Facultad de Ingeniería

Tecnicatura Universitaria en
Procesamiento y Explotación de Datos

Algoritmos y estructuras de datos

Trabajo práctico nº 1

Colignon, Sabrina
Narváez, Micaela
Samaniego, Francisco

Ejercicio 1

Orden de complejidad de los métodos:

Para el análisis de las funciones que hemos desarrollado, en los casos de:

- **esta_vacia()**
- **tamanio()**
- **agregar(item)**
- **anexar(item)**
- **extraer(posición)**

  Todas tienen orden de complejidad 0(1).

En los casos de:

- **insertar(posicion, item)**
- **copiar()**
- **invertir()**
- **concatenar(Lista)**

  Todas tienen orden de complejidad O(1) en el mejor caso, en el caso promedio O(n) y en el peor caso O(n).

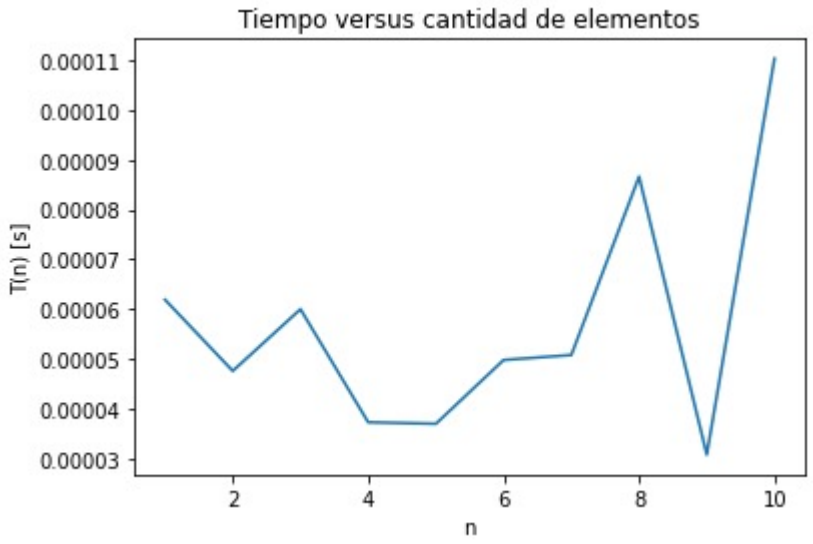
En el caso de:

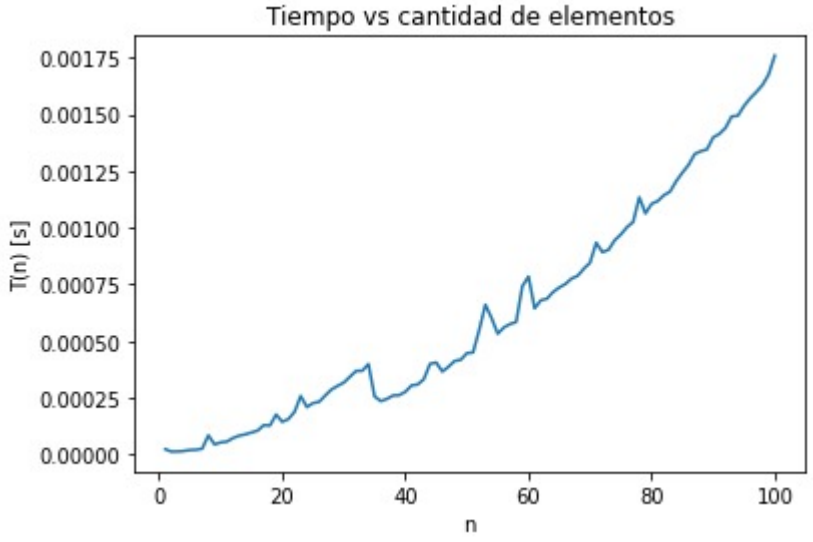
- **ordenar()**

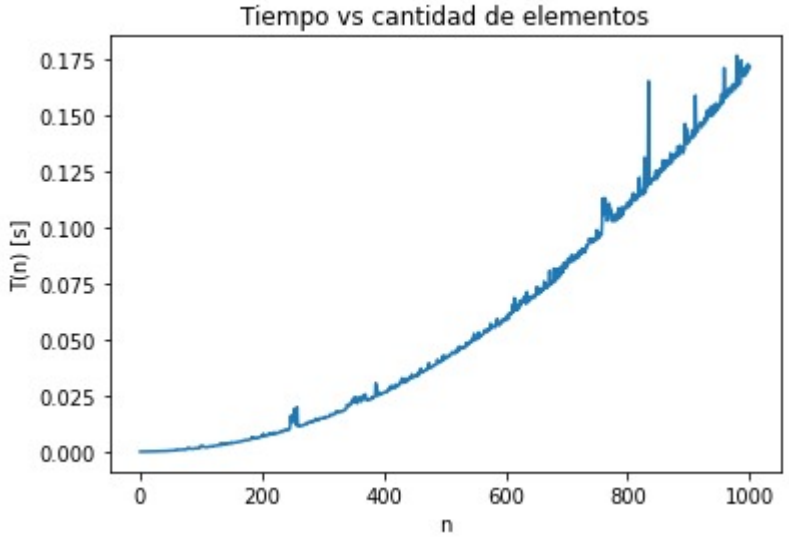
  Tiene orden de complejidad O(n) en el mejor caso, en el caso promedio y peor caso tiene orden de complejidad O(n^2).

