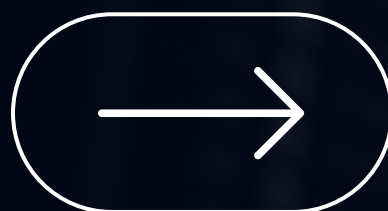


# Ordenação paralela de dados.

EM PYTHON

ALUNOS

KAIO RECK  
FELIPE DE MACEDO  
RYAN BARCELOS





# COMO FUNCIONA O PROGRAMA DE ORDENAÇÃO?

O programa solicita ao usuário que insira o nome do arquivo que contém os dados a serem ordenados. Também solicita ao usuário que insira o número de threads que deseja. Em seguida o programa realiza a execução e entrega em outros arquivos os resultados.

# PROGRAMA:

```
Digite o nome do arquivo que contém os dados a serem ordenados: numerosale.txt
Gostaria de executar este código? (sim/não): sim
Quantas threads você gostaria de usar para ordenar os dados? 2
Os dados ordenados usando threads foram salvos em um arquivo.
Os dados ordenados sem threads foram salvos em um arquivo.
O tempo de execução da abordagem paralela foi de: 0.14049983024597168
```

# Principais funções e importações

## FUNÇÕES

1. QUICKSORT
2. PIVOT
3. OPEN(FILE, MODE)

## IMPORTS

4. CONCURRENT.FUTURES
5. IMPORT THREADING
6. IMPORT TIME

## “ O QUE É O QUICKSORT? ”

É um algoritmo de ordenação muito utilizado devido à sua eficiência em média e melhor caso. Ele funciona dividindo a lista em sub-listas menores, com base em um elemento pivot, e recursivamente ordenando essas sub-listas.


## “ O QUE É O QUICKSORT? ”

**Utilizamos Quicksort para fazer a ordenação de dados**

```
7 def quick_sort(arr):  
8     # Se a lista estiver  
ordenar  
9     if len(arr) <= 1:  
10    return arr
```

Na abordagem paralela, a função quick\_sort é chamada dentro da função ordenar\_dados, que é então executada em cada thread para ordenar uma parte dos dados

```
# Função para ordenar dados em uma thread  
def ordenar_dados(dados):  
    return quick_sort(dados)
```





## *O QUE É O PIVOT ?*



Divide a lista em duas partes durante cada etapa do processo de ordenação no algoritmo "quicksort". Este processo de divisão e ordenação recursiva continua até que toda a lista esteja ordenada.

“

## O QUE É O PIVOT?

”

```
12 # Seleciona um número central da lista como pivô
13 pivot = arr[len(arr) // 2]
14
15 # Divide a lista em três partes: menores, iguais e maiores que o pivô
16 esquerda = [x for x in arr if x < pivot]
17 meio = [x for x in arr if x == pivot]
18 direita = [x for x in arr if x > pivot]
19
20 # Combina recursivamente as listas ordenadas (menores, iguais,
    maiores) e retorna o resultado
21 return quick_sort(esquerda) + meio + quick_sort(direita)
22
```





## ***O QUE É O PIVOT?***



O pivot é escolhido como o elemento que está na posição central da lista (arr).  
Esse método de escolha do pivô é uma abordagem comum e geralmente eficiente.

Depois que o pivô é escolhido, a lista é particionada em três partes:

Elementos menores que o pivot.

Elementos iguais ao pivot.

Elementos maiores que o pivot.

Essa divisão é realizada para que os elementos menores que o pivot sejam movidos para a esquerda dele e os elementos maiores sejam movidos para a direita, enquanto os elementos iguais permanecem no lugar.

## “E COMO É FEITA A LEITURA DOS DADOS DOS ARQUIVOS?”

É utilizada a função ‘open()’ embutida do Python que é usada para abrir arquivos. Ela recebe argumentos: o nome do arquivo ‘nome\_arquivo’ a ser aberto.

```
# Lê os dados do arquivo
with open(nome_arquivo, "r") as arquivo:
    dados = [linha.strip() for linha in arquivo]
```

# Importações utilizadas no código



## Concurrent.futures

Permite que você execute funções de forma assíncrona usando pools de threads e processos. É utilizado para criar uma ThreadPoolExecutor

## Time

É utilizada para medir o tempo de execução das diferentes abordagens de ordenação (paralela e sequencial)

## Threading

Fornece funções para trabalhar com threads em Python. As threads são usadas para executar tarefas simultaneamente e são úteis para operações extensivas.