BUSCA BINÁRIA

DISCIPLINA: ANÁLISE DE ALGORITMOS
PROF. DR. HERBERT OLIVEIRA ROCHA
ALUNOS: EDUARDO IZIDORIO E SHELLY LEAL

BUSCA BINÁRIA

A busca binária é um algoritmo clássico utilizado para localizar rapidamente um elemento em um array ordenado.

Ela funciona dividindo o array ao meio a cada passo: compara o valor do meio com o alvo e descarta metade dos elementos, repetindo esse processo até encontrar o valor ou esgotar as possibilidades.

Isso faz com que a busca binária seja muito eficiente, principalmente em grandes conjuntos de dados.

BUSCA BINÁRIA

```
BUSCA_BINÁRIA_RECURSIVA(ARRAY, ALVO, INÍCIO, FIM, COMPARAÇÕES):
  COMPARAÇÕES ← COMPARAÇÕES + 1
  SE FIM ≥ INÍCIO:
    MEIO ← (INÍCIO + FIM) // 2
    SE ARRAY[MEIO] == ALVO:
      RETORNA MEIO
    SENÃO SE ARRAY[MEIO] < ALVO:
      RETORNA BUSCA_BINÁRIA_RECURSIVA(ARRAY, ALVO, MEIO + 1, FIM, COMPARAÇÕES)
    SENÃO:
      RETORNA BUSCA_BINÁRIA_RECURSIVA(ARRAY, ALVO, INÍCIO, MEIO – 1, COMPARAÇÕES)
  RETORNA –1
```

FUNÇÃO DE CUSTO – BUSCA BINÁRIA

- A cada passo, faz uma comparação e divide o array ao meio
- Função de custo (recorrência):

$$T(n) = T(n/2) + c$$

EXPANSÃO DA RECORRÊNCIA

$$T(n) = T\left(\frac{n}{2}\right) + c$$

$$\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + c \\
&= \left[T\left(\frac{n}{4}\right) + c\right] + c \\
&= T\left(\frac{n}{4}\right) + 2c \\
&= \left[T\left(\frac{n}{8}\right) + c\right] + 2c \\
&= T\left(\frac{n}{8}\right) + 3c \\
&\vdots \\
&= T\left(\frac{n}{2^k}\right) + k \cdot c
\end{aligned}$$

FUNÇÃO DE CUSTO FINAL E COMPLEXIDADE

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\log_2 n = k$$

$$T(n) = T(1) + c \cdot \log_2 n$$

$$T(n) = O(\log n)$$

- Busca binária é eficiente, mas...
- Se os dados são bem distribuídos, será que dá para “adivinhar” melhor onde está o elemento?
- **A busca por interpolação** tenta “estimar” a posição do valor, ao invés de sempre ir para o meio.

BUSCA POR INTERPOLAÇÃO

A busca por interpolação é um algoritmo para encontrar um elemento em um array ordenado.

Ela é parecida com a busca binária, mas, em vez de sempre ir para o meio do array, ela tenta estimar a posição onde o valor procurado pode estar, usando uma fórmula baseada no valor do elemento procurado e nos valores dos extremos do array.

BUSCA POR INTERPOLAÇÃO

```
BUSCA_INTERPOLACAO(ARRAY, ALVO):
    INÍCIO ← 0
    FIM ← TAMANHO(ARRAY) – 1
    ENQUANTO INÍCIO ≤ FIM E ALVO ≥ ARRAY[INÍCIO] E ALVO ≤ ARRAY[FIM]:
        SE INÍCIO == FIM:
            SE ARRAY[INÍCIO] == ALVO:
                RETORNA INÍCIO
            RETORNA –1
        POS ← INÍCIO + ((ALVO – ARRAY[INÍCIO]) × (FIM – INÍCIO)) // (ARRAY[FIM] – ARRAY[INÍCIO])
        SE ARRAY[POS] == ALVO:
            RETORNA POS
        SENÃO SE ARRAY[POS] < ALVO:
            INÍCIO ← POS + 1
        SENÃO:
            FIM ← POS – 1
    RETORNA –1
```

