

M2 Estructuras de datos FIN A

ACTIVIDAD 1

| **Tutor:** | **Luis Sierra Betancourt** |
| --- | --- |
| **Estudiante:** | **José Ramón Ibáñez Posadas** |
| **Matricula:** | **BNL098377** |

| Monterrey, Nuevo León | Sábado, 10 de Febrero de 2024 |
| --- | --- |

INTRODUCCIÓN

Los algoritmos de ordenamiento son fundamentales en el ámbito de la informática y la ciencia de la computación, ya que permiten organizar conjuntos de datos de manera eficiente. Entre los algoritmos más básicos y comúnmente utilizados se encuentran el algoritmo de ordenamiento por intercambio, el algoritmo de burbuja y el algoritmo de selección directa. En este estudio, se compararán estos tres algoritmos desde diferentes perspectivas, incluyendo su definición, función, características, ventajas y desventajas. Esta comparación proporcionará una comprensión más profunda de cómo funcionan estos algoritmos y cuándo es más apropiado utilizar cada uno de ellos.

.

DESARROLLO

CUADRO COMPARATIVO

| Aspecto | Intercambio | Burbuja | Selección Directa |
| --- | --- | --- | --- |
| Definición | También conocido como Bubble Sort, es un algoritmo simple de ordenamiento que recorre repetidamente la lista, compara elementos adyacentes y los intercambia si están en el orden incorrecto. | Similar al algoritmo de intercambio, el Bubble Sort también es un algoritmo de ordenamiento simple que repite el proceso de comparación e intercambio de pares de elementos adyacentes si están en el orden incorrecto. | Conocido también como Selection Sort, es un algoritmo de ordenamiento que divide la lista en dos partes: la parte ordenada y la parte no ordenada, y encuentra el elemento más pequeño en la parte no ordenada y lo intercambia con el primer elemento de la parte no ordenada. |
| Función | Ordena una lista moviendo los elementos más grandes hacia el final de la lista en cada iteración. | Ordena una lista moviendo los elementos más grandes hacia el final de la lista en cada iteración. | Ordena una lista moviendo el elemento más pequeño a su posición correcta en cada iteración. |
| Características | - Simple de implementar.  - Ineficiente para listas grandes debido a su complejidad O(n^2).  - Útil para listas pequeñas o casi ordenadas. | - Simple de entender e implementar.  - Ineficiente para listas grandes debido a su complejidad O(n^2).  - Funciona bien con listas casi ordenadas. | - Fácil de implementar.  - Eficiente para listas pequeñas.  - Ineficiente para listas grandes debido a su complejidad O(n^2). |
| Ventajas | - Fácil de entender e implementar.  - Útil para listas pequeñas o casi ordenadas. | - Fácil de entender e implementar.  - Útil para listas pequeñas o casi ordenadas. | - Fácil de implementar.  - Eficiente para listas pequeñas. |
| Desventajas | - Ineficiente para listas grandes debido a su complejidad O(n^2).  - No es recomendado para uso en aplicaciones donde se requiera eficiencia en el tiempo. | - Ineficiente para listas grandes debido a su complejidad O(n^2).  - No es recomendado para uso en aplicaciones donde se requiera eficiencia en el tiempo. | - Ineficiente para listas grandes debido a su complejidad O(n^2).  - No es recomendado para uso en aplicaciones donde se requiera eficiencia en el tiempo. |

EJERCICIO

Con base al [tema 2](https://cnci.blackboard.com/bbcswebdav/courses/BbCont10/Contenido/2Test/I_EstDatos_Claudia_DEMOMAR20/clase_02/clase_02.html) de tu curso, determina cual es el número mayor del siguiente arreglo, para resolverlo utiliza el algoritmo de burbuja, deberás anexar el procedimiento.



Para determinar el número mayor del arreglo utilizando el algoritmo de burbuja, primero vamos a ordenar el arreglo de forma ascendente y luego identificamos el último elemento, que será el número mayor. Aquí está el procedimiento paso a paso:

1. **Arreglo inicial:** [80, 47, 56, 87, 62, 94]

2. **Paso 1:** Comenzamos comparando el primer y segundo elemento. Si el primero es mayor que el segundo, los intercambiamos. Seguimos haciendo esto hasta llegar al final del arreglo.

- Comparación: 80 > 47, no se intercambian.

- Comparación: 80 > 56, no se intercambian.

- Comparación: 80 > 87, no se intercambian.

- Comparación: 87 > 62, se intercambian: [80, 47, 56, 62, 87, 94]

- Comparación: 87 < 94, no se intercambian.

3. **Paso 2:** Repetimos el mismo proceso, pero esta vez comenzando desde el primer elemento hasta el penúltimo (ya que el último elemento ya está en su posición final).

- Comparación: 80 > 47, no se intercambian.

- Comparación: 80 > 56, no se intercambian.

- Comparación: 80 > 62, no se intercambian.

- Comparación: 80 < 87, no se intercambian.

4. **Paso 3:** Repetimos el proceso nuevamente, pero esta vez comenzando desde el primer elemento hasta el antepenúltimo.

- Comparación: 80 > 47, no se intercambian.

- Comparación: 80 > 56, no se intercambian.

- Comparación: 80 < 62, no se intercambian.

5. **Paso 4:** Continuamos hasta que no haya más elementos por comparar.

- Comparación: 80 > 47, no se intercambian.

- Comparación: 80 < 56, no se intercambian.

6. **Arreglo ordenado:** [47, 56, 62, 80, 87, 94]

El número mayor del arreglo original es el último elemento del arreglo ordenado, que es 94. Por lo tanto, el número mayor del arreglo [80, 47, 56, 87, 62, 94] es 94.

CONCLUSIÓN

Tras analizar y comparar los algoritmos de ordenamiento por intercambio, burbuja y selección directa, se puede concluir que, aunque todos ellos son relativamente simples de entender e implementar, tienen diferentes eficiencias y adecuaciones para distintos escenarios. Mientras que el algoritmo de selección directa muestra una mejor eficiencia en términos de tiempo para listas pequeñas, los algoritmos de intercambio y burbuja tienden a ser menos eficientes, especialmente para listas más grandes debido a su complejidad O(n^2). Es importante considerar estas características al seleccionar un algoritmo de ordenamiento para una aplicación específica, eligiendo aquel que mejor se adapte a las necesidades de rendimiento y tamaño de los datos a ordenar.

BIBLIOGRAFÍA

Universidad CNCI. (2024). Algoritmos de Intercambio y selección. Recuperado de <https://cnci.blackboard.com/bbcswebdav/courses/BbCont10/Contenido/2Test/I_EstDatos_Claudia_DEMOMAR20/clase_02/clase_02.html>

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley.