**Jean Paul Ortiz**

**Juan Jose Arias Murillo**

**Sebastian Zapata**

**Jhonny Benítez**

**Guía 2**

**Juego de Adivinanza de Palabras Versión 2.0**

**1. Correcciones y mejoras.**

**1.1 Correcciones:**

* El código terminaba de ejecutar, cuando se le colocaba un carácter al momento de responder la pregunta adicional, ya fue corregido y sencillamente dice que esta mala, que debe ser un número entero.

**1.2 Mejoras:**

* Se le añadió un nuevo jugador durante las partidas.
* Se le añadieron más preguntas adicionales de forma aleatoria.
* Se le añadieron niveles de dificultades a las preguntas, dando diferentes puntajes por medio de ellas.

**1.3 Archivos**

Los archivos están en formato txt, donde se crearon, se les adjunta la información, y se sigue conservando al añadir nueva información al final:

* Un archivo almacena las preguntas con sus respuestas, tanto de adivinanzas como de las preguntas adicionales.
* Un archivo para guardar los nombres de los jugadores con sus respectivos puntajes durante cada partida.
* Un archivo donde se resumen los datos anteriores, ordenando los puntajes de mayor a menos, mediante el método de ordenación MergeSort.

**2. Método de ordenación MergeSort.**

**2.1 Descripción:**

El algoritmo de ordenación MergeSort, también conocido como ordenación por mezcla, es un algoritmo de ordenación eficiente y versátil que se basa en el principio de "divide y vencerás". Divide el problema de ordenar una lista en subproblemas más pequeños de menor tamaño, los ordena recursivamente y luego los combina para obtener la lista final ordenada.

**2.2 Funcionamiento:**

* **Dividir:** Se divide la lista en dos sublistas de aproximadamente igual tamaño.
* **Conquistar:** Se ordenan recursivamente las dos sublistas usando el mismo algoritmo MergeSort.
* **Combinar:** Se combinan las dos sublistas ordenadas en una sola lista ordenada.

**2.3 La combinación:**

La parte crucial del algoritmo es la combinación de las dos sublistas ordenadas. Se comparan los elementos de las dos sublistas y se va agregando el menor elemento a la lista final ordenada. Este proceso se repite hasta que se hayan agregado todos los elementos de ambas sublistas.

**2.4 Complejidad computacional:**

El algoritmo MergeSort tiene una complejidad computacional promedio y en el peor caso de O(n log n), donde n es el número de elementos en la lista. Esto significa que el tiempo que tarda el algoritmo en ordenar una lista crece logarítmicamente con el tamaño de la lista.

**2.5 Ventajas:**

* **Eficiente:** Es uno de los algoritmos de ordenación más eficientes, con una complejidad computacional de O(n log n).
* **Estable:** Preserva el orden original de los elementos iguales.
* **Versátil:** Se puede utilizar para ordenar cualquier tipo de datos comparables.
* **Fácil de implementar:** La implementación del algoritmo es relativamente sencilla.

**2.6 Desventajas:**

* **Requiere memoria adicional:** El algoritmo requiere memoria adicional para almacenar las sublistas durante la división y la combinación.
* **No es tan eficiente para listas pequeñas:** Para listas pequeñas, otros algoritmos de ordenación como la ordenación por inserción pueden ser más eficientes.

**2.7 Desventajas:**

* **Requiere memoria adicional:** El algoritmo requiere memoria adicional para almacenar las sublistas durante la división y la combinación.
* **No es tan eficiente para listas pequeñas:** Para listas pequeñas, otros algoritmos de ordenación como la ordenación por inserción pueden ser más eficientes.

**2.8 Ejemplo:**

Texto

Descripción generada automáticamente

**2.9 Conclusión:**

Es un algoritmo de ordenación eficiente, estable y versátil que se utiliza ampliamente en diversos campos de la informática. Es una buena opción para ordenar listas grandes de datos, especialmente cuando se requiere preservar el orden original de los elementos iguales.

**3. Libro Code Complete de Steve McConnell**

**Diapositivas.**