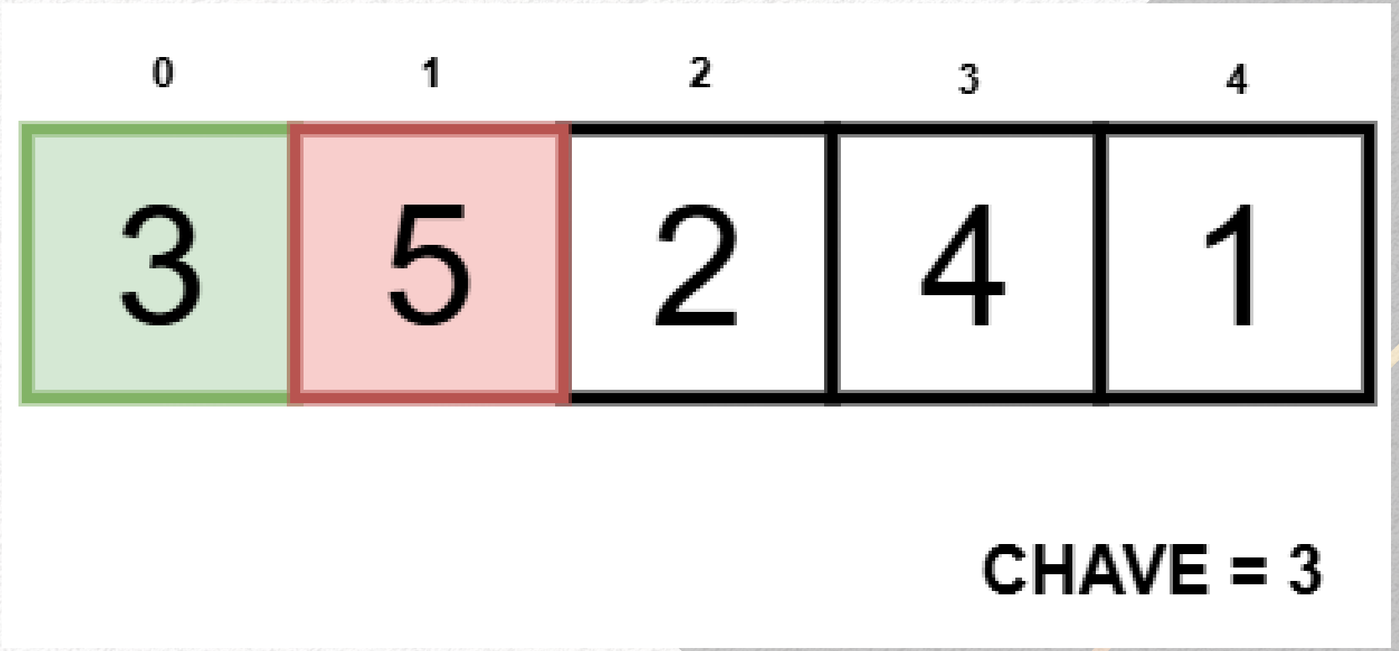
EXEMPLO DE FUNCIONAMENTO



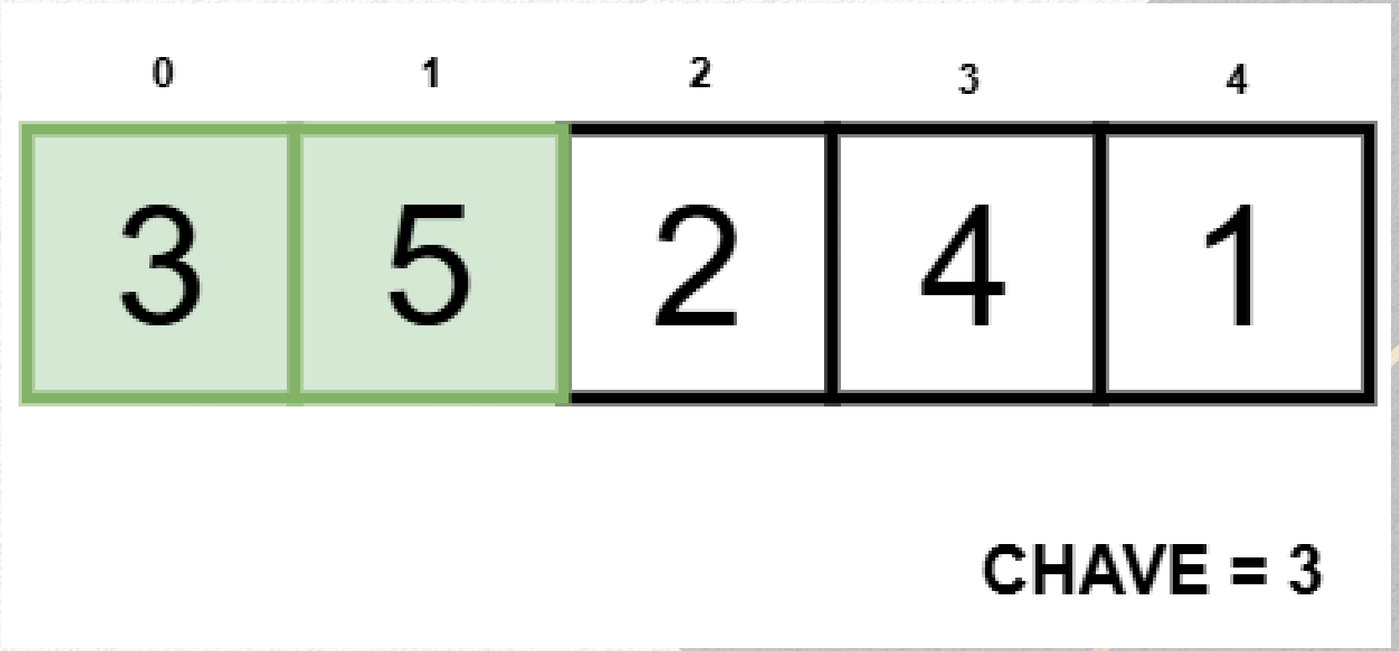
EXEMPLO DE FUNCIONAMENTO



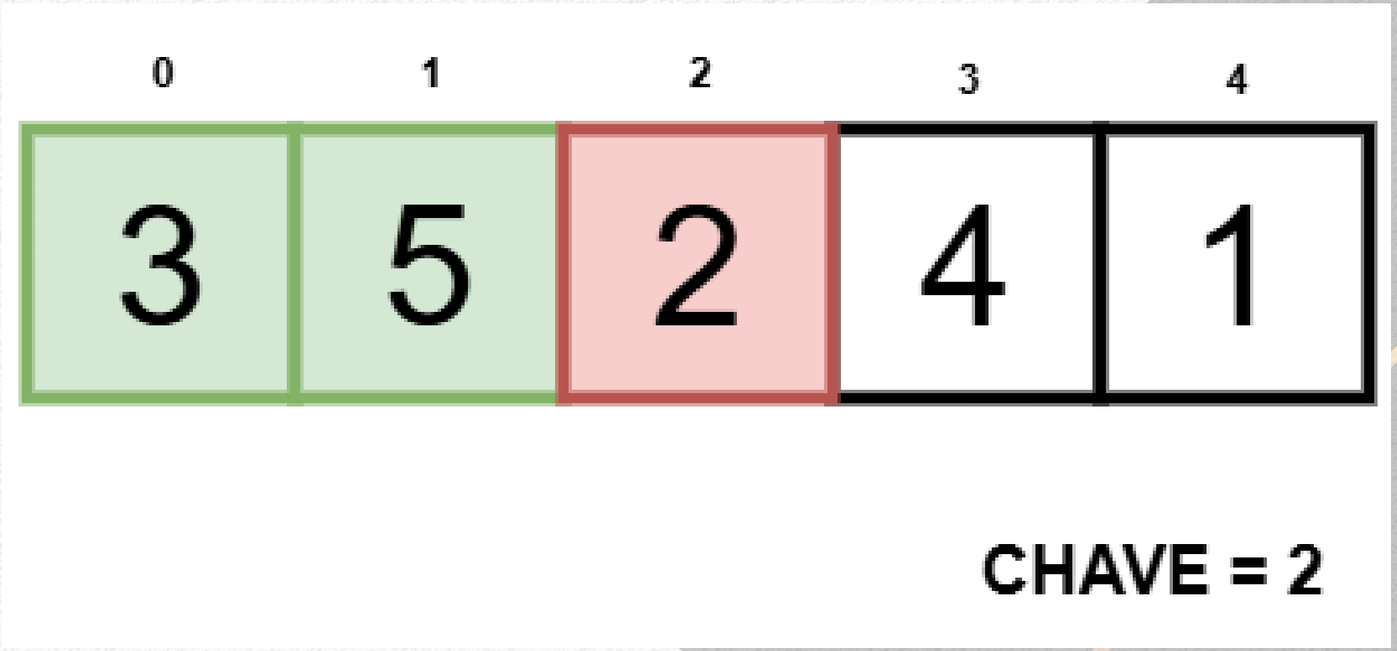
EXEMPLO DE FUNCIONAMENTO



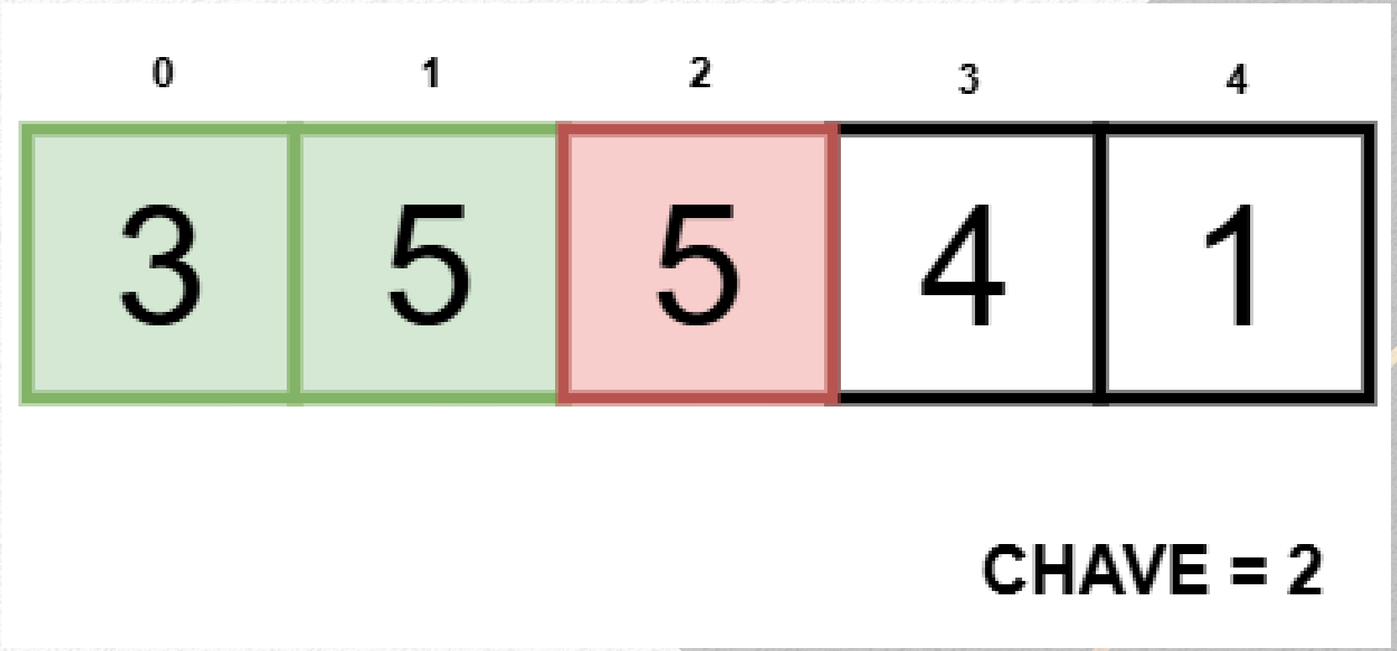
EXEMPLO DE FUNCIONAMENTO



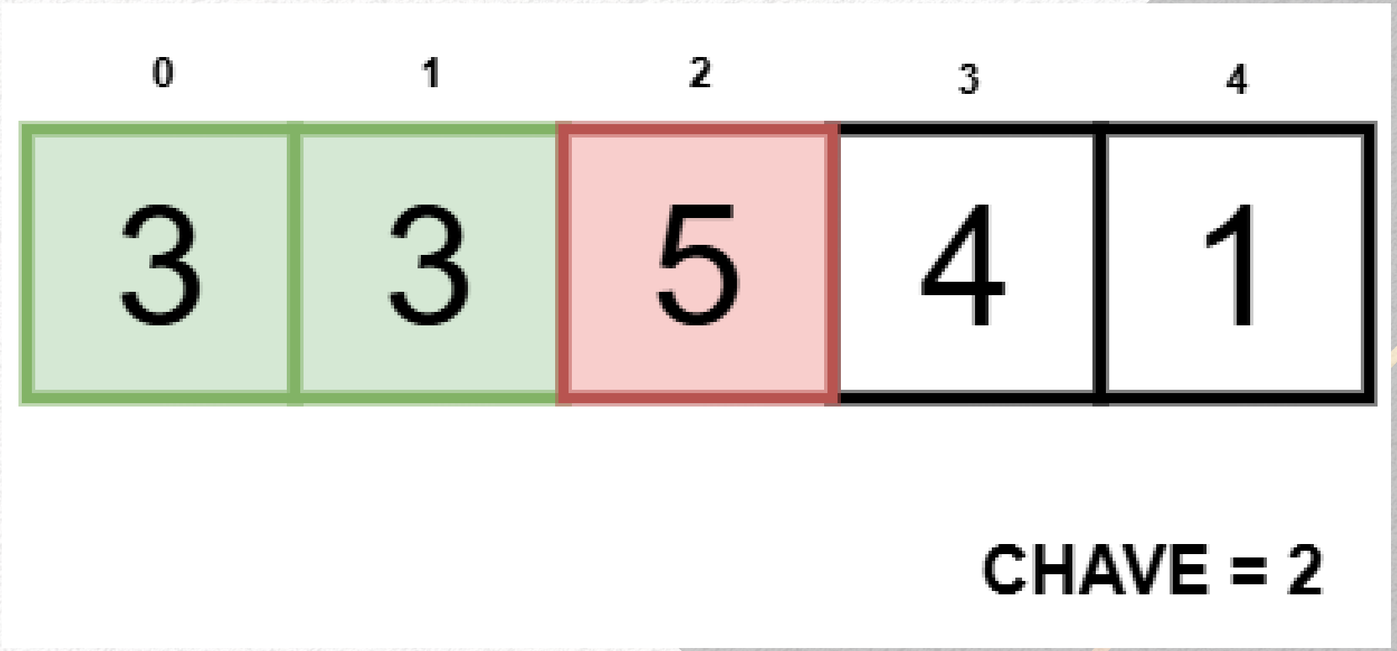
EXEMPLO DE FUNCIONAMENTO



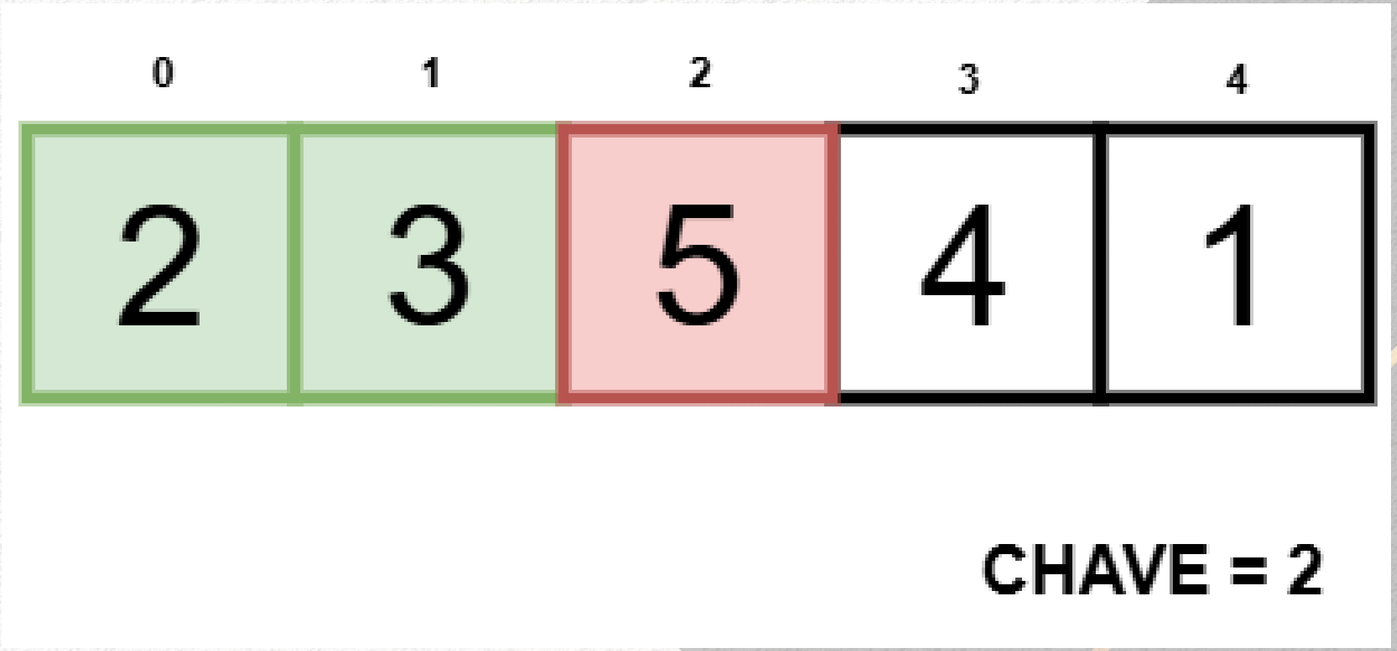
EXEMPLO DE FUNCIONAMENTO



