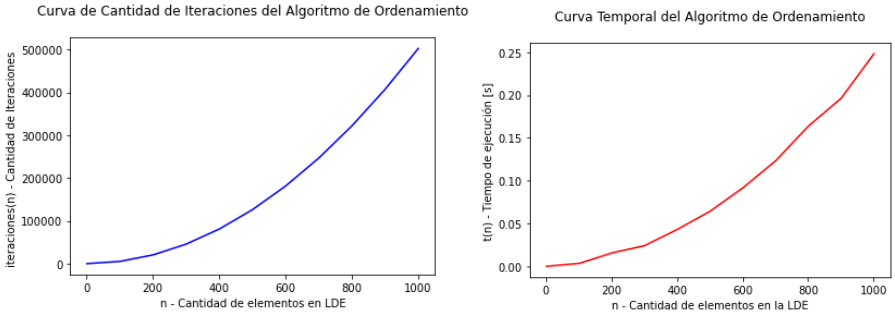
Ejercicio1:

En este ejercicio se implementó una lista doblemente enlazada, la cual se hizo el análisis de complejidad en un script dentro de la carpeta pruebas. El algoritmo de ordenamiento implementado es el de selección y las graficas



Se agrego una prueba que de iteraciones en función de la cantidad de elementos que permite independizarnos del software y nos permite tener una métrica de la eficiencia del algoritmo. Si tenemos en cuenta que el ordenamiento por selección tiene un orden de complejidad de O(n^2), en nuestro caso obtuvimos un resultado menor y tras varias implementaciones con distintos datos, la grafica del numero de iteraciones no variaba.

Ejercicio 2

En este ejercicio se nos proporciona la problemática de programar el “Juego de la Guerra”, y dentro de sus requerimientos se nos establecen nombres de métodos a seguir, y la implementación de una semilla para poder dirigir un test sobre el juego.

Para brindar una solución diseñamos un tipo abstracto de dato (TAD), cola doble, el cual posee métodos de inserción y eliminación de sus componentes en ambos extremos, sabe su tamaño y si está vacía. Este TAD se diseña empleando herencia de un TAD mas genérico, la lista doblemente enlazada del ejercicio 1. Una vez establecida esta capa de abstracción, nos centramos en desarrollar la app del juego de guerra propiamente dicho. Esta app cuenta con la capacidad de generar un mazo de cartas de póker de 52 cartas, respetando los 4 mazos característicos, y luego barajar según una semilla aleatoria y repartir las cartas del mazo entre 2 jugadores fijos. Una vez repartido el mazo original, se está en condiciones de comenzar el juego según las reglas establecidas en el enunciado del ejercicio.

La lógica establecida para obtener un ganador por cada turno es por comparación de las cartas, y siendo el caso que sean iguales, en valor, se da lugar a la “Guerra”. Este suceso es una condición especial en la cual se añaden mas cartas al tablero y se vuelve a realizar una comparación para establecer un ganador. Como estadísticamente es posible que sucedan 2 “Guerras” seguidas, se contempló en el código de la aplicación este suceso, pero no más de 2 Guerras seguidas debido a que la probabilidad de ocurrencia de este suceso es muy baja.

La comunicación de resultados, número de turno, cantidad de cartas y cartas en el tablero se realiza a través del terminal en una interfaz grafica sencilla.

Ejercicio 3

En este ejercicio se nos solicitaba generar un archivo de 100 Mb y mediante el método de ordenamiento externo “Mezcla Natural”, ordenar el mismo de forma creciente. Este archivo ordenado se guarda como un archivo distinto al original dado que es de interés el poder compararlos posteriormente. A modo de ejemplo se deja una imagen del funcionamiento lógico esperado del método mezcla natural.

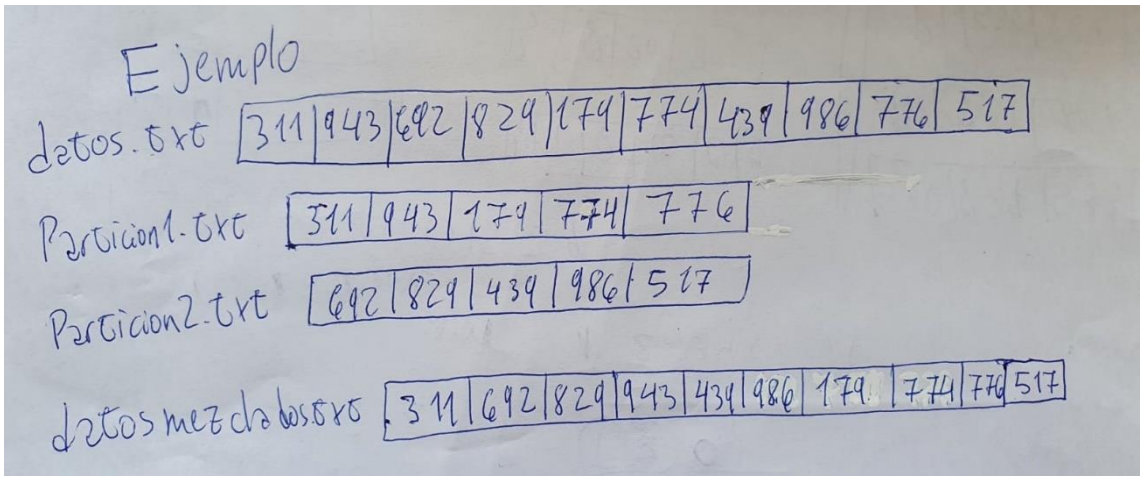


Imagen N°1: Método de ordenamiento Mezcla Natural.

Como vemos ilustrado, inicialmente contamos con un archivo “datos”, el cual tiene un tamaño n. Como primera iteración si divide este archivo en 2 “particiones”, estos nuevos archivos son completados a medida que se satisface una condición de comparación entre en dato en “datos” y el próximo dato del mismo.

Posteriormente se realiza una fusión en la cual se comparan los primeros valores de cada una de las particiones, se escribe en el archivo “datos mezclados” el menor de estos, y luego se avanza en la lectura y comparación de valores provenientes del archivo que fue denominado como mayor. De esta forma se escriben los valores ordenados del archivo “mayor” hasta que su siguiente valor es menor que su actual. En esta instancia, se guarda esta última posición y se alterna al archivo que originalmente fue clasificado como menor. Siguiendo la misma lógica descripta hasta el momento, se escribe en el archivo de mezcla los datos de ambas particiones hasta que una de ellas finaliza. Por último, si fuera necesario, se completa el archivo mezcla con los datos sobrantes de las particiones que no fueron comparadas por diferencias de tamaños.

Informe de errores

- **Ejercicio 2:** Cuando uno de los jugadores se queda sin cartas, el juego debe terminar y anunciar como ganador al jugador opuesto. Pero siendo que existen las evaluaciones de tamaños de mazos, cuando se llega a esta condición de “mazo nulo”, el juego no corta y sigue solicitando cartas, lo que produce un AtributeError.
- **Ejercicio 3:** Considerando la lógica antes descripta, se logra que se divida el archivo original creado en dos particiones, pero en el momento de fusionar las mismas en un archivo mezclado, se cometen errores de interpretación que no nos permitieron finalizar la aplicación.