FUNÇÃO DE CUSTO E COMPLEXIDADE

FUNÇÃO DE CUSTO (ESPERADA PARA DADOS UNIFORMES):

$$T(n) = T(n - f(n)) + c$$

Onde f(n) representa o “salto” estimado

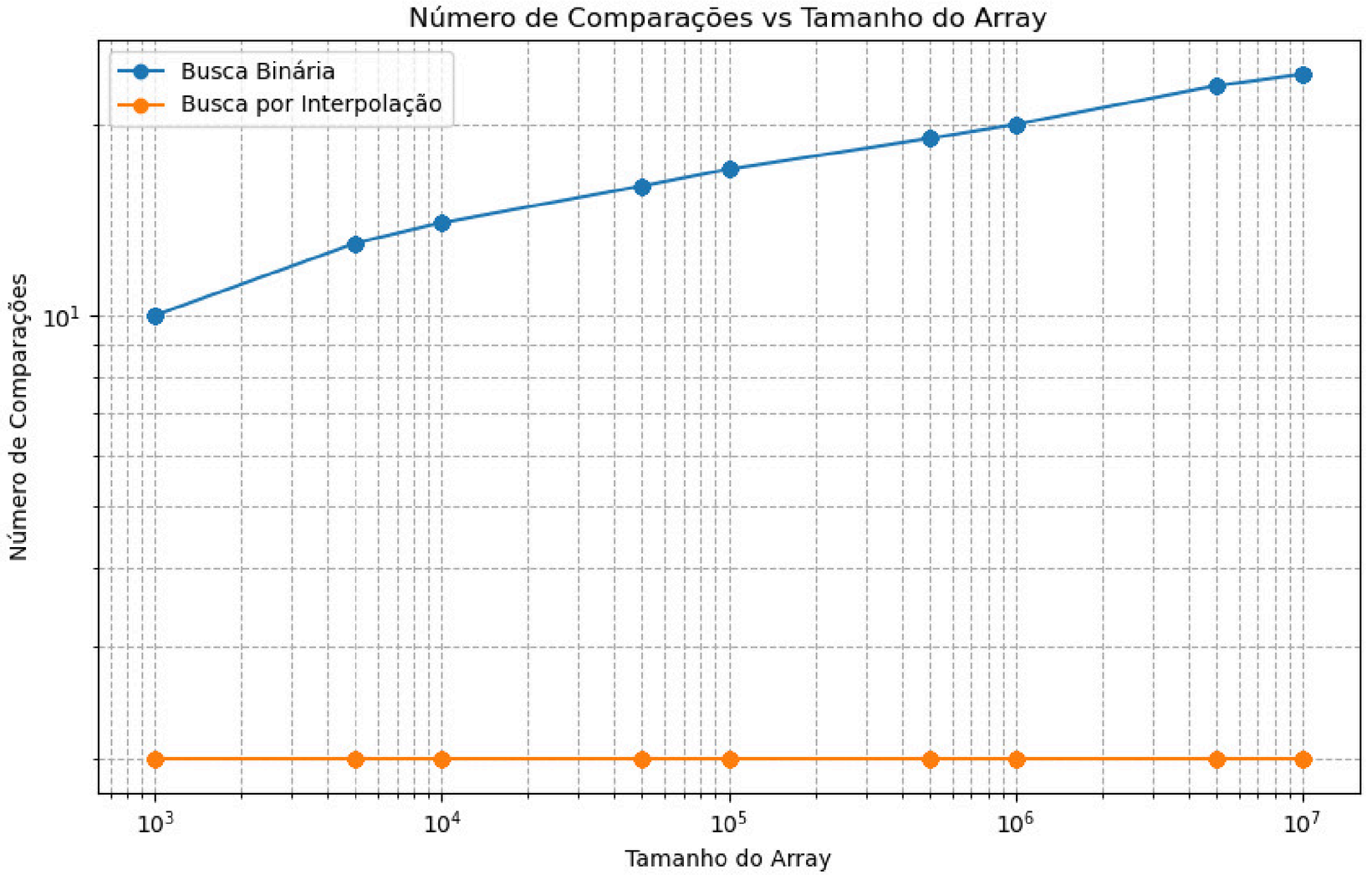
Complexidade esperada: O (log log n)

Pior caso: O (n)