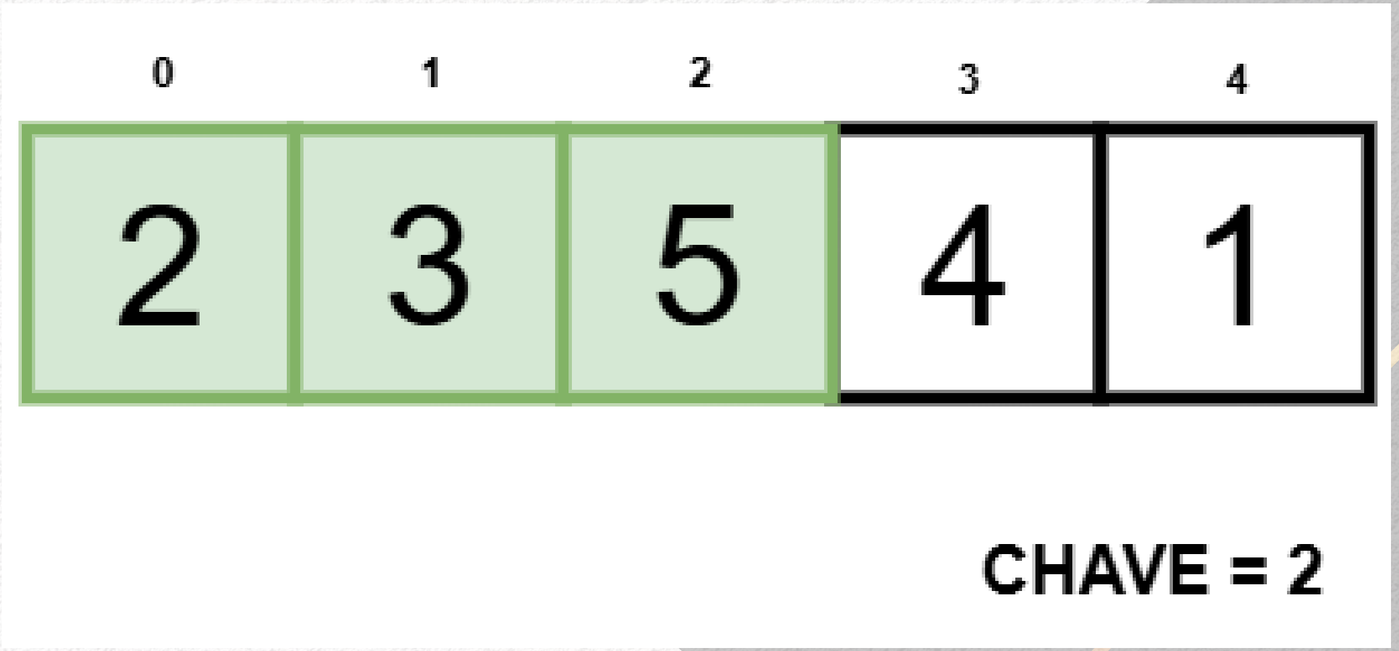
EXEMPLO DE FUNCIONAMENTO



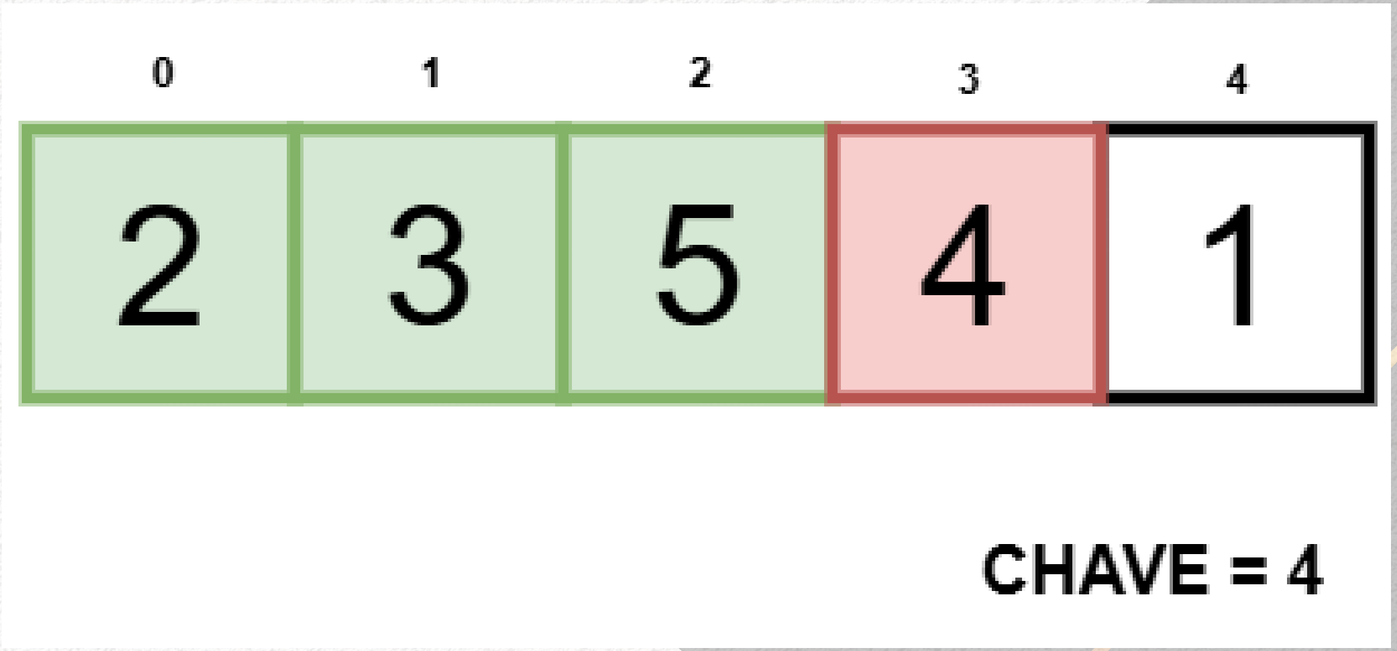
EXEMPLO DE FUNCIONAMENTO



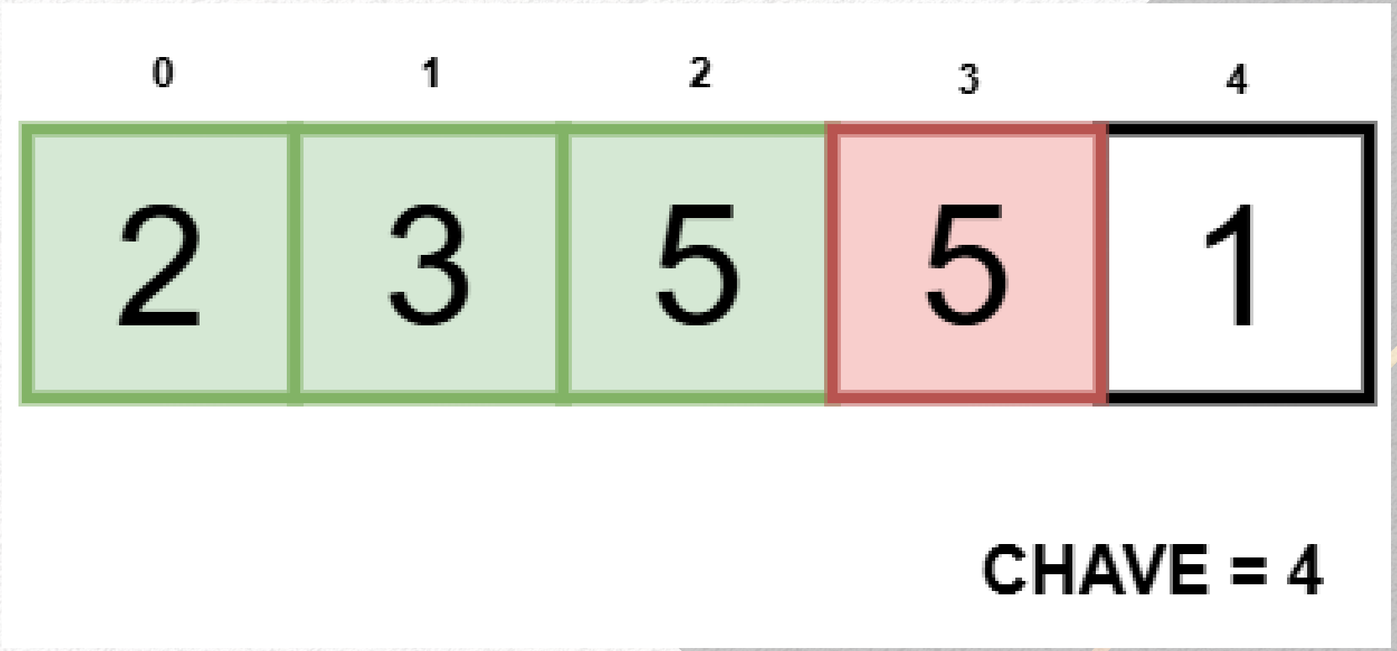
EXEMPLO DE FUNCIONAMENTO



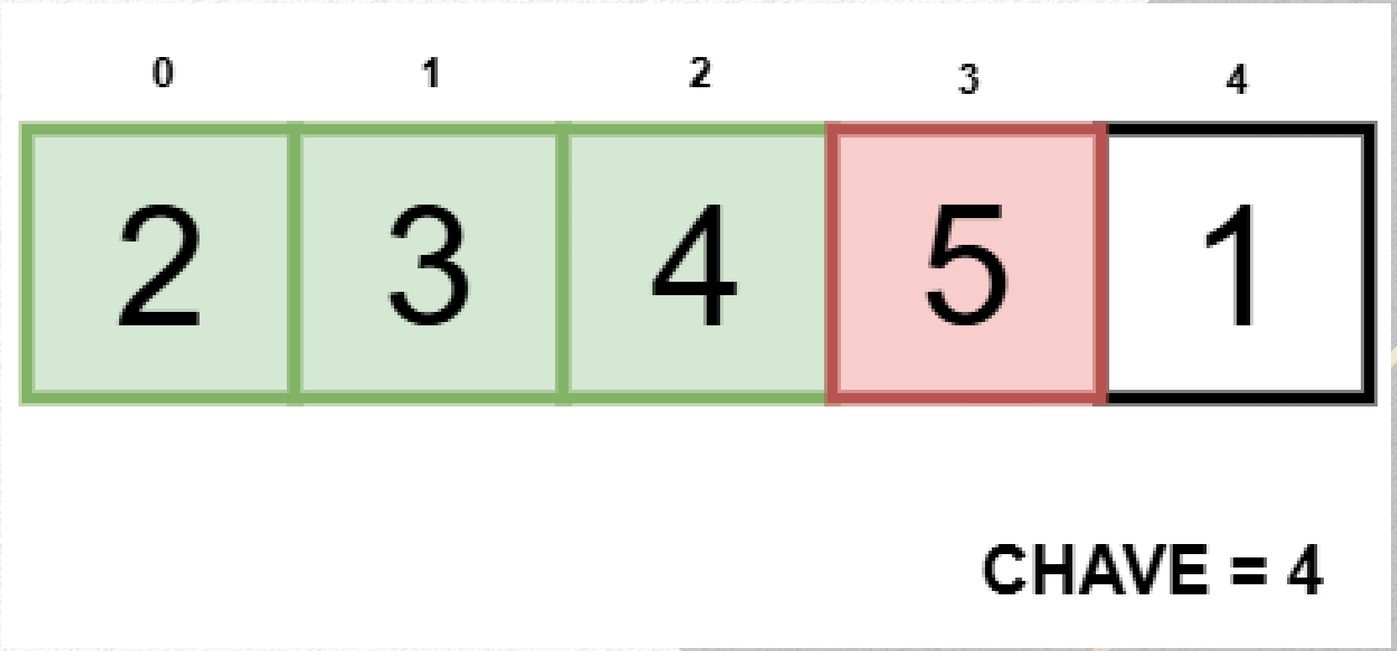
EXEMPLO DE FUNCIONAMENTO



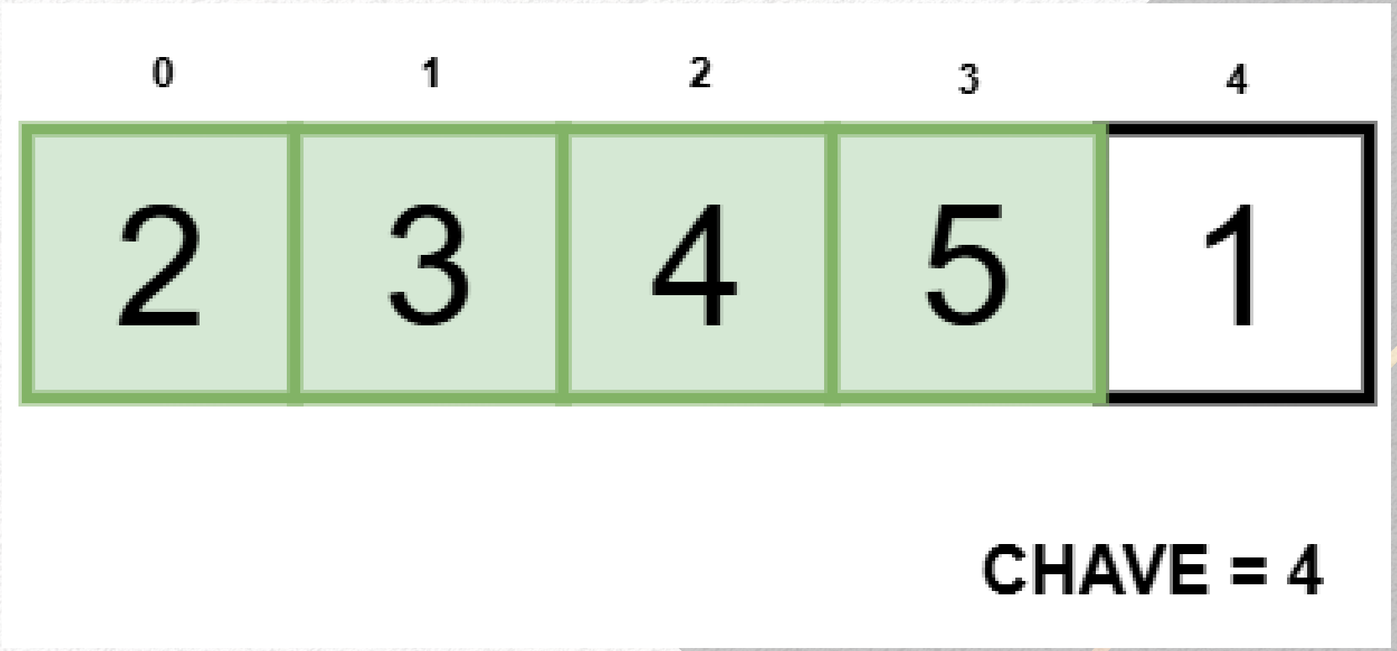
EXEMPLO DE FUNCIONAMENTO



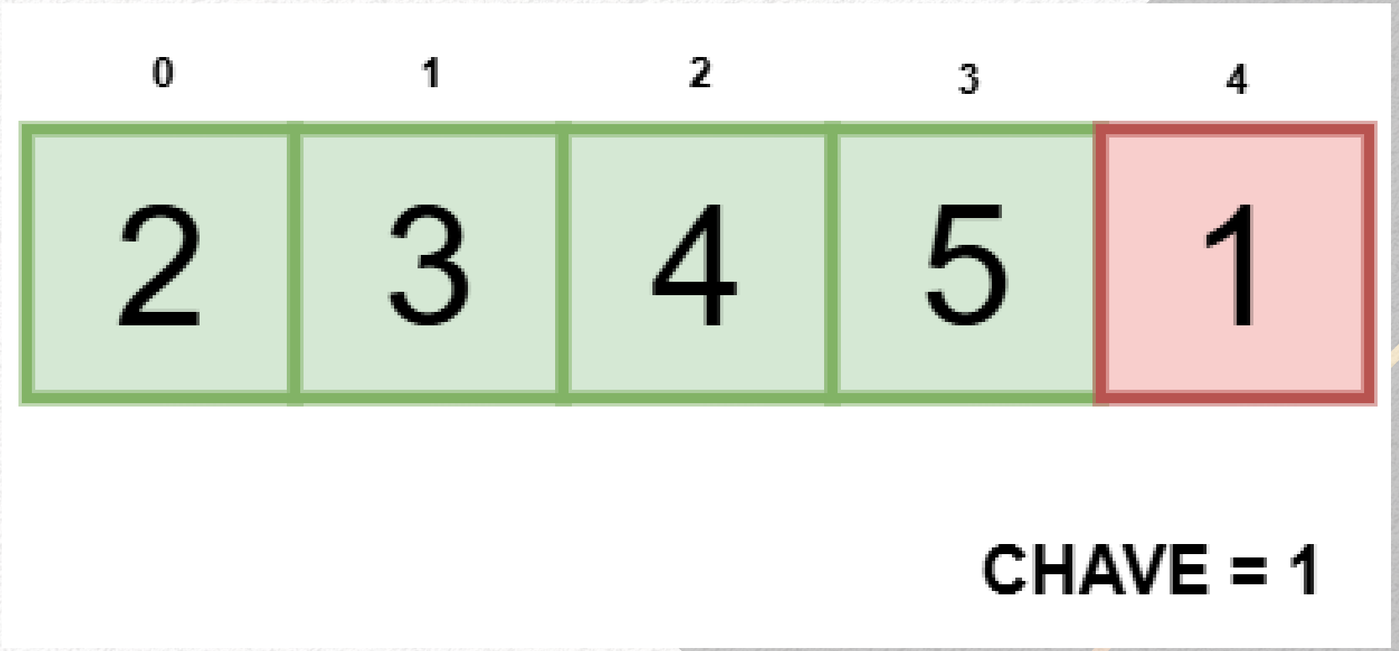
EXEMPLO DE FUNCIONAMENTO



