**Tarea Corta #2**

Andres Murillo Murillo – C15424

Tabla Comparativa de Algoritmos de Ordenamiento

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo | Mejor Caso | Peor Caso | Caso Promedio | Espacio |
| Insertion Sort |  |  |  |  |
| Selection Sort |  |  |  |  |
| Merge Sort |  |  |  |  |
| Heap Sort |  |  |  |  |
| Counting Sort |  |  |  |  |
| Radix Sort |  |  |  |  |

**1. Insertion Sort (Ordenamiento por Inserción)**

* Mejor caso: Eficiente para conjuntos pequeños de datos o casi ordenados,  *.*
* Peor caso y caso promedio: Ineficiente para conjuntos grandes o desordenados,   
  .
* Espacio requerido: Mínimo, solo necesita espacio adicional constante,

**2. Selection Sort (Ordenamiento por Selección)**

* Mejor, peor y caso promedio: Siempre tiene complejidad cuadrática, independientemente del orden de los datos.
* Espacio requerido: También es mínimo y constante, .

**3. Merge Sort (Ordenamiento por Mezcla)**

* Caso promedio y peor caso: , donde cada división del arreglo implica pasar por todos los elementos y se hace una cantidad de divisiones proporcional al logaritmo del número de elementos .
* Mejor caso: También , ya que incluso en el mejor de los casos se realizan las mismas divisiones y fusiones del arreglo.
* Espacio: , requiere memoria adicional para almacenar las divisiones temporales del arreglo durante la ordenación.

**4. Heap Sort (Ordenamiento por montículo)**

* Mejor caso: tiene un comportamiento que refleja la eficiencia .
* Peor caso: aun en la situación más desfavorable su tiempo de ejecución es
* Caso promedio: , que indica que, en el caso promedio, el tiempo de ejecución será proporcional a también.

**5. Counting Sort (Ordenamiento por Conteo)**

* Mejor caso: , incluso en el mejor de los casos, donde los datos están distribuidos de manera que permiten un conteo rápido.
* Peor caso: , lo mismo sucede en el peor caso, siendo el rango de los números a ordenar.
* Caso promedio: , refleja que el tiempo de ejecución es usualmente proporcional a .
* Espacio: , debido al espacio necesario para los conteos.

**6. Radix Sort (Ordenamiento por Residuos)**

* Mejor caso: , aquí 𝑑d representa el número de dígitos y el rango de valores por dígito.
* Peor caso: , el peor caso tiene la misma complejidad que el mejor caso.
* Caso promedio: , esto indica que, en promedio, el algoritmo es lineal con respecto al número de dígitos y el rango de valores.
* Espacio: , necesita espacio adicional para la ordenación de los "buckets".