En el caso de los algoritmos con ordenamiento O(1) deberíamos ver una línea recta constante, en el segundo caso (O(n)) una gráfica lineal y en el último caso (O(n^2)) una gráfica cuadrática.

Siguiendo las indicaciones del presente trabajo práctico “El algoritmo de ordenamiento del método “ordenar” de la Lista debe tener la eficiencia del algoritmo de ordenamiento por inserción, o mejor.” Seleccionamos el ordenamiento por burbuja, cuyo orden de complejidad es el mismo que inserción, a sabiendas que él algoritmo de burbuja es menos eficiente, no pudimos desarrollarlo correctamente. Utilizando 10, 100 y 1000 iteraciones podemos ver cómo el algoritmo va formando la gráfica esperada.



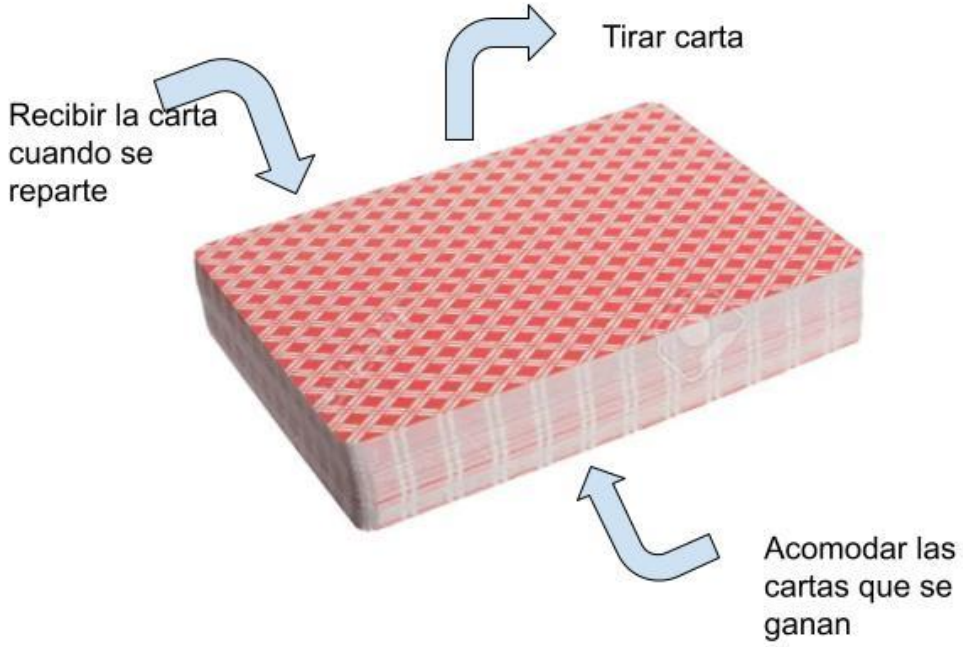




Se adjunta el .py en el cual está el código para graficar el método ordenar y su gráfica.

Ejercicio 2

Decidimos que el “gestor” de este juego sea la clase Juego_guerra, es decir que esta clase será la encargada de repartir las cartas, iniciar el juego, comparar y decidir qué carta es mayor lo que implica decidir cuál de los dos jugadores se lleva la o las cartas, esta clase es la que va contando los turnos.
Elegimos la estructura de cola doble para construir los mazos de los jugadores ya que deben recibir las cartas (se suman por arriba) que se reparten, luego sacar cartas del mazo y colocar las que gane al final



Esa estructura de cola doble internamente contiene la lista doblemente enlazada del **ejercicio 1.**

Ejercicio 3

Ahora dirigimos nuestra atención a usar una estrategia de dividir y conquistar como una forma de mejorar el desempeño de los algoritmos de ordenamiento.

Para realizar este algoritmo nos basamos en el libro *TADs, Estructuras de datos y resolución de problemas con c + + de Nyhoff,* que con la ayuda de un pseudocódigo nos ayudó a comprender lo que debíamos pasar a un algoritmo computacional.

También para facilitar la comprensión y el trabajo decidimos hacer una función para dividir el archivo original, una función para mezclar y luego una de ordenamiento que utiliza las dos últimas de forma recursiva hasta que el archivo está ordenado.