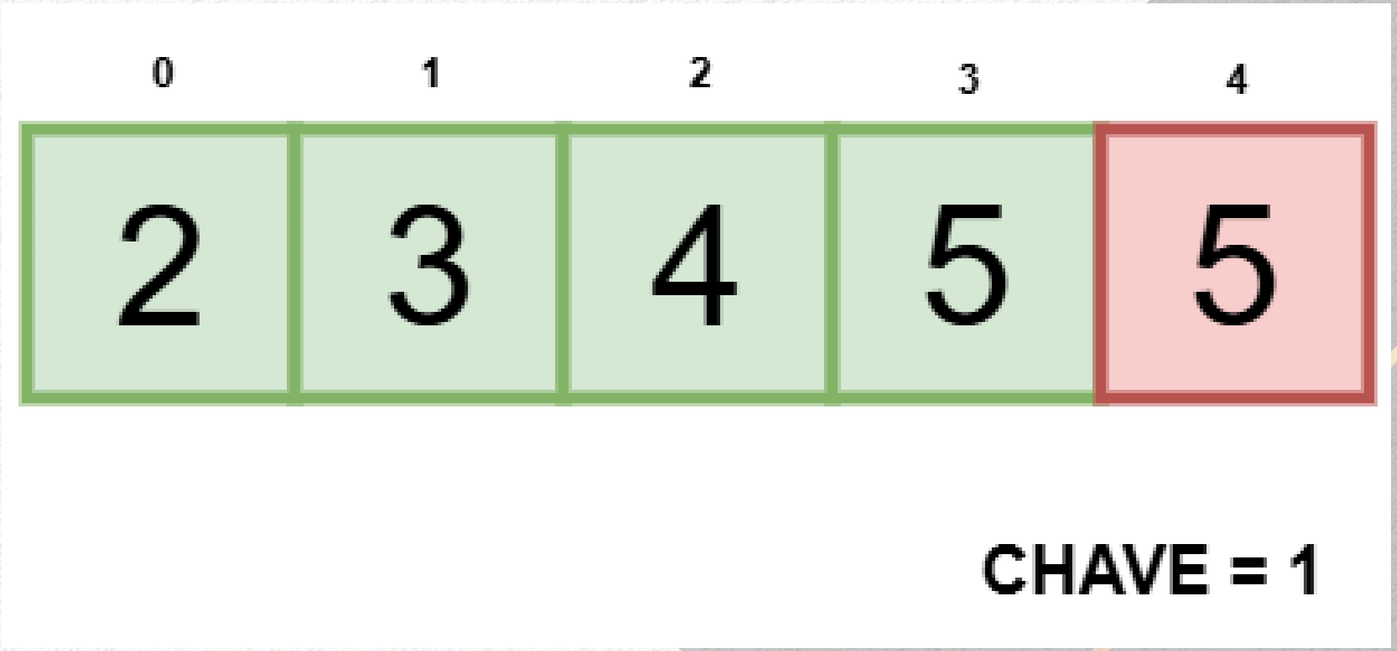
EXEMPLO DE FUNCIONAMENTO



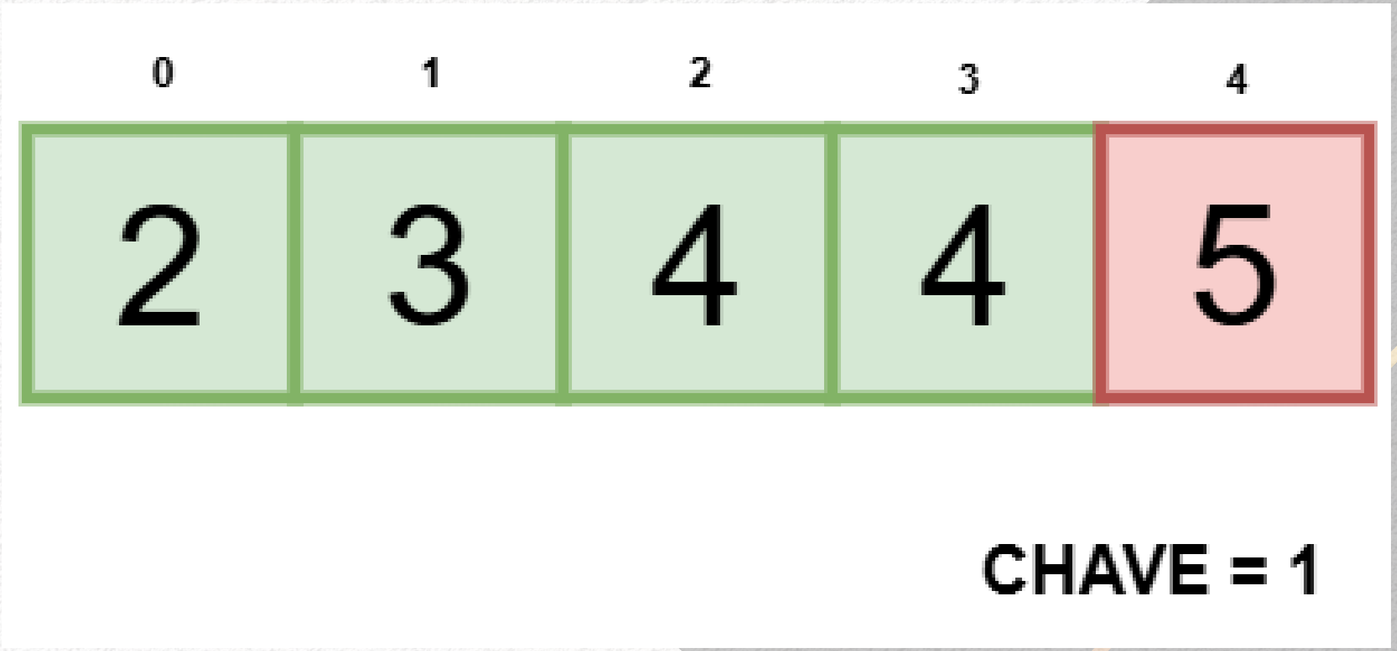
EXEMPLO DE FUNCIONAMENTO



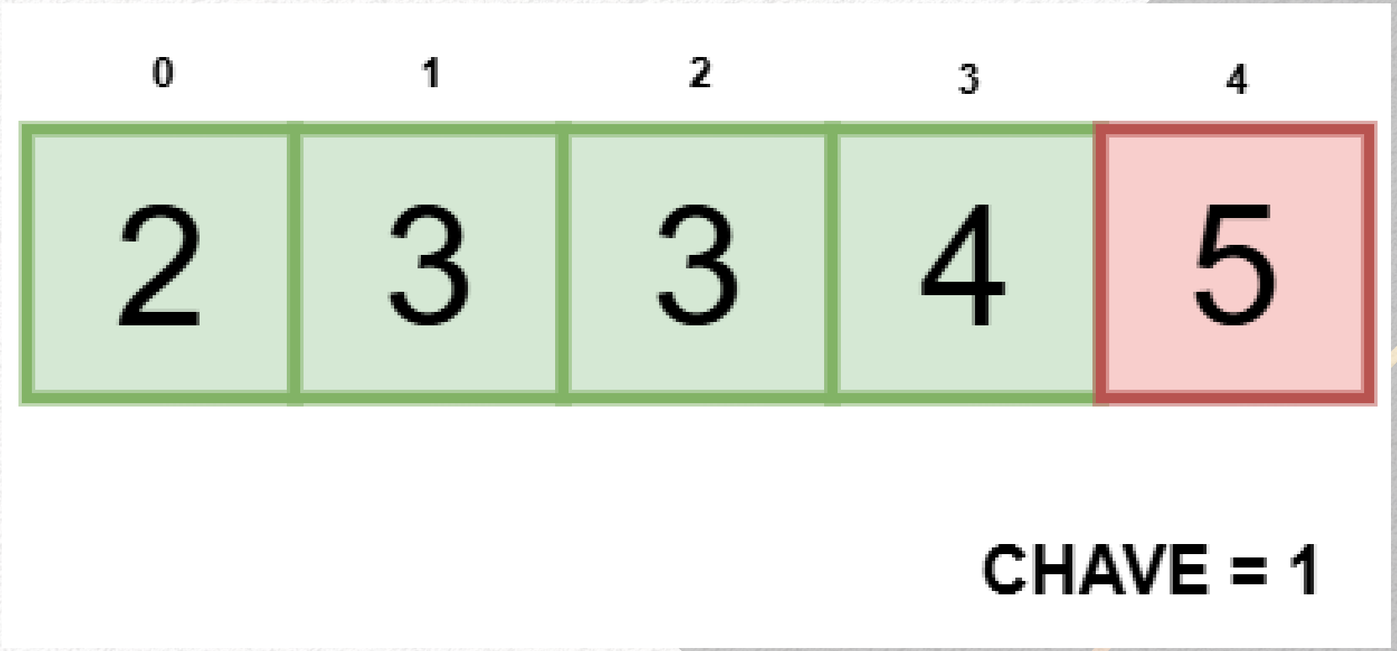
EXEMPLO DE FUNCIONAMENTO



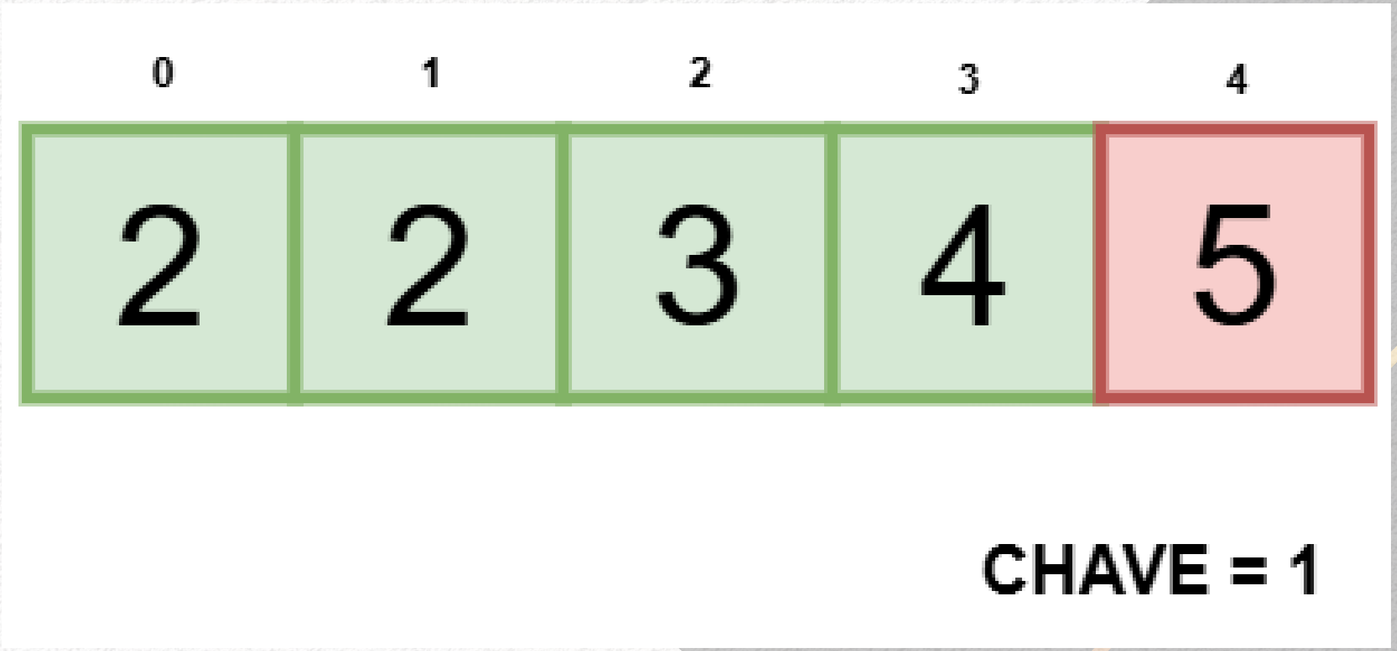
EXEMPLO DE FUNCIONAMENTO



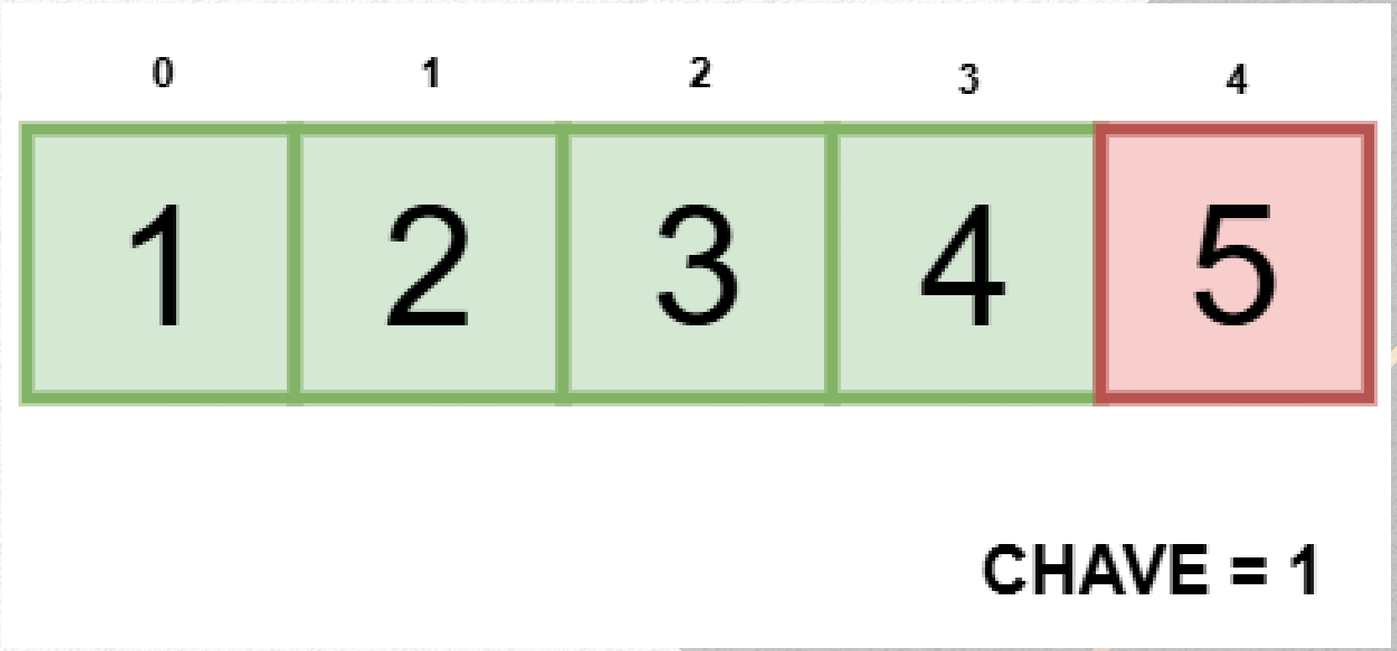
EXEMPLO DE FUNCIONAMENTO



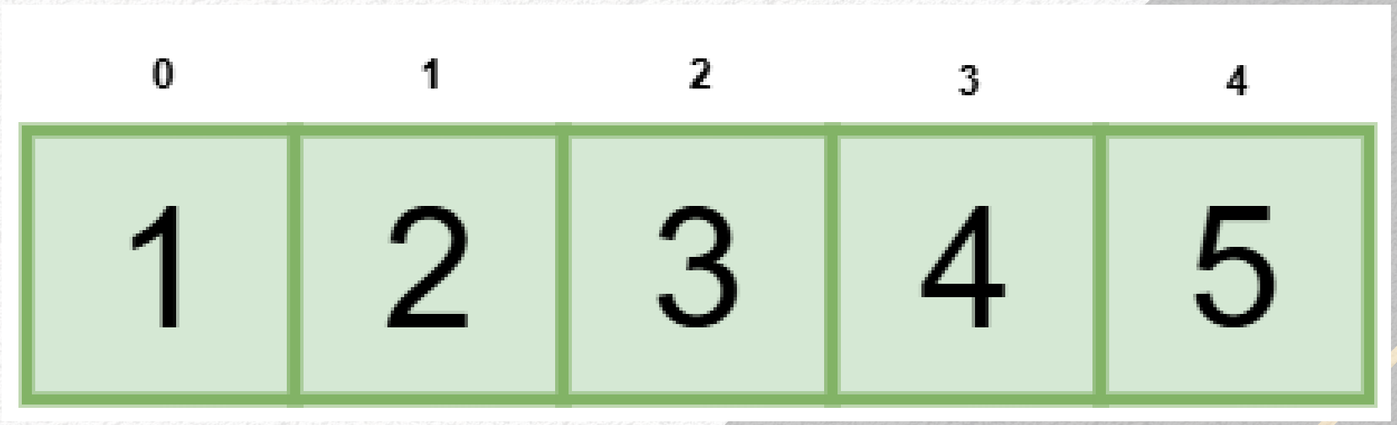
EXEMPLO DE FUNCIONAMENTO



EXEMPLO DE FUNCIONAMENTO