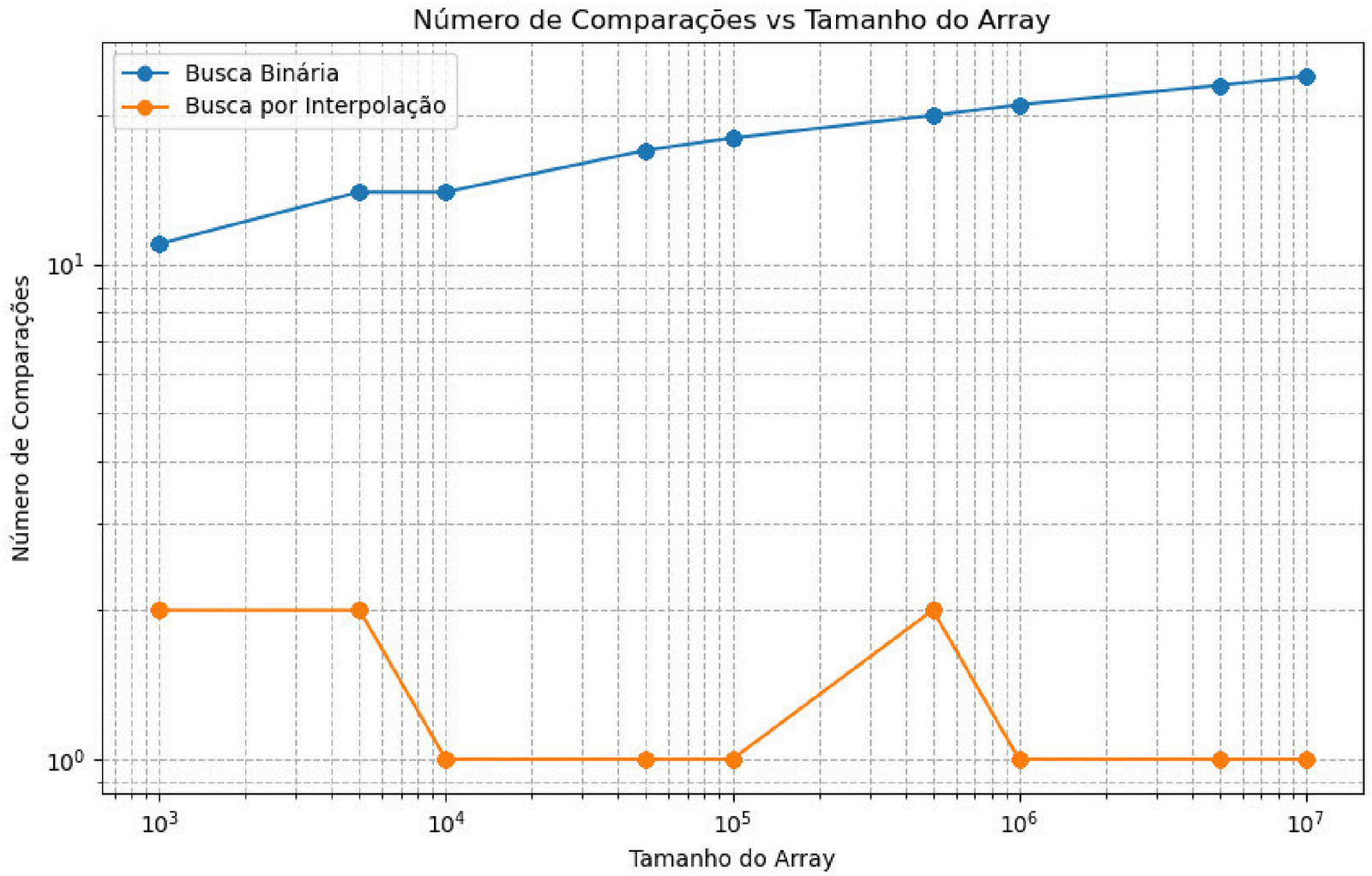
METODOLOGIA EXPERIMENTAL E BENCHMARK

- **Geração dos dados:**
  - **Arrays de diferentes tamanhos: 10, 100, 1.000, 10.000, 100.000, 1.000.000, 5.000.000, 10.000.000**
- **Tipos de distribuição:**
  - **Números uniformes e ordenados**
- **Execução dos testes:**
  - **Cada algoritmo busca o último elemento do array (pior caso)**
  - **Cada experimento repetido 20 vezes para cada tamanho e tipo de array**
- **Métricas avaliadas:**
  - **Número de comparações realizadas**

Arrays Ordenados



Arrays Uniformes



QUANDO USAR BUSCA POR INTERPOLAÇÃO?

- Dados ordenados e uniformemente distribuídos (ex:dados sequenciais, planilhas financeiras, etc.)
- Para outros tipos de distribuição, a busca binária costuma ser mais confiável.

BUSCA BINÁRIA X BUSCA POR INTERPOLAÇÃO

Algoritmo	Recorrência	Complexidade Esperada	Pior Caso
Busca Binária	$T(n)=T(n/2) + c$	$O(\log n)$	$O(\log n)$
Interpolação	$T(n)=T(n - f(n)) + c$	$O(\log\log n)$	$O(n)$

OBRIGADO
PELA ATENÇÃO