Miguel Angel Olvera Razon
21110443

Inteligencia Artificial

Metodos de ordenamiento

La ordenación, o clasificación, es una tarea fundamental en el campo de la informática y la ciencia de la computación. Se refiere al proceso de organizar un conjunto de elementos en un orden específico, ya sea ascendente o descendente, según un criterio determinado. Los algoritmos de ordenación desempeñan un papel crucial en la optimización y mejora del rendimiento de una amplia gama de aplicaciones informáticas, desde bases de datos hasta algoritmos de búsqueda y procesamiento de datos.

La elección del algoritmo de ordenación adecuado es esencial y depende de diversos factores, como el tamaño de los datos, la estructura de los datos y los requisitos de tiempo y recursos. Algunos algoritmos son más eficientes en términos de tiempo, mientras que otros pueden ser más adecuados para datos específicos, como números enteros, cadenas de texto o datos dispersos. Comprender los diferentes algoritmos de ordenación y sus características es esencial para tomar decisiones informadas al desarrollar software eficiente y escalable.

Además de los algoritmos de ordenación tradicionales, la informática también ha visto el desarrollo de algoritmos especializados para casos específicos, como algoritmos de ordenación paralela para sistemas con múltiples núcleos de procesador y algoritmos de ordenación externa para manejar conjuntos de datos demasiado grandes para caber en la memoria principal. La ordenación es un tema fundamental que sigue siendo relevante en la resolución de problemas informáticos en la actualidad.

Ordenamiento de burbuja (Bubble Sort):

- Simple pero ineficiente.
- Compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto.
- Tiene una complejidad de tiempo de O(n^2) en el peor caso.

Ordenamiento por selección (Selection Sort):

- Encuentra el elemento más pequeño y lo coloca al principio.
- Continúa buscando el siguiente elemento más pequeño y lo coloca en su posición adecuada.
- Tiene una complejidad de tiempo de O(n^2) en el peor caso.

Ordenamiento por inserción (Insertion Sort):

- Divide la lista en una parte ordenada y una parte desordenada.
- Toma elementos de la parte desordenada y los inserta en la parte ordenada.
- Tiene una complejidad de tiempo de O(n^2) en el peor caso.

Ordenamiento por mezcla (Merge Sort):

- Divide la lista en mitades, las ordena por separado y luego las fusiona.
- Es eficiente y tiene una complejidad de tiempo de O(n log n) en el peor caso.

Ordenamiento rápido (Quick Sort):

- Divide la lista en elementos más pequeños y más grandes con respecto a un pivote.
- Luego, ordena las dos sublistas de forma recursiva.
- Tiene una complejidad de tiempo de O(n^2) en el peor caso, pero generalmente es O(n log n).

Ordenamiento por montones (Heap Sort):

- Crea un montón (estructura de datos especial) y extrae repetidamente el elemento máximo.
- Tiene una complejidad de tiempo de O(n log n) en todos los casos.

Ordenamiento por cuenta (Counting Sort):

- Adecuado para números enteros con un rango limitado conocido.
- Cuenta la frecuencia de cada elemento y luego lo coloca en su posición correcta.
- Tiene una complejidad de tiempo de O(n + k), donde k es el rango de valores.

Ordenamiento por cubetas (Bucket Sort):

- Divide la lista en "cubetas" y luego las ordena individualmente, generalmente utilizando otro algoritmo de ordenamiento.
- Adecuado para datos distribuidos uniformemente.
- Tiene una complejidad de tiempo promedio de O(n + n^2/k + k), donde k es el número de cubetas.

Ordenamiento por radix (Radix Sort):

- Ordena los números por sus dígitos, desde el dígito menos significativo hasta el más significativo o viceversa.
- Adecuado para números enteros y cadenas.
- Tiene una complejidad de tiempo de O(n * k), donde k es el número de dígitos.