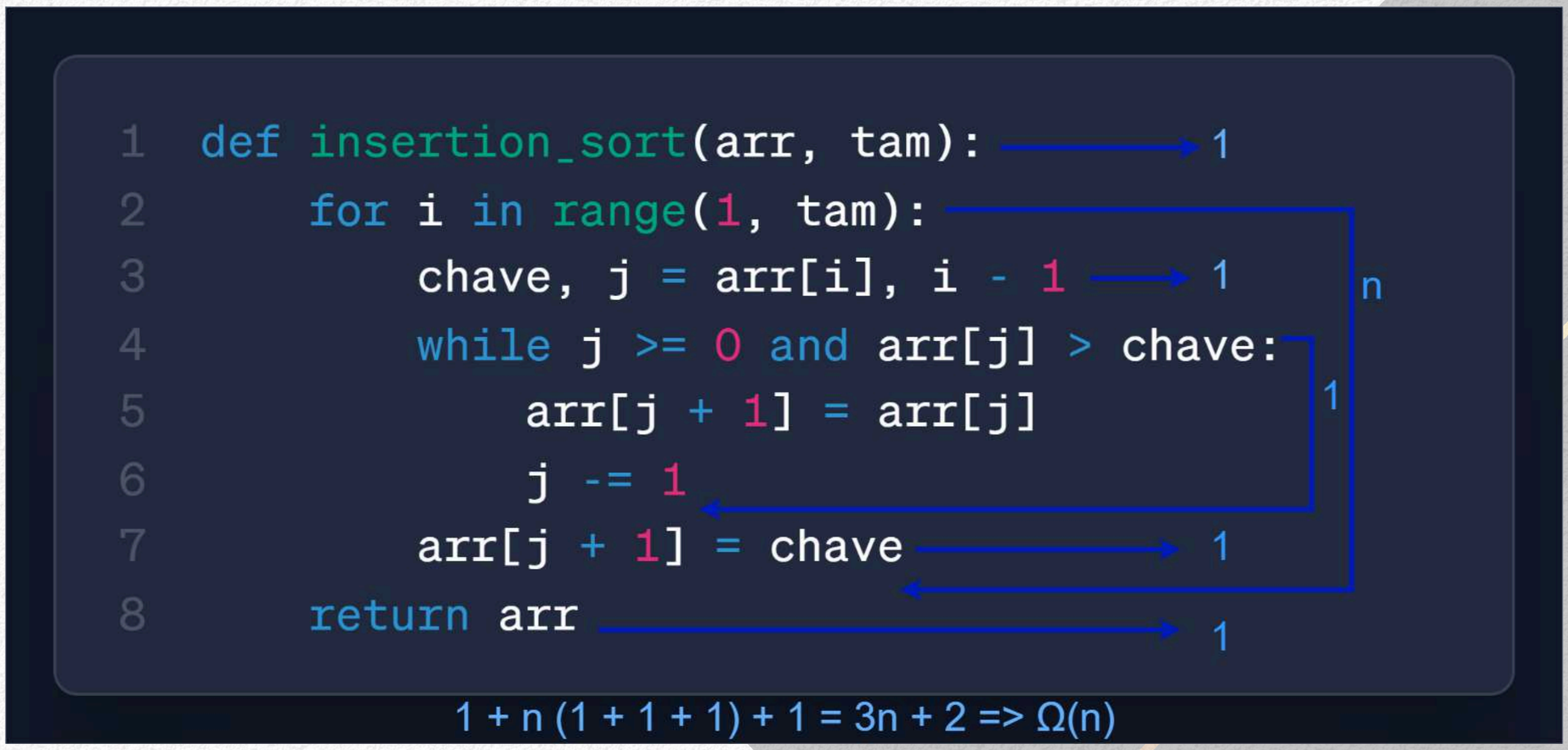


EXEMPLO DE FUNCIONAMENTO

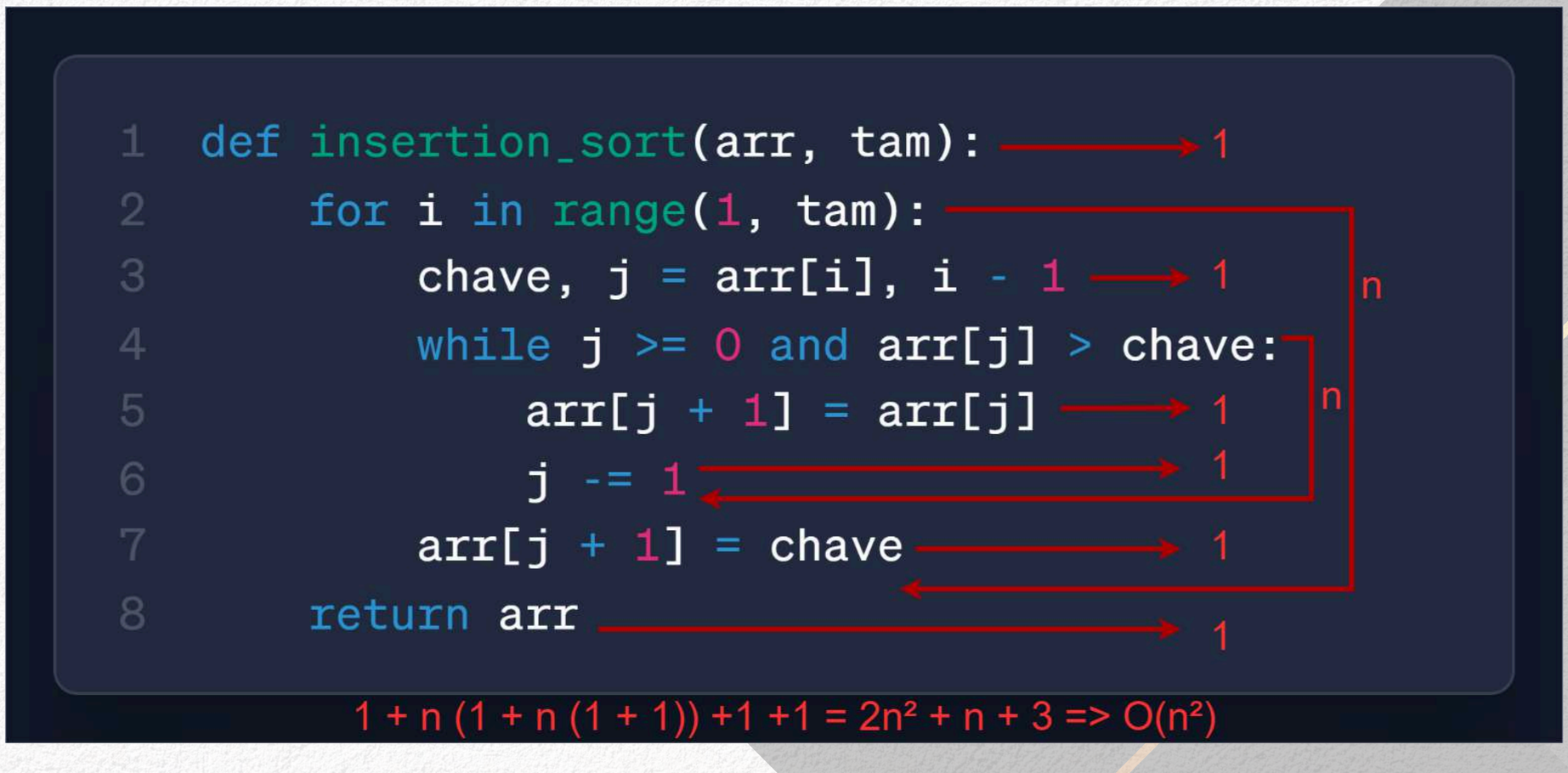


ANÁLISE DE CUSTO

ANÁLISE DE CUSTO MELHOR CASO



ANÁLISE DE CUSTO PIOR CASO



BINARY INSERTION SORT

ASPECTOS HISTÓRICOS

ASPECTOS HISTÓRICO

Assim como o insertion sort, o binary insertion sort não possue autor ou data precisa, surgiu como variação prática em materiais acadêmicos após os primeiros