Laboratorio N-3.1

Antes de comenzar ...

1. Instalar dependecias y creacion del `package.lock.json`

   ```
   npm i
   ```

2. El codigo fue separado en carpetas `functions` y `clases` su funcion tiene una relacion directa con el nombre respectivo.
3. Compilar el programa

   ```
   node index.js > data/data.csv
   ```

   La salida de consola sera almacenada en un archivo `.csv`

Primer ejercicio

1. ***Para elementos desordenados entonces siempre convendrá utilizar mergesort. ¿Será cierto eso?.***

   Solucion:

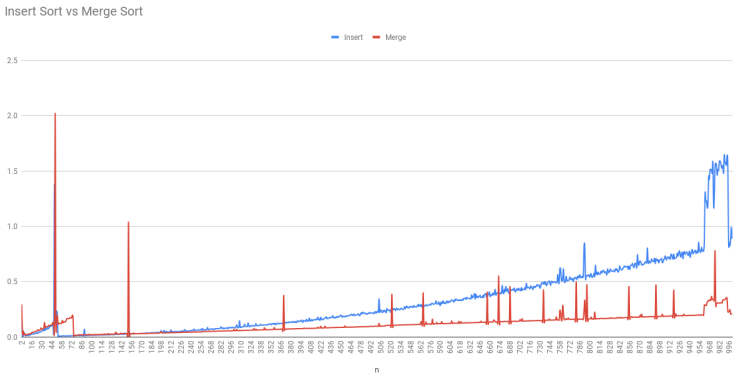
   FALSO, solo para elementos desordenados grandes de n, Merge Sort es el mas optimo algoritmo de ordenacion.

2. ***Sabemos que para n suficientemente grande, esto se cumple a no ser que los elementos ya estén ordenados, pero ¿qué pasaría si n es pequeño?***

   Solucion:

   Para elementos de n, si n es pequeño, el ***Insert Sort*** se ejecuta mas rapido que el ***Merge Sort.***

3. ***Encontrar, de forma gráfica, para qué valores de n, insertsort es mejor que mergesort. La respuesta para la mayoría de computadores está en un valor inferior a n < 1000.***



Solucion:

1. El siguiente grafico muestra la ***comparacion de costo de tiempo entre el Insert Sort y Merge Sort***, aplicados a arreglos aleatorios de tamaño 2 hasta $N <= 1000$.
2. Dado los resultados obtenidos a partir de $n <= 200$ ***Insert Sort es mejor que Merge Sort***.