

# RADIXSORT GRUPO 4

ESTRUTURA DE DADOS I

# SUMARIO

- 01** LÓGICA DE FUNCIONAMENTO
- 02** APLICAÇÕES PRÁTICAS
- 03** COMPLEXIDADE DOS ALGORITMOS
- 04** ANÁLISE DE COMPLEXIDADE
- 05** VANTAGENS E DESVANTAGENS
- 06** FUNCIONAMENTO EM CODIGO
- 07** DISCUSSÃO COMPARATIVA

# RADIX SORT

Radix sort é um algoritmo de **ordenação não comparativa** que organiza dados com base na representação posicional dos números.

Ordenações não comparativas são algoritmos de ordenação que não dependem de comparações diretas entre os elementos para determinar a ordem.



# LÓGICA DO FUNCIONAMENTO

Ele começa pelo dígito menos significativo e segue até o mais significativo, classificando os elementos em cada etapa de acordo com o valor do dígito atual.

Ao finalizar o processamento de todos os dígitos, os dados estão totalmente ordenados. Esse método é eficiente para conjuntos com um número fixo de dígitos e é muito usado em aplicações onde a velocidade é crucial.

# LÓGICA DO FUNCIONAMENTO

## Unidade

Na primeira iteração, organiza os números com base no dígito das unidades (o último dígito de cada número).

## Dezenas

Na segunda iteração, organiza a lista reordenada de acordo com o dígito das dezenas.