métodos de ordenação, sendo usado em contextos educacionais e para listas pequenas devido à sua simplicidade e eficiência.

DEFINIÇÃO

BINARY INSERTION SORT

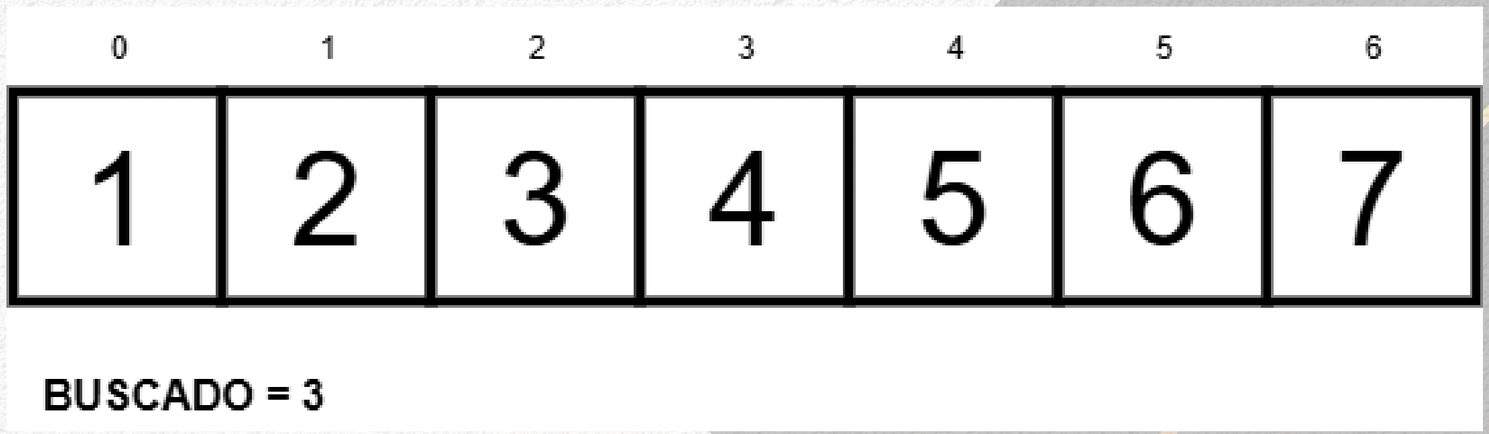
O Binary Insertion Sort é uma variação do Insertion Sort que otimiza a busca pela posição correta de cada elemento na sub-array ordenada, utilizando busca binária em vez de busca linear, reduzindo o número de comparações.

