

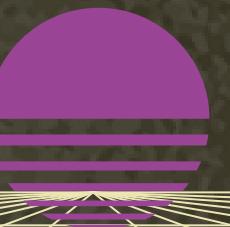


The logo consists of the word "GNOME" in a large, bold, black sans-serif font above the word "SORT" in a slightly smaller, bold, black sans-serif font. Both words are centered horizontally. A thin horizontal line with a small gap in the middle connects the two words.

GNOME

SORT

Algoritmos e Estrutura de Dados I



Discentes:

Douglas Patrick Cardozo Freitas

Enthony Araujo de Oliveira

João Vitor Ribeiro Pacó

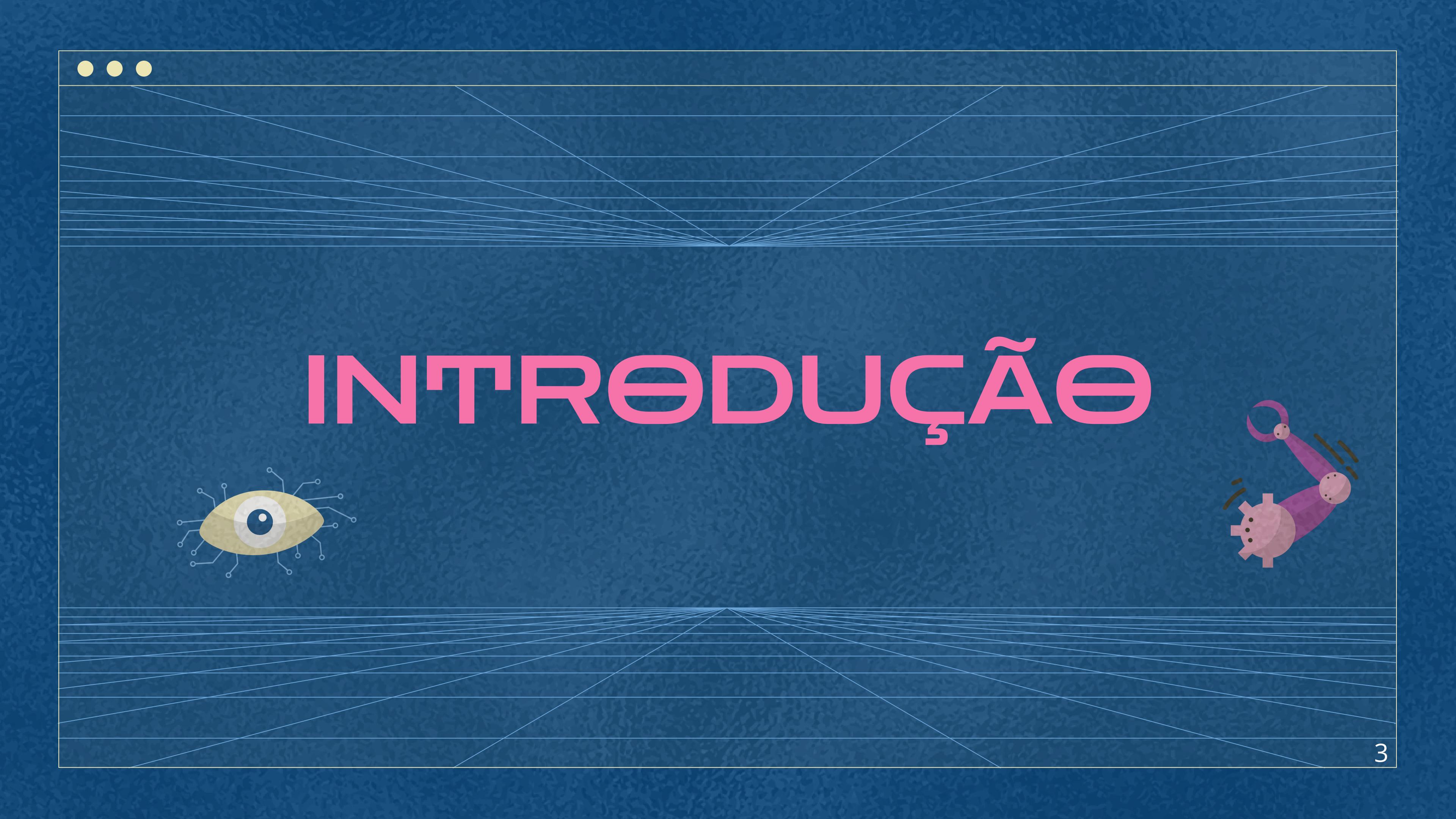
José Augusto de Souza Costa

Pablo Rubens de Moura

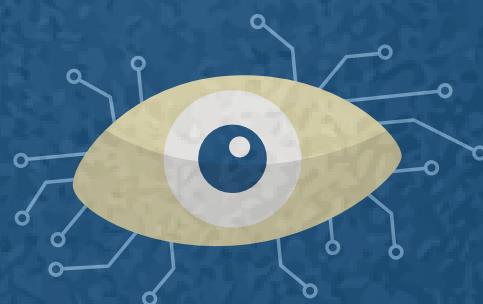
Docente:

George Felipe Fernandes Vieira





INTRODUÇÃO



O que é?

O Gnome Sort, também conhecido como Stupid Sort, é um algoritmo de ordenação por comparação que funciona de forma extremamente simples e intuitiva, organizando elementos de um array em ordem crescente ou decrescente sem a necessidade de estruturas auxiliares complexas. Ele opera diretamente no próprio array (in-place), comparando sempre pares de elementos adjacentes, e sua mecânica é inspirada em uma metáfora encantadora: um pequeno gnomo jardineiro que caminha pelo jardim arrumando vasos de flores.

Imagine uma fileira de vasos com flores de alturas diferentes – cada vaso representa um valor no array. O gnomo começa na posição zero e compara o vaso atual com o próximo. Se o vaso da frente for menor, ele troca os dois de lugar e dá um passo para trás, para verificar se a troca não desorganizou algo antes. Caso contrário, se tudo estiver em ordem, ele simplesmente avança um passo. Esse movimento continua até que o gnomo percorra todo o array e volte ao início sem precisar fazer mais trocas.

Criador

Criado pelo cientista da computação iraniano Hamid Sarbazi-Azad em 2000, o algoritmo recebeu o nome "Gnome Sort" exatamente por causa dessa analogia com o duende do jardim, que organiza os vasos com paciência e atenção aos detalhes. O apelido "Stupid Sort" é uma brincadeira irônica: embora pareça ingênuo ou até "burro" por não usar otimizações sofisticadas, ele é funcional, fácil de entender e surpreendentemente eficiente em certos cenários, como quando o array já está quase ordenado.



Desempenho computacional

Melhor caso:

