1. Código-Fonte:

R: https://github.com/dsarafael/Atividade_Algoritmos_Ordenacao/tree/main

2. Tabela de tempos de execução:

R: Encontra-se no item 4 (Análise Experimental) no arquivo Tabela com tempo.pdf

3. Tabela de quantidade de comparações e trocas:

Algoritmo	Comparações	Trocas/Movimentações	Observações
Bubble Sort	$O(n^2)$	0 a $O(n^2)$	Estável
Insertion Sort	$O(n)$ a $O(n^2)$	$O(n)$ a $O(n^2)$	Estável, ótimo para quase ordenado
Selection Sort	$O(n^2)$	$O(n)$	Poucas trocas
Merge Sort	$O(n \log n)$	$O(n \log n)$	Estável, exige memória extra
Quick Sort	$O(n \log n)$ médio	$O(n \log n)$ médio	Pior caso $O(n^2)$
Shell Sort	$O(n \log n)$ a $O(n^{1.5})$	semelhante	Depende da sequência de gaps

4. Tabela de recursões (para os algoritmos que usam este conceito);

Algoritmo	Melhor caso	Caso médio	Pior caso
Merge Sort	$2n - 1$	$2n - 1$	$2n - 1$
Quick Sort	$\approx 2n$	$\approx 1.386 \cdot n \log_2(n)$	$\approx n^2 / 2$

5. Gráfico comparativo de desempenho:
R: Encontra-se no item 4 (Análise Experimental) no arquivo Gráfico com tempo.pdf

6. Tabela resumo das complexidades:

Algoritmo	Melhor caso	Caso médio	Pior caso
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Shell Sort	$O(n \log n)$	$O(n^{1.5})$	$O(n^2)$

7. Conclusão Reflexiva

a. Compare os desempenhos teóricos e práticos.

Algoritmo	Complexidade Teórica (Melhor / Médio / Pior)	Desempenho Prático	Comentário Geral
Bubble Sort	O(n) / O(n^2) / O(n^2)	Extremamente lento em qualquer entrada acima de n=100	Confirma teoria: cresce quadraticamente.
Insertion Sort	O(n) / O(n^2) / O(n^2)	Muito rápido para n=10,100; degrada em n=1000+	Prática confirma teoria: ótimo para listas pequenas ou quase ordenadas.
Selection Sort	O(n^2) / O(n^2) / O(n^2)	Próximo ao Insertion para n pequeno, pior para n grande	Prática confirma teoria: sempre lento, pois sempre faz n^2 comparações.
Merge Sort	O(n log n) / O(n log n) / O(n log n)	Um dos mais rápidos em todas as entradas	Tempo estável confirma teoria.
Quick Sort	O(n log n) / O(n log n) / O(n^2)	O mais rápido nos casos aleatórios; pior em listas ordenadas	Resultados mostram risco de cair no pior caso.
Shell Sort	Entre O(n $\log^2$ n) e O($n^{1.2}$) dependendo do gap	Desempenho muito bom; quase sempre supera Insertion e Selection	Prática geralmente melhor que a teoria prevê.

b. Comente em quais situações cada algoritmo seria mais adequado.

Algoritmo	Melhor uso recomendado	Evitar quando
Bubble Sort	Ensino básico	Qualquer aplicação real
Insertion Sort	Pequenas entradas, vetores quase ordenados	Vetores grandes
Selection Sort	Quando trocas custam caro	Em qualquer cenário de desempenho
Merge Sort	Grandes entradas; precisa ser estável	Pouca memória disponível
Quick Sort	Maior velocidade média	Entrada ordenada/inversa sem tratamento de pivô
Shell Sort	Tamanho médio; rapidez sem complexidade	Quando se exige estabilidade