CÓDIGO

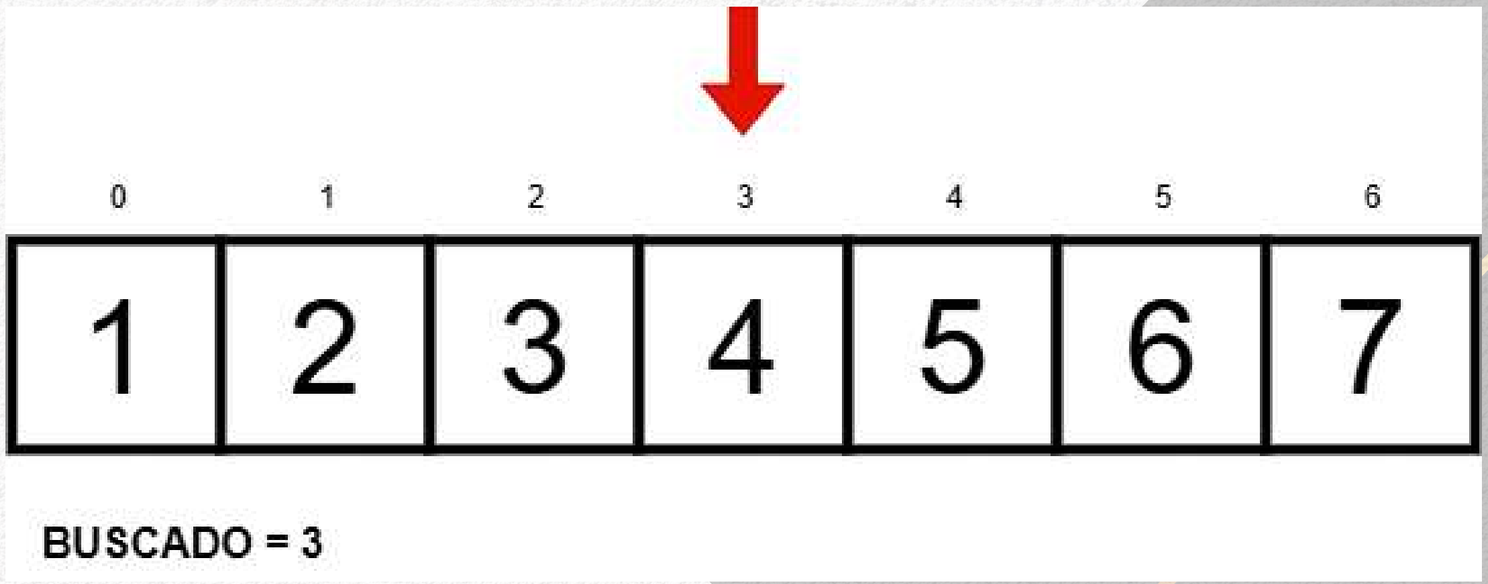
```python
def binary_search(arr, chave, start, end):
    while start < end:
        mid = (start + end) // 2
        if arr[mid] < chave:
            start = mid + 1
        else:
            end = mid
    return start

def binary_insertion_sort(arr, tam):
    for i in range(1, tam):
        chave = arr[i]
        pos = binary_search(arr, chave, 0, i)
        for j in range(i, pos, -1):
            arr[j] = arr[j-1]
        arr[pos] = chave
    return arr
```