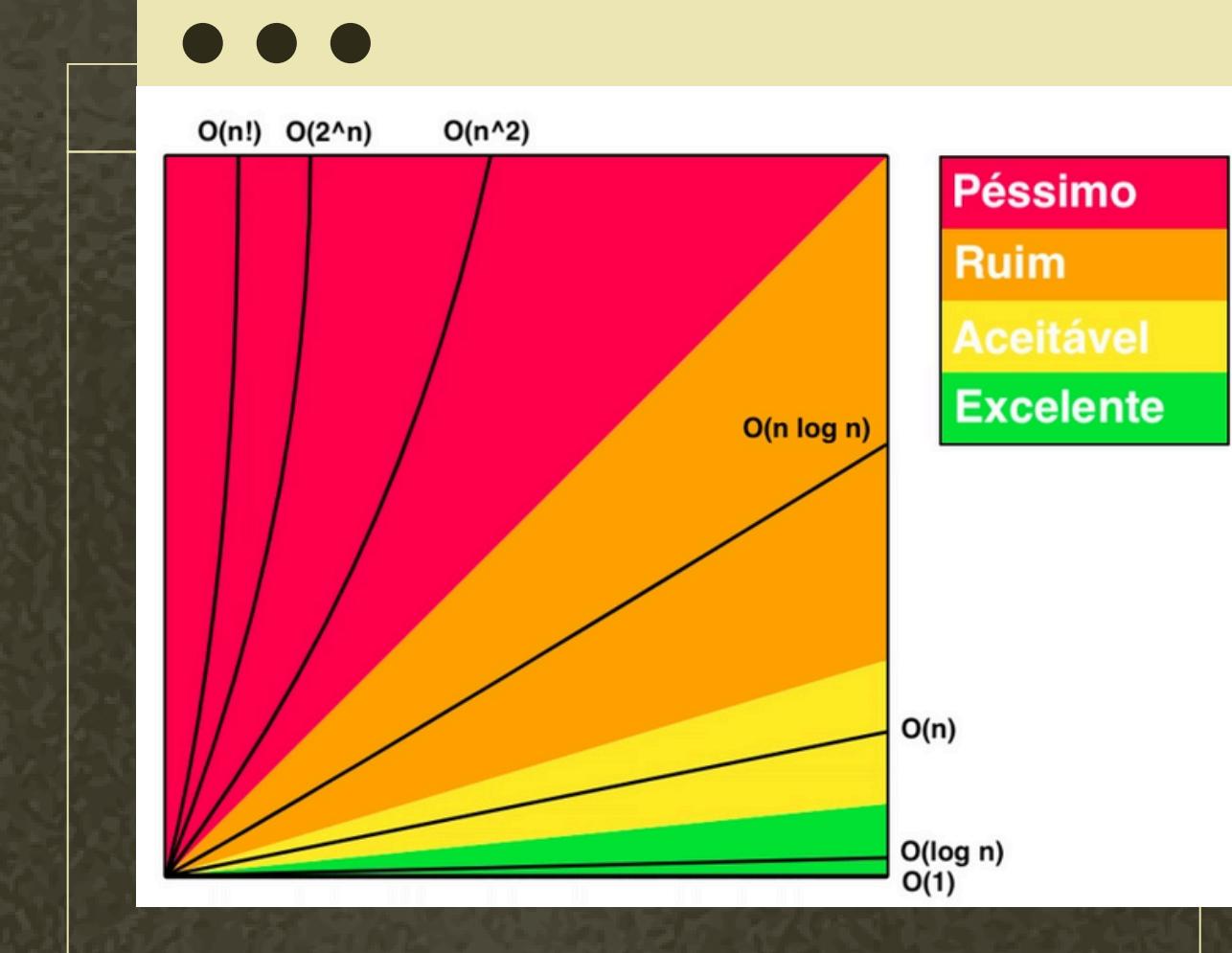
$$\Theta(n)$$

Médio caso:

$$\Theta(n^2)$$

Pior caso::

$$\Theta(n^2)$$



Regras do algoritmo

Regra 1: O gnomo sempre começa na posição \emptyset do array (índice pos = \emptyset).

Regra 2: Em cada passo, o gnomo olha para o vaso atual (array[pos]) e o vaso anterior (array[pos - 1]), mas apenas se ainda houver um próximo vaso.

Regra 3: Se array[pos] > array[pos - 1], avança (pos++), senão troca e volta (pos--); .

Regra 4: Continue até o jardim estar perfeito. O gnomo nunca para enquanto não terminar o trabalho. Ele repete as regras 2 e 3 até que chegue ao final do array (pos == tamanho), volte ao início (pos = \emptyset) e não precise mais trocar nada.





Pseudo-código



Algorithm 1 Gnome Sort

Require: Array $A[0 \dots n - 1]$ (índices 0-based), inteiro n (tamanho de A)

Ensure: A ordenado em ordem não decrescente

```
1: procedure GNOMESORT( $A, n$ )
2:    $pos \leftarrow 0$ 
3:   while  $pos < n$  do
4:     if  $pos = 0$  or  $A[pos] \geq A[pos - 1]$  then
5:        $pos \leftarrow pos + 1$ 
6:     else
7:       SWAP( $A[pos], A[pos - 1]$ )                                 $\triangleright$  troca os elementos
8:        $pos \leftarrow pos - 1$ 
9:     end if
10:   end while
11: end procedure
```





VANTAGENS E DESVANTAGENS



Vantagens



Implementação simples:

O algoritmo é fácil de entender e implementar, tornando-se uma boa opção para fins didáticos e situações em que a simplicidade é prioritária



Estável:

Mantém a ordem relativa entre elementos iguais, o que pode ser importante em alguns contextos.



