

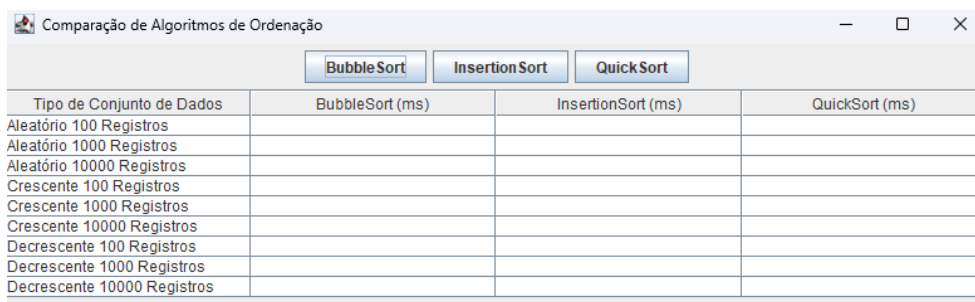
## Relatório de Análise de Algoritmos de Ordenação

A comparação entre os algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort mostra como cada um deles lida de forma diferente com os dados, o que afeta o tempo de processamento dependendo do tipo de conjunto de dados.

1. **Bubble Sort:** Este é o método de ordenação mais lento entre os três. No caso de dados desordenados ou decrescentes, ele demora muito para organizar os elementos, já que verifica e troca cada elemento várias vezes. Quando os dados já estão em ordem crescente, ele se torna um pouco mais rápido, mas ainda é menos eficiente que os outros algoritmos.
2. **Insertion Sort:** Embora seja mais rápido que o Bubble Sort, o Insertion Sort também não é muito eficiente com grandes volumes de dados desordenados. No entanto, ele lida melhor com conjuntos que já estão ordenados o que reduz o número de passos necessários para completar a ordenação. Assim, ele se torna mais eficiente em cenários onde os dados estão próximos da ordem desejada.
3. **Quick Sort:** Este é o algoritmo mais rápido dos três e é muito eficiente para lidar com grandes volumes de dados. Ele tem um desempenho consistente e superior, independentemente do tipo de ordenação inicial (aleatório, crescente ou decrescente). Isso se deve ao fato de ele dividir o conjunto de dados e ordenar partes menores, o que diminui o tempo necessário para processar cada conjunto de dados

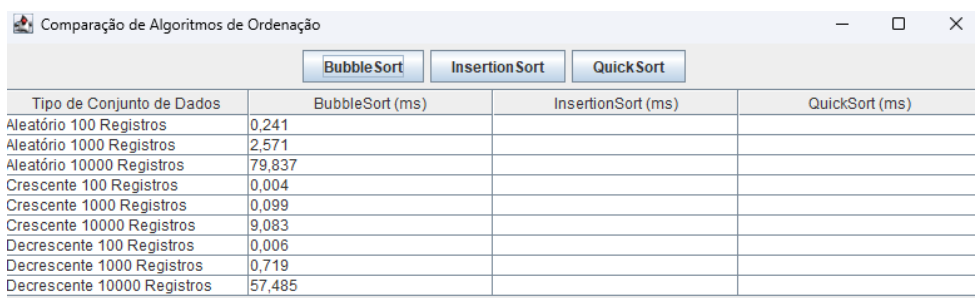
### EXECUÇÃO DO CÓDIGO:

#### \*Tela Inicial



Tipo de Conjunto de Dados	BubbleSort (ms)	InsertionSort (ms)	QuickSort (ms)
Aleatório 100 Registros			
Aleatório 1000 Registros			
Aleatório 10000 Registros			
Crescente 100 Registros			
Crescente 1000 Registros			
Crescente 10000 Registros			
Decrescente 100 Registros			
Decrescente 1000 Registros			
Decrescente 10000 Registros			

#### \*Processamento Bubble Sort



Tipo de Conjunto de Dados	BubbleSort (ms)	InsertionSort (ms)	QuickSort (ms)
Aleatório 100 Registros	0,241		
Aleatório 1000 Registros	2,571		
Aleatório 10000 Registros	79,837		
Crescente 100 Registros	0,004		
Crescente 1000 Registros	0,099		
Crescente 10000 Registros	9,083		
Decrescente 100 Registros	0,006		
Decrescente 1000 Registros	0,719		
Decrescente 10000 Registros	57,485		

## \*Processamento Insertion Sort

	Bubble Sort	Insertion Sort	Quick Sort
Tipo de Conjunto de Dados	BubbleSort (ms)	InsertionSort (ms)	QuickSort (ms)
Aleatório 100 Registros	0,241	0,098	
Aleatório 1000 Registros	2,571	3,063	
Aleatório 10000 Registros	79,837	17,098	
Crescente 100 Registros	0,004	0,019	
Crescente 1000 Registros	0,099	0,030	
Crescente 10000 Registros	9,083	0,203	
Decrescente 100 Registros	0,006	0,010	
Decrescente 1000 Registros	0,719	0,261	
Decrescente 10000 Registros	57,485	23,376	

## \*Processamento Quick Sort

	Bubble Sort	Insertion Sort	Quick Sort
Tipo de Conjunto de Dados	BubbleSort (ms)	InsertionSort (ms)	QuickSort (ms)
Aleatório 100 Registros	0,241	0,098	0,038
Aleatório 1000 Registros	2,571	3,063	0,352
Aleatório 10000 Registros	79,837	17,098	0,814
Crescente 100 Registros	0,004	0,019	0,014
Crescente 1000 Registros	0,099	0,030	0,158
Crescente 10000 Registros	9,083	0,203	0,063
Decrescente 100 Registros	0,006	0,010	0,004
Decrescente 1000 Registros	0,719	0,261	0,212
Decrescente 10000 Registros	57,485	23,376	20,546

Os resultados mostram que o Quick Sort é o algoritmo mais rápido e eficiente para ordenar dados de diferentes tamanhos e tipos (aleatórios, crescentes e decrescentes), destacando-se especialmente em conjuntos grandes. O Insertion Sort é eficiente com dados já ordenados, mas perde desempenho em conjuntos maiores e desordenados. O Bubble Sort é o mais lento em todos os cenários, sendo menos eficiente, especialmente em dados aleatórios e grandes.