EXEMPLO DE FUNCIONAMENTO

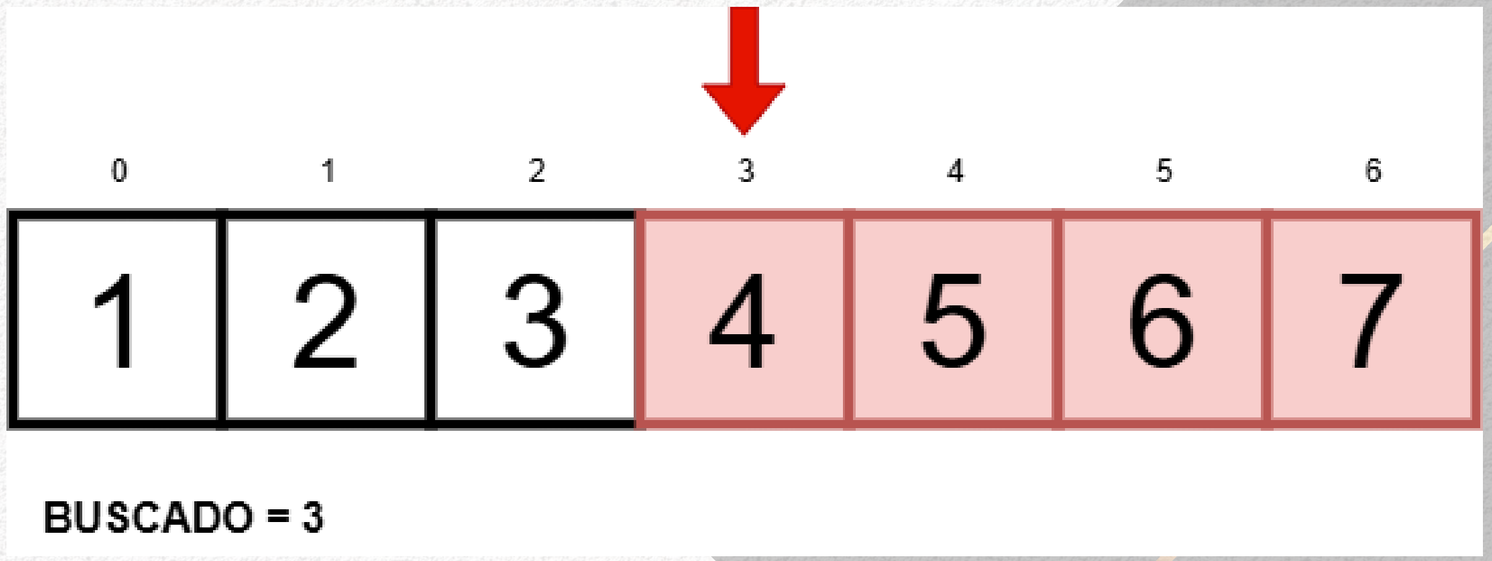
EXEMPLO DE FUNCIONAMENTO



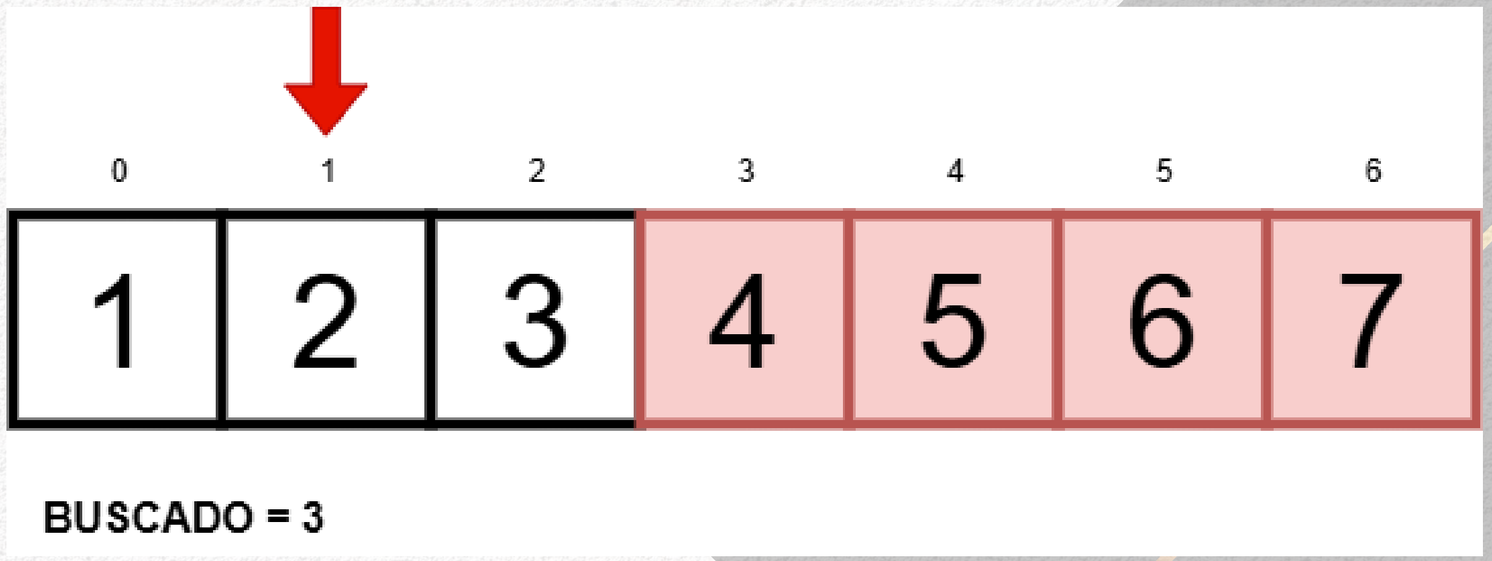
EXEMPLO DE FUNCIONAMENTO



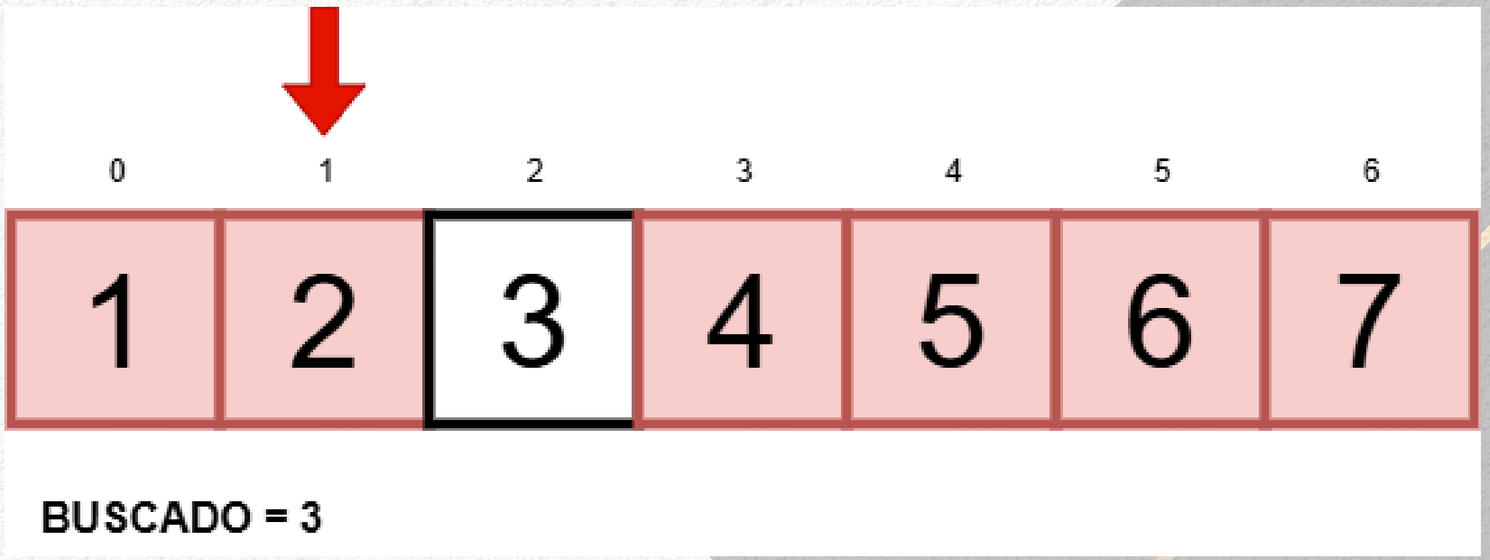
EXEMPLO DE FUNCIONAMENTO



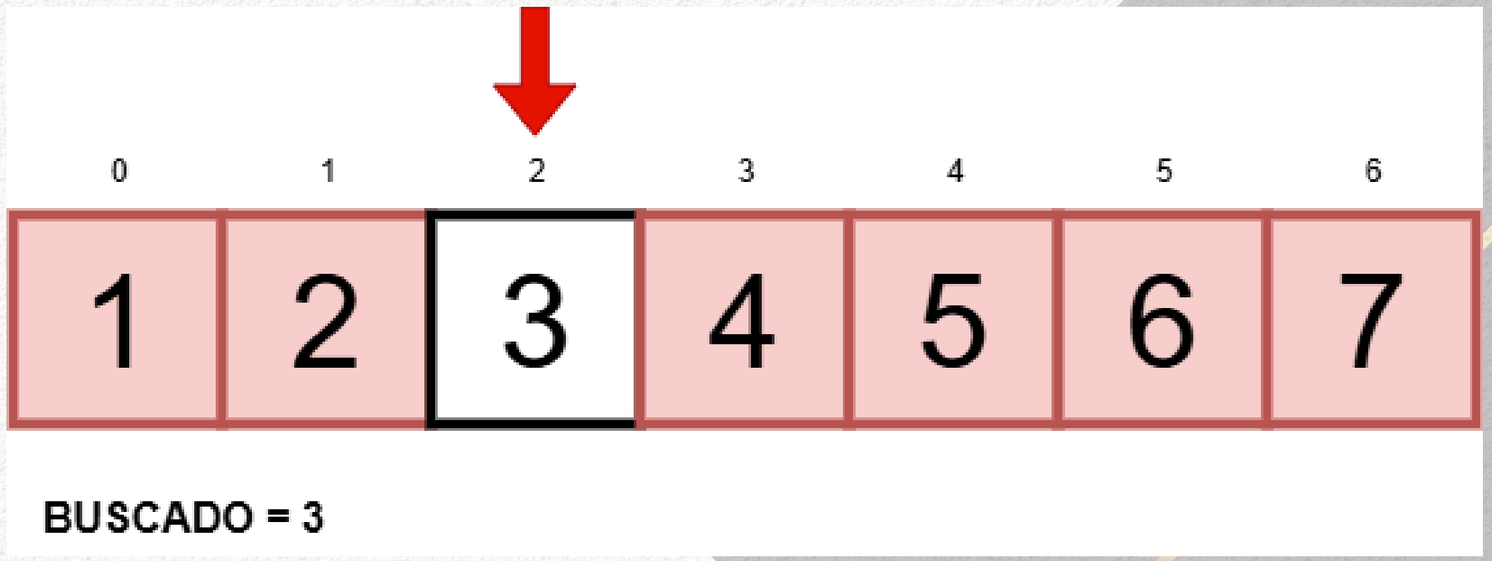
EXEMPLO DE FUNCIONAMENTO



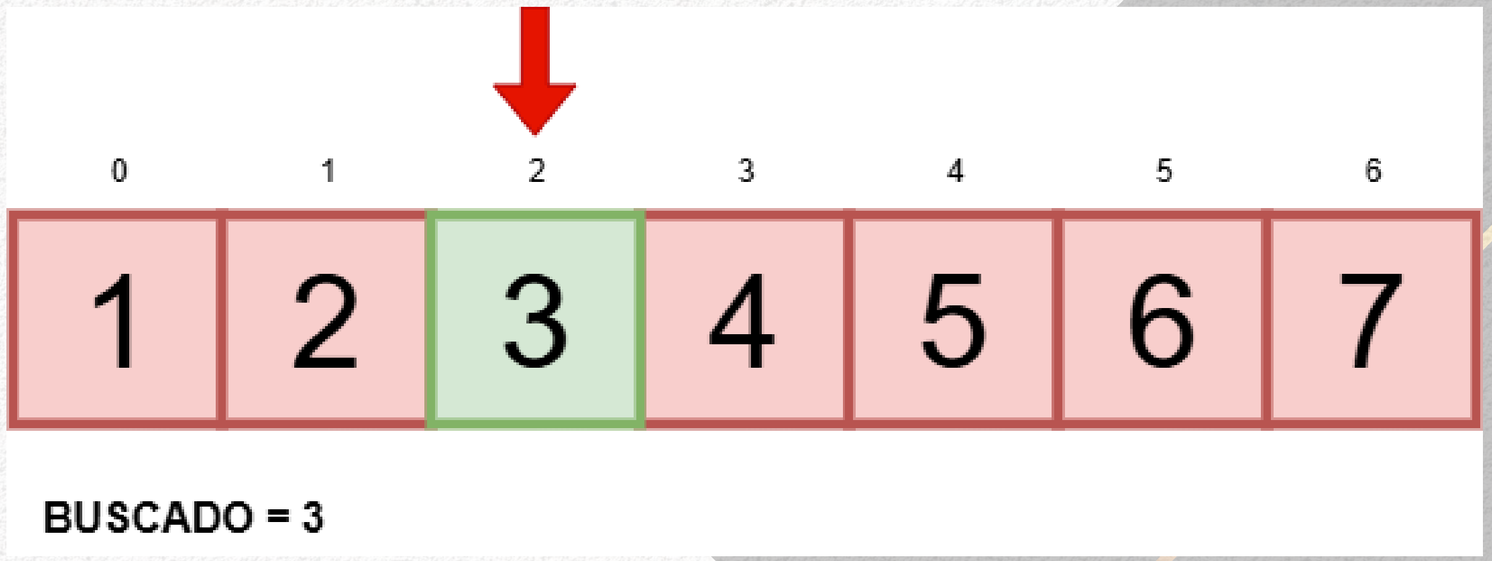
EXEMPLO DE FUNCIONAMENTO



EXEMPLO DE FUNCIONAMENTO



EXEMPLO DE FUNCIONAMENTO

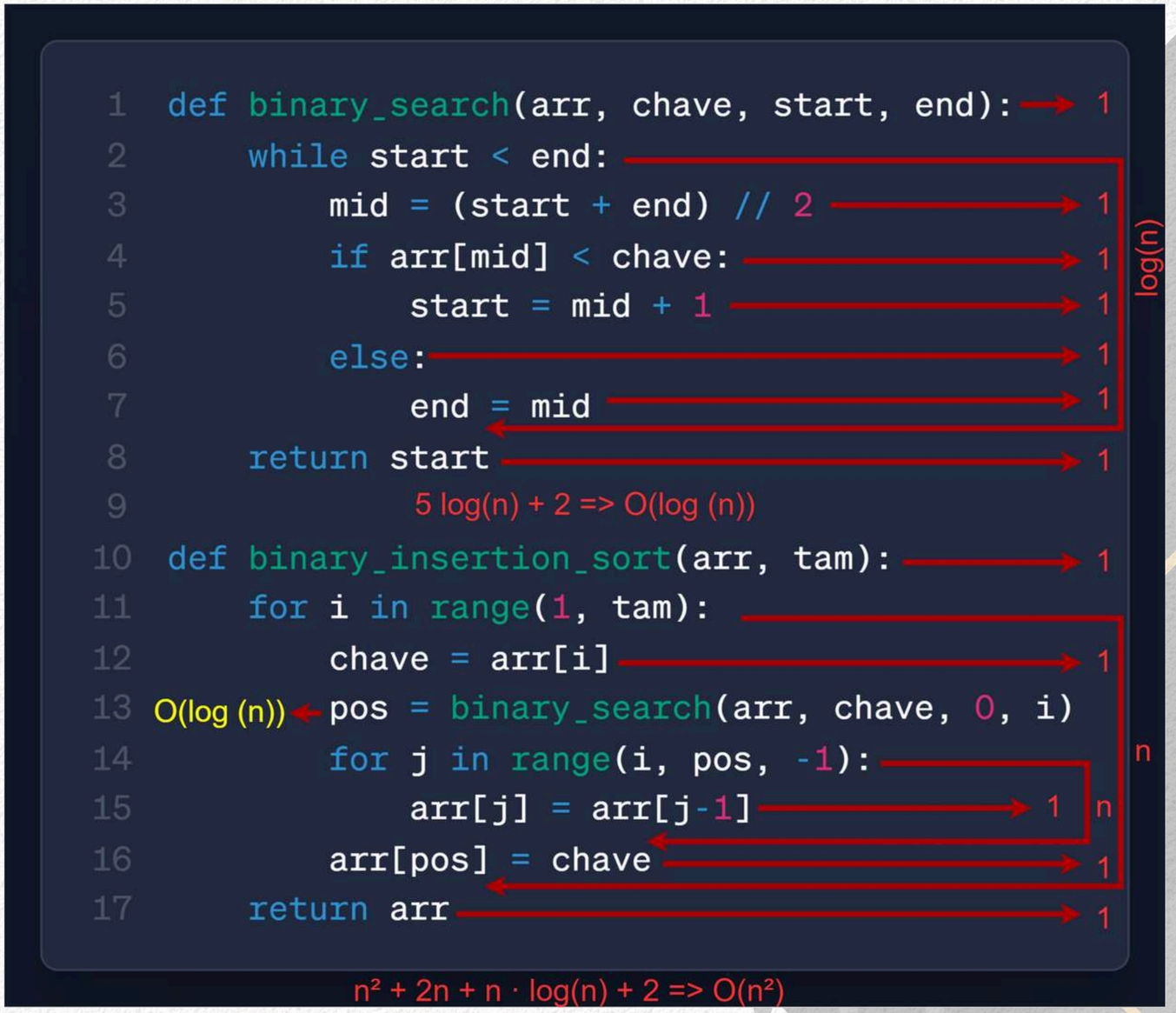


ANÁLISE DE CUSTO

ANÁLISE DE CUSTO MELHOR CASO

```
1  def binary_search(arr, chave, start, end):                    → 1
2      while start < end:                                        ─┐
3          mid = (start + end) // 2                              → 1  │
4          if arr[mid] < chave:                                  → 1  │
5              start = mid + 1                                   → 1  │ log(n)
6          else:                                                 → 1  │
7              end = mid                                         → 1  │
8      return start                                              → 1 ─┘
9              5 log(n) + 2 => O(log (n))
10 def binary_insertion_sort(arr, tam):                          → 1
11     for i in range(1, tam):                                   ─┐
12         chave = arr[i]                                        → 1  │
13 O(log (n)) ← pos = binary_search(arr, chave, 0, i)            │ n
14         for j in range(i, pos, -1):                           ─┐  │
15             arr[j] = arr[j-1]                                 → 1 1 │
16         arr[pos] = chave                                      → 1  │
17     return arr                                                → 1 ─┘

          n · log(n) + 3n +2 = Ω(n · log(n))
```

ANÁLISE DE CUSTO PIOR CASO



TESTES

TESTES

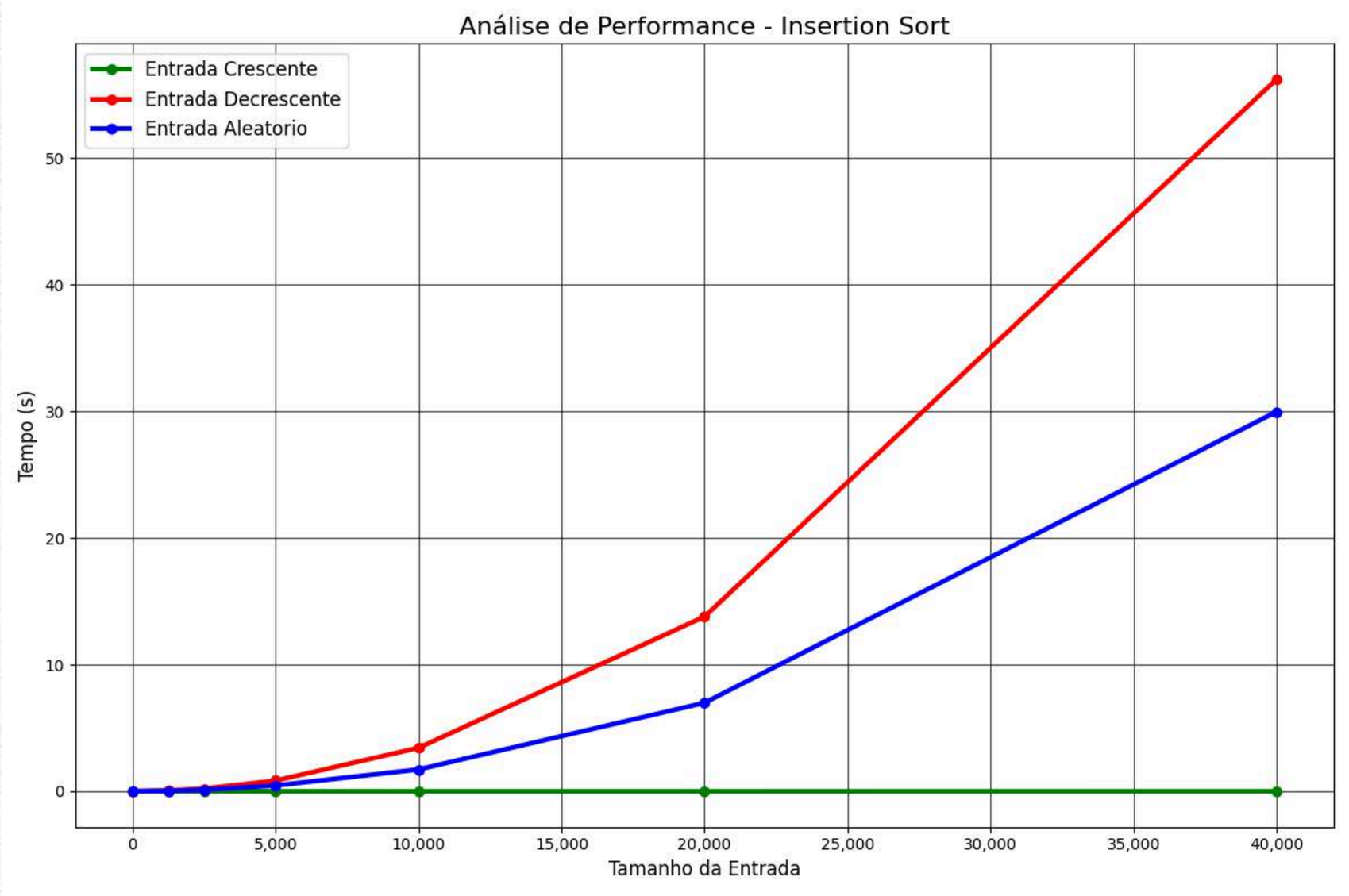
Sets usados: 1.250, 2.500, 5.000, 10.000, 20.000 e 40.000

