Relatório

1. Funcionamento de Cada Estratégia de Escolha do Pivô

- **Primeiro Elemento como Pivô**: Nesta estratégia, o pivô é sempre o primeiro elemento do subarray em consideração. É simples de implementar, mas pode levar a um desempenho ruim para arrays ordenados ou quase ordenados, pois tende a resultar em partições desbalanceadas.
- **Último Elemento como Pivô**: Nesta variação, o pivô é sempre o último elemento do subarray. O comportamento é similar ao do primeiro elemento, e também pode ser desfavorável para arrays já ordenados ou quase ordenados.
- **Pivô Aleatório**: Escolhe um elemento aleatório do subarray como pivô. Esta estratégia visa minimizar a chance de ocorrer uma partição desbalanceada, garantindo um desempenho melhor para arrays com diferentes distribuições.
- **Mediana de Três**: Nesta abordagem, o pivô é escolhido como a mediana dos valores do início, meio e fim do subarray. Esta técnica geralmente resulta em partições mais balanceadas, o que melhora o desempenho em comparação com as abordagens que usam elementos fixos (primeiro ou último).

2. Desempenho Observado

Os resultados dos testes mostraram que as diferentes estratégias de escolha do pivô impactam significativamente o desempenho do QuickSort.

- Para arrays **ordenados** e **quase ordenados**, as estratégias que utilizam o primeiro ou último elemento como pivô tiveram desempenho inferior, especialmente para arrays maiores (como 10.000 elementos), indicando partições desbalanceadas e maior profundidade da recursão.
- O **pivô aleatório** teve um desempenho geralmente melhor do que as estratégias que utilizam o primeiro ou último elemento, mas apresentou alguma variação dependendo do tipo de array.
- A estratégia da **mediana de três** consistentemente apresentou o melhor desempenho, independentemente do tamanho do array ou do tipo, devido à sua capacidade de produzir partições mais balanceadas.

3. Discussão

- Para **arrays ordenados**, o **pivô aleatório** e a **mediana de três** evitaram o pior caso, garantindo uma execução mais eficiente.
- A **mediana de três** foi a estratégia mais eficiente em todos os cenários testados, indicando sua superioridade em manter o equilíbrio das partições.
- O **pivô aleatório** mostrou-se uma boa alternativa, especialmente para arrays aleatórios.
- Tanto o **primeiro** quanto o **último pivô** apresentaram grandes tempos de execução para arrays ordenados, demonstrando a importância de evitar escolher um pivô fixo, especialmente para entradas que podem ser previsivelmente ordenadas.