Bom desempenho para listas pequenas ou quase ordenadas:

Para vetores pequenos ou já parcialmente ordenados, pode ser eficiente, realizando poucas trocas



Desvantagens



Baixo desempenho para listas grandes:

Θ número de comparações e trocas cresce rapidamente com o aumento do tamanho da lista, tornando-o inviável para grandes volumes de dados por conta da complexidade quadrática $\Theta(n^2)$



Muitas movimentações:

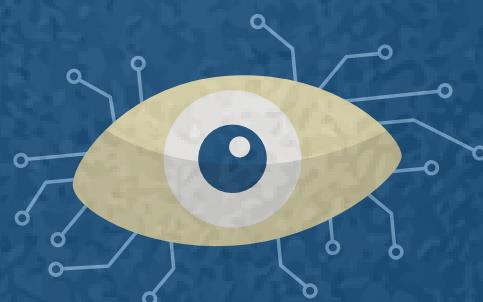
Θ algoritmo efetua uma sequência grande de trocas, especialmente em listas desordenadas, o que pode levar à lentidão.



Pouco utilizado em aplicações reais:

Devido à ineficiência para grandes conjuntos de dados, raramente é escolhido em aplicações práticas – algoritmos como Quick Sort e Merge Sort são preferidos.

IMPLEMENTAÇÃO



Implementação

● ● ● Função de troca dos elementos

```
void swap(int arr[], int i){  
    int temp = arr[i];  
  
    arr[i] = arr[i-1];  
    arr[i-1] = temp;  
}
```

● ● ● Função GnomeSort

```
void gnomeSort(int arr[], int n) {  
    int i = 0;  
  
    while (i < n)  
    {  
        if((i == 0) || (arr[i] ≥ arr[i-1])){  
            i++;  
        }  
        else  
        {  
            swap(arr, i);  
            i--;  
        }  
    }  
}
```

Implementação

- ● ● Função para mostrar o array

```
void printArray(int arr[], int n){  
    printf("\n");  
    printf("Lista organizada com Gnome sort: ");  
    for (int i = 0; i < n; i++)  
    {  
        printf("%d ", arr[i]);  
    }  
}
```