Ordem: crescente, decrescente e aleatório

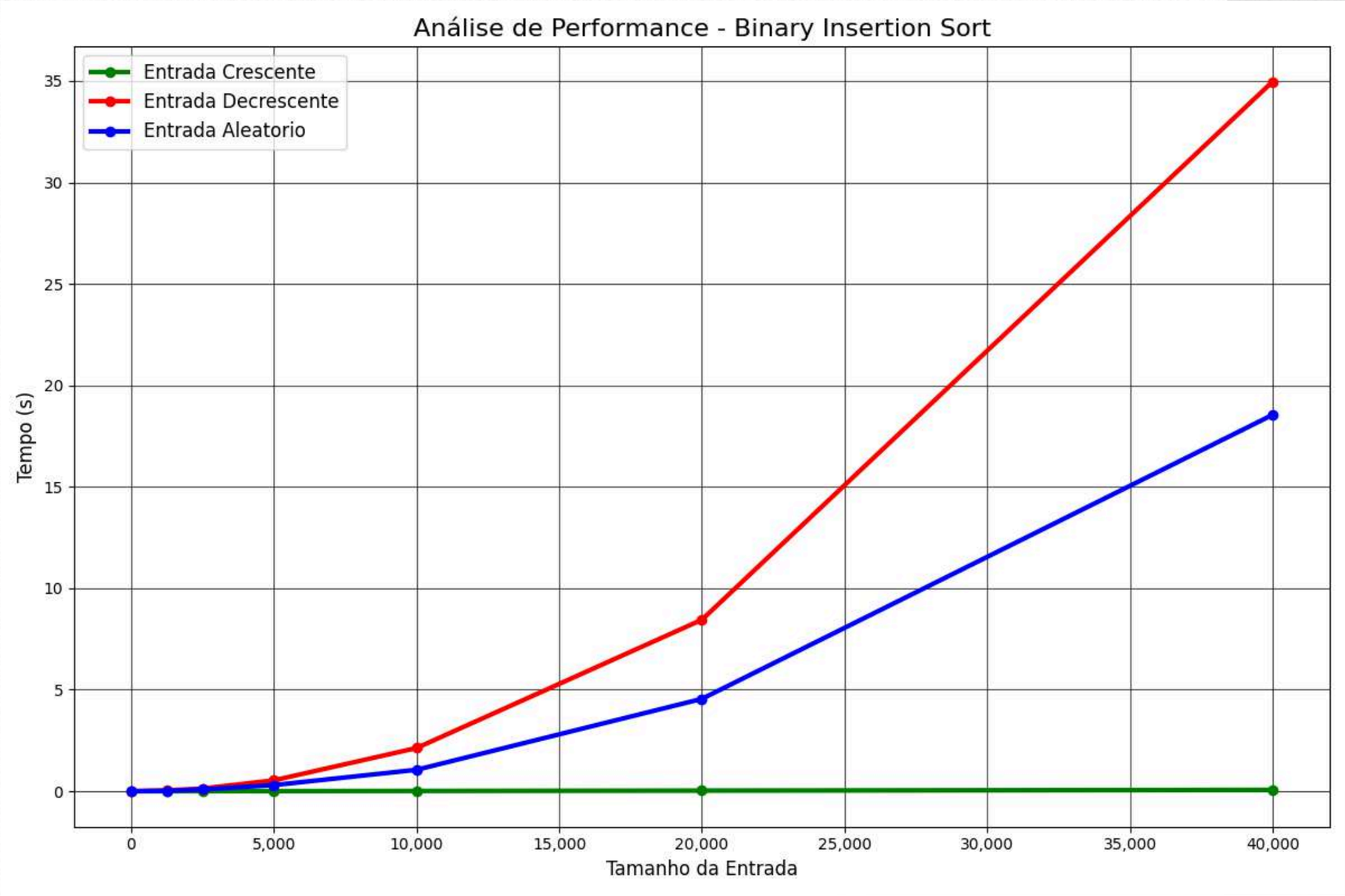
Quantidade de execuções: 30

GRÁFICOS DE TEMPO

INSERTION

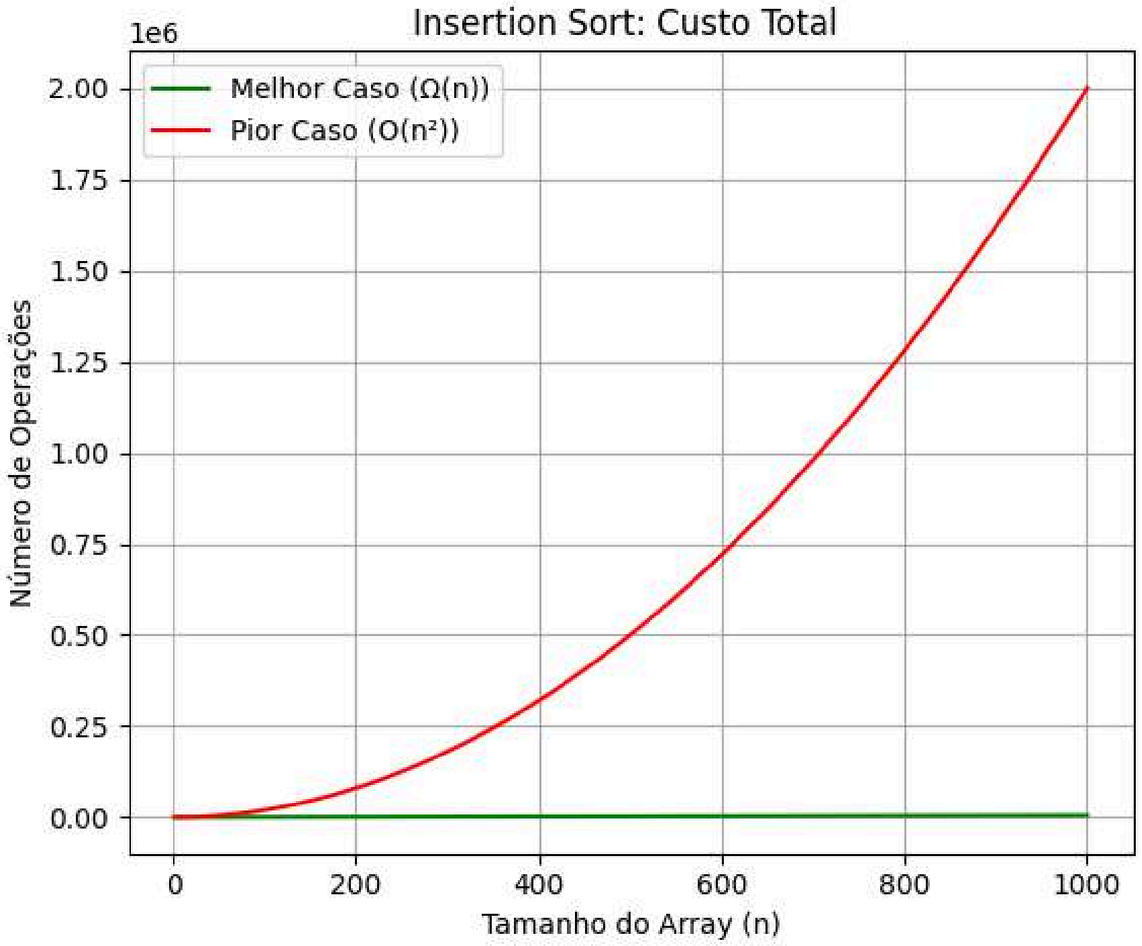

Análise de Performance - Insertion Sort
Entrada Crescente
Entrada Decrescente
Entrada Aleatorio
Tempo (s)
0
10
20
30
40
50
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
Tamanho da Entrada

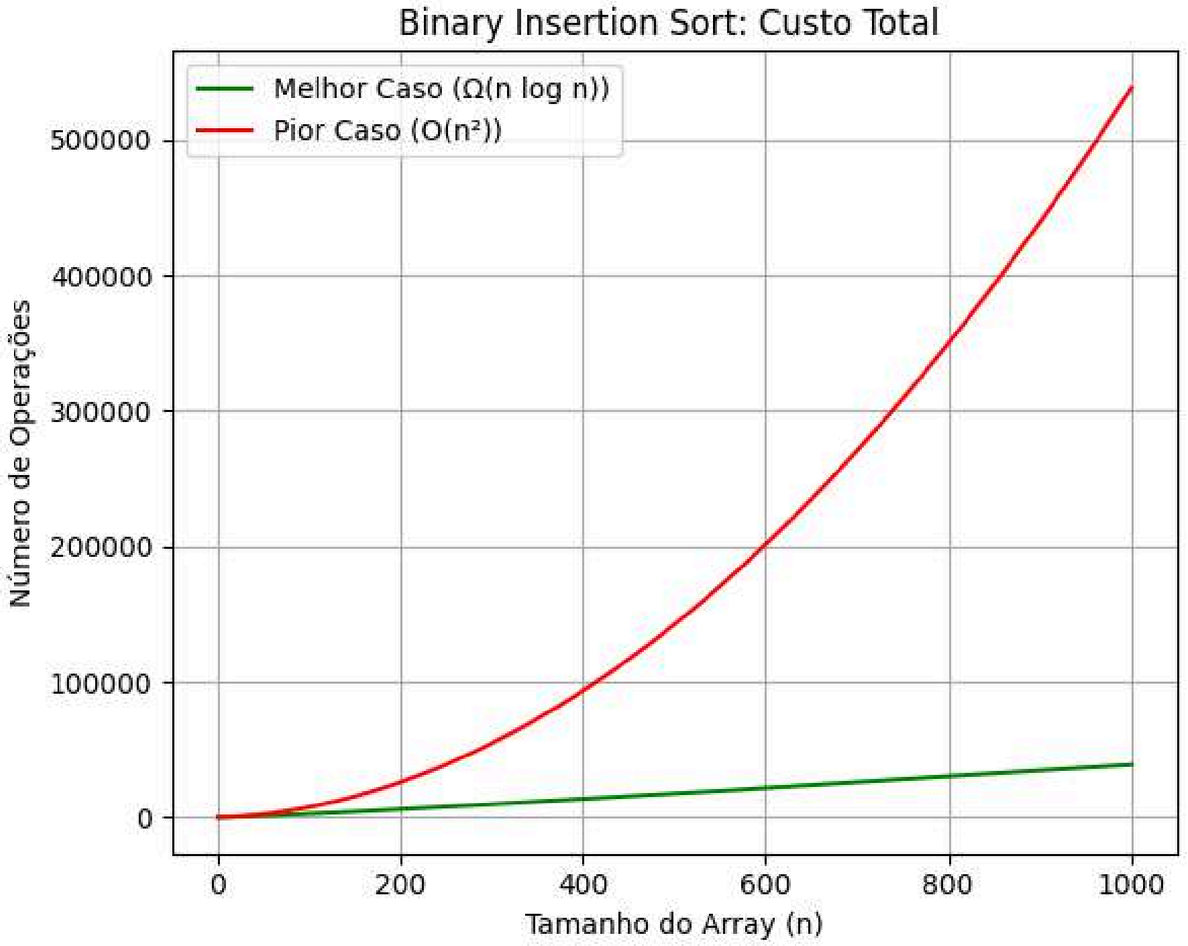
BINARY INSERTION



GRÁFICOS DE CUSTO

GRÁFICOS DE CUSTO





CONCLUSÃO

CONCLUSÃO

Com isso, concluímos que o Binary Insertion Sort se mostra mais eficiente, pois, embora tenha a mesma complexidade assintótica do Insertion Sort tradicional, reduz o número de comparações na busca da posição de inserção. Essa vantagem é mais evidente em vetores grandes e desordenados, onde a economia de comparações melhora o desempenho, tornando-o mais eficaz na prática mesmo mantendo o mesmo custo de deslocamentos.

REFERÊNCIAS

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos: teoria e prática. 3. ed. Rio de Janeiro: Elsevier, 2012.

SZWARCFITER, Jayme Luiz; LOPES, Liliane Maria da Silva. Estruturas de dados e seus algoritmos. 2. ed. Rio de Janeiro: LTC, 2010.

SIMIC, Milos. Binary Insertion Sort. Baeldung: Computer Science, 18 mar. 2024. Disponível em: https://www.baeldung.com/cs/binary-insertion-sort. Acesso em: [16/09/2025].