Insertion Sort:

Melhor Caso: O(n) - No melhor caso, quando a lista está ordenada, o algoritmo só precisa percorrer a lista uma vez para verificar que cada elemento está na posição correta.

Pior Caso: O(n^2) - No pior caso, quando a lista está reversamente ordenada, o algoritmo precisa percorrer a lista e fazer muitas comparações e movimentos de elementos para colocar cada elemento na posição correta.

Selection Sort:

Melhor Caso: O(n^2) - No melhor caso, o número de comparações e movimentos de elementos é sempre o mesmo, independentemente da ordem da lista.

Pior Caso: O(n^2) - O pior caso ocorre quado a lista está ordenada inversamente.

Bubble Sort:

Melhor Caso: O(n) - No melhor caso, quando a lista já está ordenada, o algoritmo precisa percorrer a lista apenas uma vez para verificar se nenhuma troca é necessária.

Pior Caso: O(n^2) - No pior caso, quando a lista está reversamente ordenada, o algoritmo precisa percorrer a lista várias vezes, fazendo muitas trocas para colocar cada elemento na posição correta.

Merge Sort:

Melhor Caso: O(n log n) - No melhor caso, o algoritmo divide a lista pela metade em cada etapa, resultando em um tempo de execução eficiente.

Pior Caso: O(n log n) - No pior caso, o algoritmo mantém a mesma complexidade, independentemente da distribuição dos valores na lista.

Quick Sort:

Melhor Caso: O(n log n) - No melhor caso, quando o pivô divide a lista em duas partes quase iguais em cada recursão, o algoritmo tem desempenho eficiente.

Pior Caso: O(n^2) - No pior caso, o algoritmo pode degradar para O(n^2) se o pivô escolhido

sempre for um dos extremos e a lista estiver ordenada ou quase ordenada.

Radix Sort

Melhor Caso: O(nk) - Quando o número de dígitos (*k*) é pequeno ou constante, o algoritmo tem um desempenho eficiente, pois processa cada dígito de cada elemento.

Pior Caso: O(nk) - Mesmo no pior caso, o algoritmo mantém a mesma complexidade *O*(*nk*), independentemente da distribuição dos valores na lista. Isso ocorre porque ele sempre processa todos os dígitos de todos os elementos.

Shell Sort:

Melhor Caso: O(n log n) - O desempenho do algoritmo depende da sequência de lacunas escolhida.

Pior Caso: O(n^2) - No pior caso, quando a sequência de lacunas não é eficaz, o algoritmo pode se comportar como um Insertion Sort simples.

Heap Sort:

Melhor Caso: O(n log n) - Sempre, pois Heap Sort mantém a mesma complexidade independentemente da distribuição dos valores.

Pior Caso: O(n log n) - Sempre, pois Heap Sort mantém a mesma complexidade independentemente da distribuição dos valores.