Relatório de Desempenho

Descrição do Algoritmo Híbrido

Detalhes da Implementação: O algoritmo híbrido combina o QuickSort com o SelectionSort, utilizando o QuickSort para dividir a lista até que as sublistas alcancem um determinado tamanho (threshold). Quando as sublistas são menores ou iguais ao threshold, o SelectionSort é utilizado para ordenar essas sublistas. O QuickSort utiliza a mediana de três elementos (primeiro, meio e último) como pivô para melhorar a escolha do pivô e, consequentemente, a eficiência da ordenação.

Desempenho:

- **Melhor Caso:** Ocorre quando a lista está quase ordenada ou já ordenada, pois o QuickSort pode particionar a lista de forma eficiente e o SelectionSort tem poucas trocas a fazer. A complexidade é O(nlogn)O(n \log n)O(nlogn).
- **Caso Médio:** Ocorre na maioria das vezes com dados aleatórios. A complexidade é O(nlogn)O(n \log n)O(nlogn), com a vantagem de o threshold reduzir a sobrecarga do QuickSort em listas pequenas.
- **Pior Caso:** Ocorre quando a lista está ordenada em ordem inversa ou quando o pivô escolhido é consistentemente o maior ou o menor elemento. A complexidade é O(n2)O(n^2)O(n2), mas a escolha da mediana de três como pivô geralmente evita esse cenário.

Metodologia do Teste de Desempenho

Conjuntos de Dados:

- **Muito Pequeno:** 1.000 elementos
- **Pequeno:** 10.000 elementos
- **Médio:** 50.000 elementos
- **Grande:** 500.000 elementos Os dados são gerados aleatoriamente com valores inteiros entre 0 e 1.000.000.

Procedimento:

1. Gerar um conjunto de dados para cada tamanho especificado.
2. Para cada algoritmo de ordenação, medir o tempo de execução usando os conjuntos de dados gerados.
3. Repetir cada teste 5 vezes para obter uma média precisa dos tempos de execução.

Número de Repetições: Cada teste é repetido 5 vezes, e a média dos tempos de execução é calculada para obter uma medida representativa do desempenho.

Resultados do Teste de Desempenho

Tamanho do Conjunto	Selection Sort	Quick Sort	Hybrid Sort (16)	Hybrid Sort (64)	Hybrid Sort (256)
1.000	0.022	0.000	0.001	0.001	0.003
10.000	2.255	0.018	0.013	0.018	0.045
50.000	58.405	0.116	0.082	0.108	0.238
500.000					

Discussão dos Resultados

1. Selection Sort

- **1.000 Elementos**: 0.022 segundos
- **10.000 Elementos**: 2.255 segundos
- **50.000 Elementos**: 58.405 segundos

Análise:

- O Selection Sort é um algoritmo de complexidade O(n2)O(n^2)O(n2), o que o torna ineficiente para grandes conjuntos de dados.
- Embora seja simples de implementar, ele não escala bem. Isso é evidente nos tempos de execução que aumentam drasticamente com o aumento do tamanho do conjunto de dados.
- É mais adequado apenas para conjuntos de dados muito pequenos ou quando a simplicidade do algoritmo é mais importante que a eficiência.

2. Quick Sort

- **1.000 Elementos**: 0.000 segundos
- **10.000 Elementos**: 0.018 segundos
- **50.000 Elementos**: 0.116 segundos

Análise:

- O Quick Sort tem uma complexidade média de O(nlogn)O(n \log n)O(nlogn), tornando-o muito eficiente para uma ampla gama de tamanhos de conjuntos de dados.
- Para conjuntos de dados pequenos a médios, o Quick Sort executa muito rapidamente, como evidenciado pelos tempos de execução baixos.

- O tempo de execução aumenta de forma controlada com o aumento do tamanho do conjunto de dados, mostrando que o algoritmo é escalável e eficiente.

3. Hybrid Sort (16)

- **1.000 Elementos**: 0.001 segundos
- **10.000 Elementos**: 0.013 segundos
- **50.000 Elementos**: 0.082 segundos

Análise:

- O Hybrid Sort (com limite 16) combina Quick Sort e Selection Sort, utilizando Selection Sort para sub-listas pequenas.
- Este algoritmo é ligeiramente mais eficiente do que o Quick Sort puro para conjuntos de dados pequenos a médios, devido à eficiência do Selection Sort em listas pequenas.
- A combinação das forças dos dois algoritmos resulta em tempos de execução muito baixos.

4. Hybrid Sort (64)

- **1.000 Elementos**: 0.001 segundos
- **10.000 Elementos**: 0.018 segundos
- **50.000 Elementos**: 0.108 segundos

Análise:

- O Hybrid Sort (com limite 64) apresenta desempenho semelhante ao Quick Sort puro para conjuntos de dados pequenos a médios.
- Como o limite de 64 elementos é maior, há menos chamadas recursivas, o que pode ser vantajoso ou desvantajoso dependendo da distribuição dos dados.
- Para conjuntos de dados pequenos a médios, a performance é muito próxima do Quick Sort puro, com ligeiras diferenças devido ao limite escolhido.

5. Hybrid Sort (256)

- **1.000 Elementos**: 0.003 segundos
- **10.000 Elementos**: 0.045 segundos
- **50.000 Elementos**: 0.238 segundos

Análise:

- O Hybrid Sort (com limite 256) começa a perder eficiência para conjuntos de dados pequenos e médios em comparação com os outros limites híbridos e o Quick Sort puro.
- Para conjuntos de dados maiores, o desempenho pode ser comprometido devido ao maior número de elementos que precisam ser ordenados pelo Selection Sort.
- O tempo de execução é maior em todos os tamanhos de conjunto de dados testados, indicando que o limite de 256 elementos pode ser muito alto para ser eficiente em comparação com limites menores.

Selection Sort é significativamente mais lento e não é adequado para grandes conjuntos de dados. **Quick Sort** e **Hybrid Sort (16)** apresentam os melhores desempenhos para os tamanhos de conjunto de dados testados, com o **Hybrid Sort (16)** sendo ligeiramente mais eficiente em alguns casos devido ao uso eficaz do Selection Sort para sub-listas pequenas. O **Hybrid Sort (64)** é competitivo, mas não necessariamente superior ao Quick Sort puro, enquanto o **Hybrid Sort (256)** perde eficiência devido ao alto limite para uso do Selection Sort.