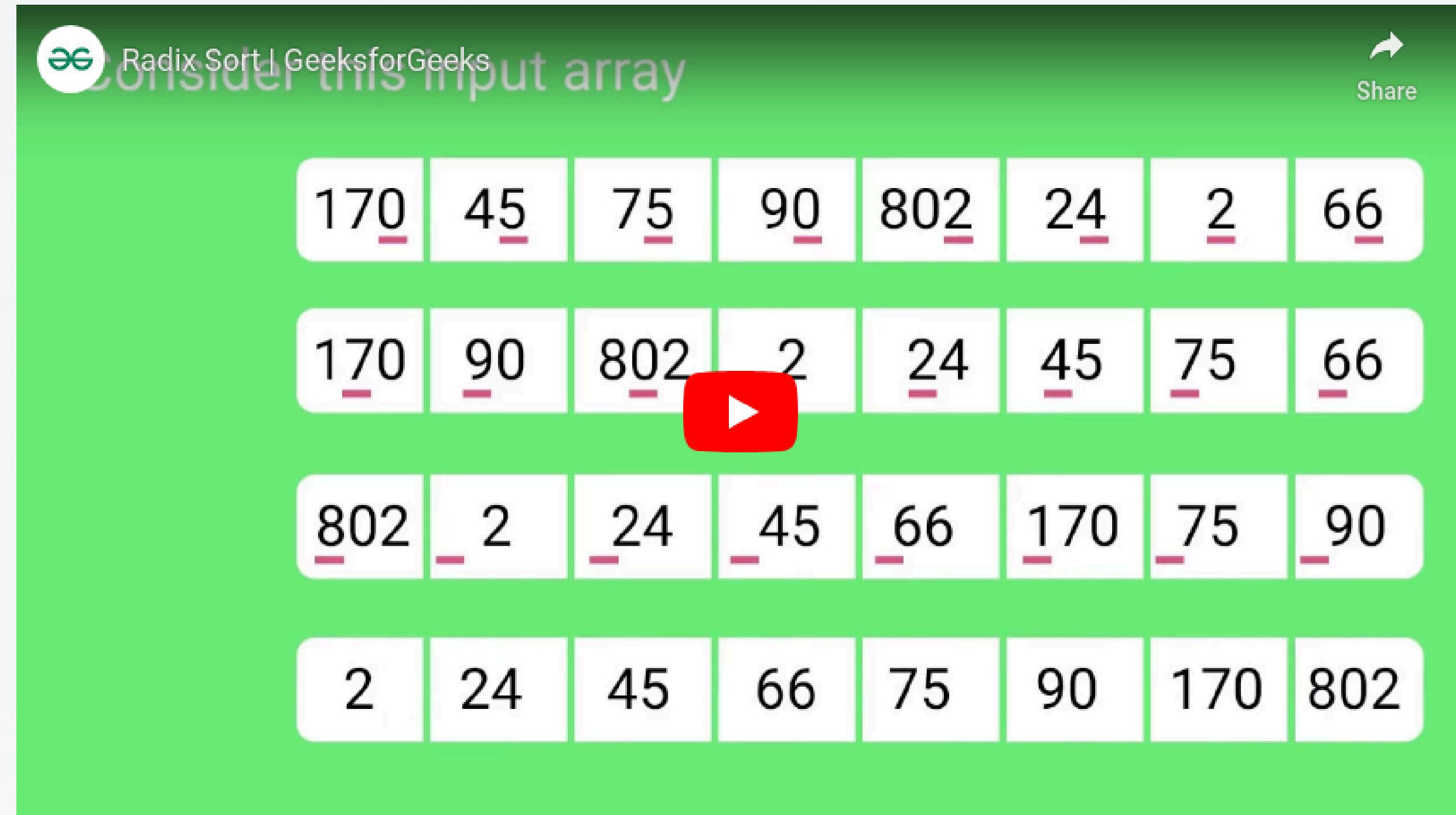
## Centenas

Na terceira iteração e a diante, os números são ordenados com base no dígito das centenas e a diante.

882	003	005	345	254	606
588	808	535	784	715	710



# VIDEO DE REFERÊNCIA

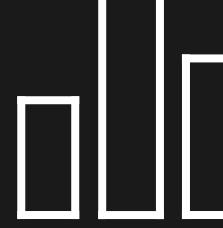


# APLICAÇÕES PRÁTICAS



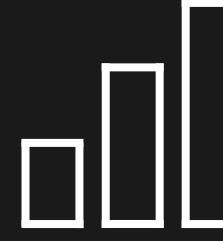
Para ordenar registros em bancos de dados, especialmente quando os dados são inteiros ou strings de tamanho fixo.

BANCO DE DADOS



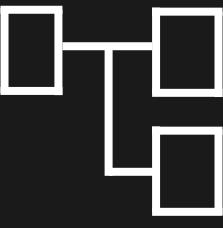
Para classificar pacotes de dados com base em seus endereços IP ou outros campos.

REDES



Para classificar grandes conjuntos de dados para fins de análise e processamento.

ANÁLISE DE DADOS



Para ordenar dados para representações visuais, como gráficos e tabelas.

VISUALIZAÇÃO

# COMPLEXIDADE DOS ALGORITMOS

- n = Número de elementos do array
- d = Número de dígitos do maior número (ou bits)
- b = Sistema numérico usado para a ordenação(decimal = 10, binária = 2, hexadecimal = 16)

Essa complexidade vem do fato de que o algoritmo processa cada dígito dos números, realizando uma contagem (Counting Sort) para cada dígito.

- Complexidade de espaço é:  $O(n+b)$ .
- Complexidade total do radix é:  $O(d(n+b))$ .

# COMPLEXIDADE DOS ALGORITMOS

53	89	150	36	633	233
150	53	633	233	36	89
633	233	36	150	53	89
36	53	89	150	233	633

$$n = 6$$

$$d = 3$$

$$b = 10$$

$$O(3(6+10))$$

# ANÁLISE DE COMPLEXIDADE

Melhor caso	$O(nk)$
Pior caso	$O(nk)$
Caso Médio	$O(nk)$

# VANTAGENS E DESVANTAGENS

## Vantagem

- Algoritmo estável;
- Tempo de execução linear para dados uniformemente distribuídos;
- Fácil de entender e implementar.

## Desvantagem

- Requer espaço de armazenamento adicional para os baldes;
- Pode ser lento para conjuntos de dados pequenos.



# **FUNCIONAMENTO EM CÓDIGO**

**LINK DO GITHUB:**

[https://github.com/ktoJun/ESTDI\\_GRUPO4\\_RADIXSORT/tree/main/C%C3%B3digo](https://github.com/ktoJun/ESTDI_GRUPO4_RADIXSORT/tree/main/C%C3%B3digo)

# OBRIGADO PELA PRESença DE TODOS

.S.  
uvv

## Nome dos Integrantes:

- Arthur Oliveira Rueda
- Erick Jun Kato
- Giovanni Persio Gonçalves
- Igor Scalzer Ratti
- Kuan Modolo Carriço
- Nathalia Bertordo Alberto
- Ramses de Oliveira Martins

# REFERÊNCIAS

- 01** A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN. \*DATA STRUCTURES AND ALGORITHMS\*. ADDISON-WESLEY, .
- 02** [HTTPS://YOUTU.BE/NU4GDUFABIM?T=21](https://youtu.be/nu4gdUFabIM?t=21)
- 03** [HTTPS://WWW.GEEKSFORGEEEKS.ORG/RADIX-SORT/](https://www.geeksforgeeks.org/radix-sort/)
- 04** [HTTPS://EN.WIKIPEDIA.ORG/WIKI/RADIX\\_SORT](https://en.wikipedia.org/wiki/Radix_sort)
- 05** [HTTPS://QIITA.COM/YP2211/ITEMS/683FA423B3F40A871064](https://qiita.com/yp2211/items/683fa423b3f40a871064)
- 06** -
- 07** -