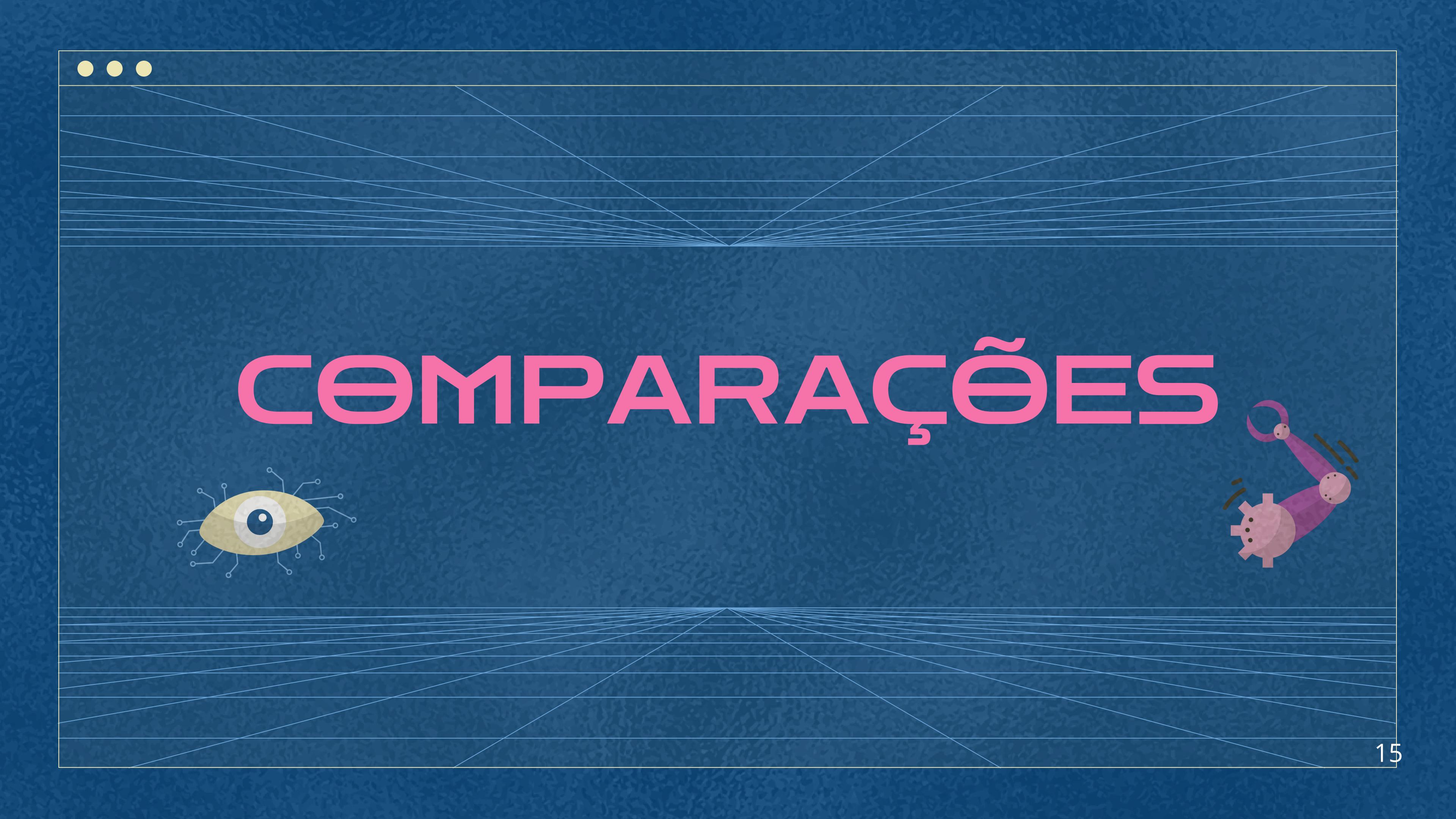
1

Implementação

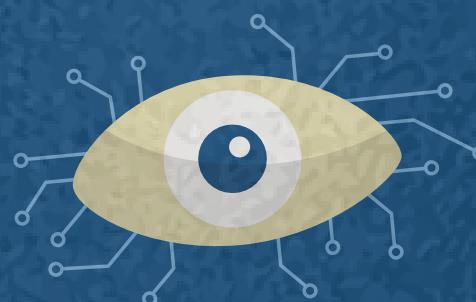
● ● ● Função main

```
int main(){
    printf("GnomeSort\n");
    int arr[] = {11,44,89,2025,-1,3,1999,-2000};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Lista original: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    gnomeSort(arr, n);
    printArray(arr,n);
    printf("\n");

    return 0;
}
```

