Trabalho de FPAA

# Máquina

Processador: i5-13600K / 5.10 GHz  
Memória RAM: 32GB DDR5 / 6000mhz  
OS: Windows 11

**Observação:** todo o código para realizar essa atividade está liberado publicamente no meu github, podendo fazer o acesso em: [vitorlany/fpaa-analise-algoritmo](https://github.com/vitorlany/fpaa-analise-algoritmo)

# Primeira Parte

Começamos com Bubble Sort, um algoritmo de complexidade , então observamos que a cada vez que aumentamos o tamanho da lista do caso de testes, mais ainda ele demora para executar, consumindo um tempo absurdo de execução.

Após partir para o algoritmo de Merge Sort, um algoritmo de , a resposta foi absurdamente mais rápida, chegando a fazer todo o processo que o Bubble Sort fez em 2 a 3 horas, em menos de 10 segundos.

Observação: a tabela está em Exel no arquivo

## Bubble Sort

Ao total, demorou cerca de 2 a 3 horas para ser finalizado.

### 62.500

A screenshot of a computer

Description automatically generated

### 125.000

A screenshot of a computer

Description automatically generated

### 250.000

A screenshot of a computer

Description automatically generated

### 375.000

A screenshot of a computer

Description automatically generated

### Resultado

Arquivo com todo o log do console



## Merge Sort

Primeira impressão que tive ao rodar o código, foi um susto ao perceber quão eficiente e rápido ele fez o mesmo trabalho que o Bubble Sort, em menos de 10 segundos, até mesmo em tamanhos gigantes como 375.000, chegando em alguns casos, quase instantâneo.

### 62.500

A screenshot of a computer

Description automatically generated

### 125.000

A screenshot of a computer

Description automatically generated

### 250.000

A screenshot of a computer

Description automatically generated

### 375.000

A screenshot of a computer

Description automatically generated

### Resultado

Arquivo com todo o log do console



# Segunda Parte

Após fazermos o teste, percebemos que o Quick Sort é muito eficaz quando temos uma lista não ordenada, chegando em 0-1ms em média, mas quando já temos uma execução consideravelmente maior, atingindo 27-30ms à cada lista.

### Aleatório

A screenshot of a computer

Description automatically generated

### Ordenado

A screenshot of a computer

Description automatically generated

### Resultado

Arquivo com todo o log do console

