Relatório de Desempenho

Descrição do Algoritmo Híbrido

Detalhes da Implementação: O algoritmo híbrido combina o QuickSort e 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.

Discussão dos Resultados

Comparação do Desempenho:

- **Selection Sort** mostrou-se ineficiente para conjuntos de dados maiores devido à sua complexidade O(n2)O(n^2)O(n2).
- **Quick Sort** teve um desempenho consistente e eficiente, especialmente para conjuntos de dados maiores.
- **Algoritmo Híbrido** mostrou um desempenho competitivo. Com thresholds menores (como 16), o algoritmo teve um desempenho ligeiramente melhor para conjuntos de dados menores devido à eficiência do SelectionSort em listas pequenas. Com thresholds maiores (como 256), o desempenho foi comparável ao QuickSort puro para conjuntos de dados maiores, mas com uma ligeira sobrecarga.

Insights:

- O **Algoritmo Híbrido** se destaca em cenários onde a lista contém muitas sub-listas pequenas, aproveitando a eficiência do SelectionSort para essas sublistas.
- Pode não ser tão eficiente em listas já ordenadas ou inversamente ordenadas, onde o QuickSort puro com otimizações adicionais (como a escolha de pivô aleatório) pode ser mais eficaz.
- O threshold ideal depende do tamanho do conjunto de dados e da natureza dos dados. Thresholds menores são melhores para dados menores e thresholds maiores são mais eficientes para dados maiores.