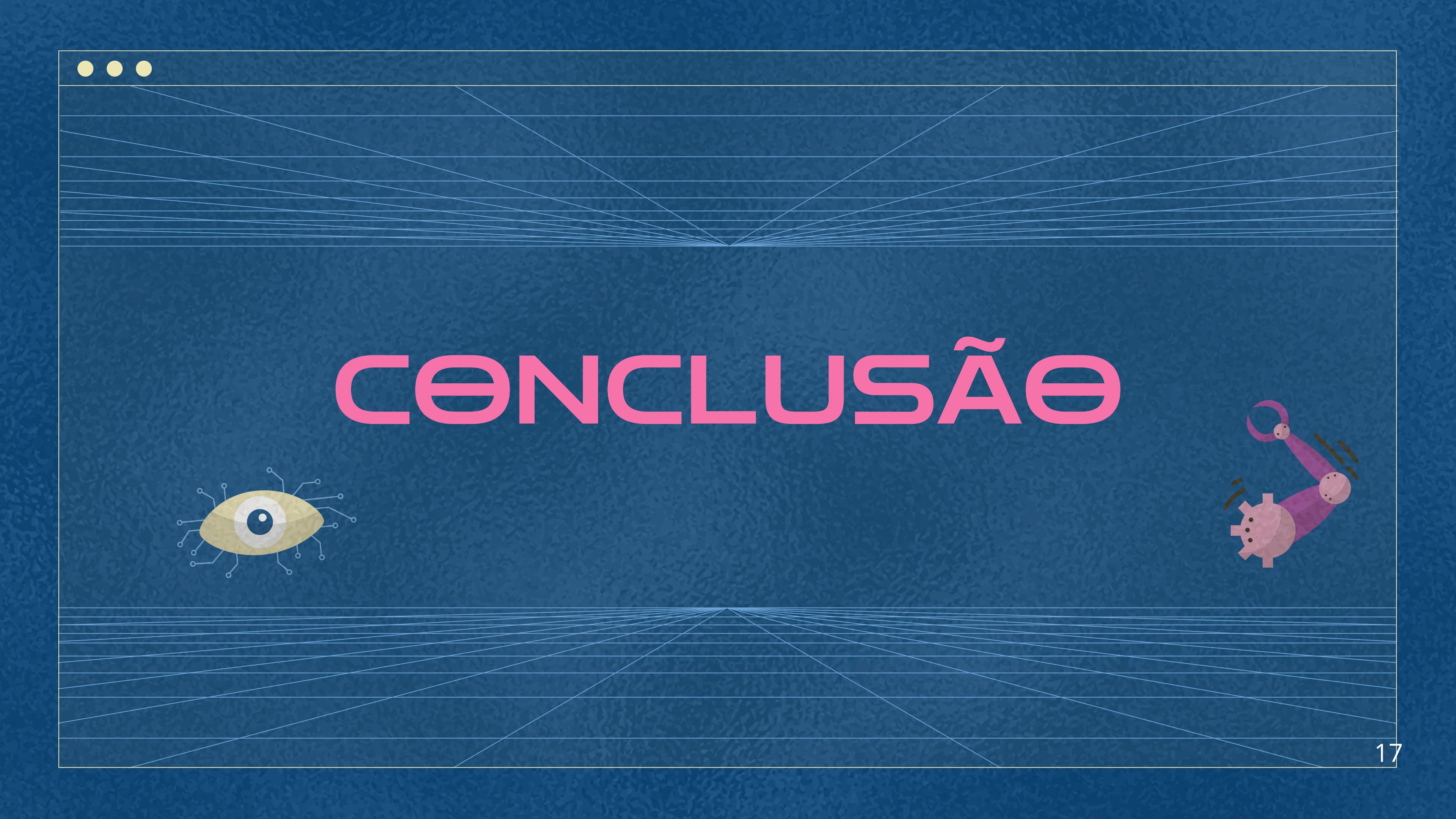


COMPARAÇÕES

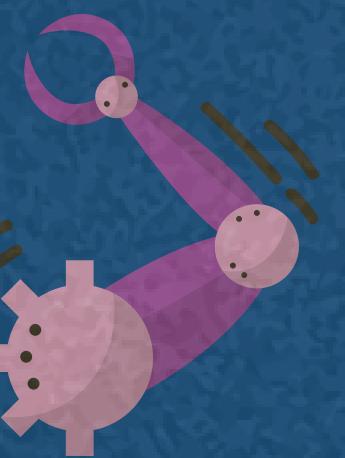
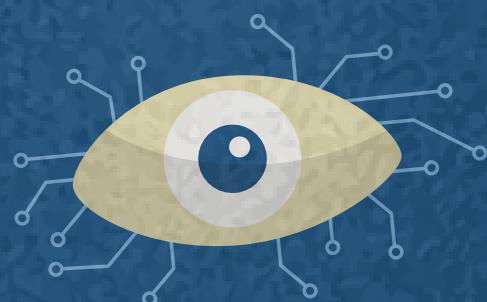


Comparações

Posição	Algoritmo	Complexidade Média	Observações
1º	Quick Sort	$O(n \log n)$	Rápido na prática, usado amplamente na indústria
2º	Merge Sort	$O(n \log n)$	Estável, eficiente para grandes volumes
3º	Heap Sort	$O(n \log n)$	Desempenho consistente, sem muita memória extra
4º	Shell Sort	Entre $O(n)$ e $O(n^{1.5})$	Algoritmo híbrido, mais rápido que inserção para listas médias
5º	Insertion Sort	$O(n^2)$	Eficiente em listas pequenas e quase ordenadas
6º	Selection Sort	$O(n^2)$	Simples, mas mais lento que inserção
7º	Bubble Sort	$O(n^2)$	Muito lento, só usado para propósitos didáticos
8º	Gnome Sort	$O(n^2)$	Simples, mas ineficiente para grandes dados



CONCLUSÃO





Conclusão

Θ Gnome Sort é um algoritmo simples e intuitivo, fácil de entender e implementar, o que o torna uma excelente ferramenta didática para introdução a algoritmos de ordenação. Ele é particularmente eficiente para listas pequenas ou quase ordenadas, pois seu melhor caso tem complexidade linear $\Theta(n)$.

Por outro lado, seu desempenho piora significativamente em listas grandes ou muito desordenadas, apresentando complexidade quadrática $\Theta(n^2)$. Isso o torna pouco prático para aplicações que demandam alto desempenho em grandes volumes de dados.

ΘBRIGADE



Você sabia? A última vez que o Vasco jogou uma semifinal de Libertadores, o juiz ainda apitava com o dedo na boca

