Grado en Diseño y Desarrollo de Videojuegos
Algoritmos para Juegos
Práctica 3: Comparativa de algoritmos de ordenación

1. Implementa una función que reciba un array de `numpy` y lo ordene utilizando el método de inserción directa.
2. Implementa una función que reciba un array de `numpy` y lo ordene utilizando *mergesort*.
3. Implementa un programa que compruebe el rendimiento de los diferentes algoritmos de ordenación sobre arrays de diferentes tamaños. Para ello, partiendo de un valor mínimo (`MIN`), un máximo (`MAX`), y un incremento (`STEP`), se creará un array aleatorio de enteros en un rango de 0 a 1000 (ver la función `np.random.randint` para más información). El primer array tendrá un tamaño de `MIN`, el segundo de `MIN+STEP`, el tercero de `MIN+2*STEP`, y así sucesivamente hasta que el tamaño supere el valor establecido en `MAX`. Cada uno de los arrays se ordenará utilizando inserción directa o mergesort, dependiendo del algoritmo que estemos analizando. A continuación, se imprimirá, para cada array, el tamaño del array ordenado y el tiempo que ha necesitado el programa para ordenarlo, separado por un tabulador (‘`\t`’). Para el análisis del tiempo es de utilidad el módulo `time`, que se deberá importar para su uso. El método `time.perf_counter()` proporciona el valor actual del reloj del sistema.
4. Analiza los resultados obtenidos por inserción directa y mergesort, copiando la salida del programa en una hoja de cálculo y representando los resultados en una gráfica. Comprueba de forma visual la complejidad de cada uno de los métodos de ordenación.
5. Para comprobar la eficiencia de `numpy`, realiza la misma prueba utilizando la función `np.sort`, y modificando su parámetro `kind`, que indica el algoritmo de ordenación utilizado.