Insertion Sort:

Melhor Caso: O(n) - Quando a lista já está ordenada, pois cada elemento é comparado com os elementos à esquerda apenas uma vez.

Pior Caso: O(n^2) - Quando a lista está reversamente ordenada ou completamente desordenada, pois cada elemento precisa ser comparado com todos os elementos à esquerda.

Selection Sort:

Melhor Caso: O(n^2) - Não há melhor caso significativamente melhor que o pior caso.

Pior Caso: O(n^2) - Semelhante ao Insertion Sort, o número de comparações é sempre o mesmo, independentemente da ordem da lista.

Bubble Sort:

Melhor Caso: O(n) - Quando a lista já está ordenada e nenhum swap é necessário durante uma passagem.

Pior Caso: O(n^2) - Quando a lista está reversamente ordenada, pois requer muitos swaps.

Merge Sort:

Melhor Caso: O(n log n) - Sempre. Mesmo quando a lista está desordenada, pois o algoritmo sempre divide a lista pela metade.

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

Quick Sort:

Melhor Caso: O(n log n) - Quando o pivô divide a lista em duas partes quase iguais em cada recursão.

Pior Caso: O(n^2) - Isso pode acontecer se o pivô escolhido sempre for um dos extremos e o array estiver ordenado ou quase ordenado.

Radix Sort:

Melhor Caso: O(nk) - Quando o número de dígitos é constante.

Pior Caso: O(nk) - Sempre. Radix Sort mantém a mesma complexidade independentemente da distribuição dos valores.

Shell Sort:

Melhor Caso: O(n log n) - A complexidade depende da sequência de lacunas escolhida.

Pior Caso: O(n^2) - Quando a sequência de lacunas não é eficaz e se comporta 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.