Miguel Méndez Murias
UO287687 - Algoritmia

Práctica 4

En esta práctica se tenía que elaborar un algoritmo que permitiese solucionar el siguiente problema:
"Un ejército enemigo ha desembarcado en las costas de nuestro país invadiendo n ciudades. Los servicios de "inteligencia" están informados de que en cada una de las ciudades invadidas se encuentran ei efectivos enemigos. Para contraatacar, el Grupo de Intervención Rápida de Defensa dispone de n equipos listos para intervenir. Cada uno de estos equipos consta de dj efectivos entrenados para este tipo de acciones y no se pueden pasar efectivos entre equipos. Tenemos que asignar un equipo a cada ciudad, pero para garantizar el éxito de la intervención en una ciudad, es necesario que dispongamos, al menos, de tantos efectivos como el enemigo."

Para ello se realizó en java un algoritmo voraz que lo solucionase siguiendo el siguiente heurístico:

Para cada ciudad en la que desembarcaban enemigos se le asigna la menor defensa que sea capaz de ganarle y que no esté ya asignada para defender otra ciudad. En caso de que ninguna defensa pudiese vencer a los enemigos se asignaba a la más débil disponible. De esta forma, podemos asegurarnos que se conseguirá siempre el máximo número de victorias posibles.

El algoritmo desarrollado que implementaba el heuristico anterior tiene una complejidad $O(n^2)$, ya que se recorren las colecciones de enemigos y defensas de forma anidada mediante dos bucles for. Además, hay muchas otras operaciones O(1) en el método, pero como los bucles tienen una complejidad O(n) y al estar anidados se convierte en una complejidad $O(n^2)$, esta es la complejidad final del método, ya que es la mayor de todas ellas.

Mediciones empíricas de tiempo.

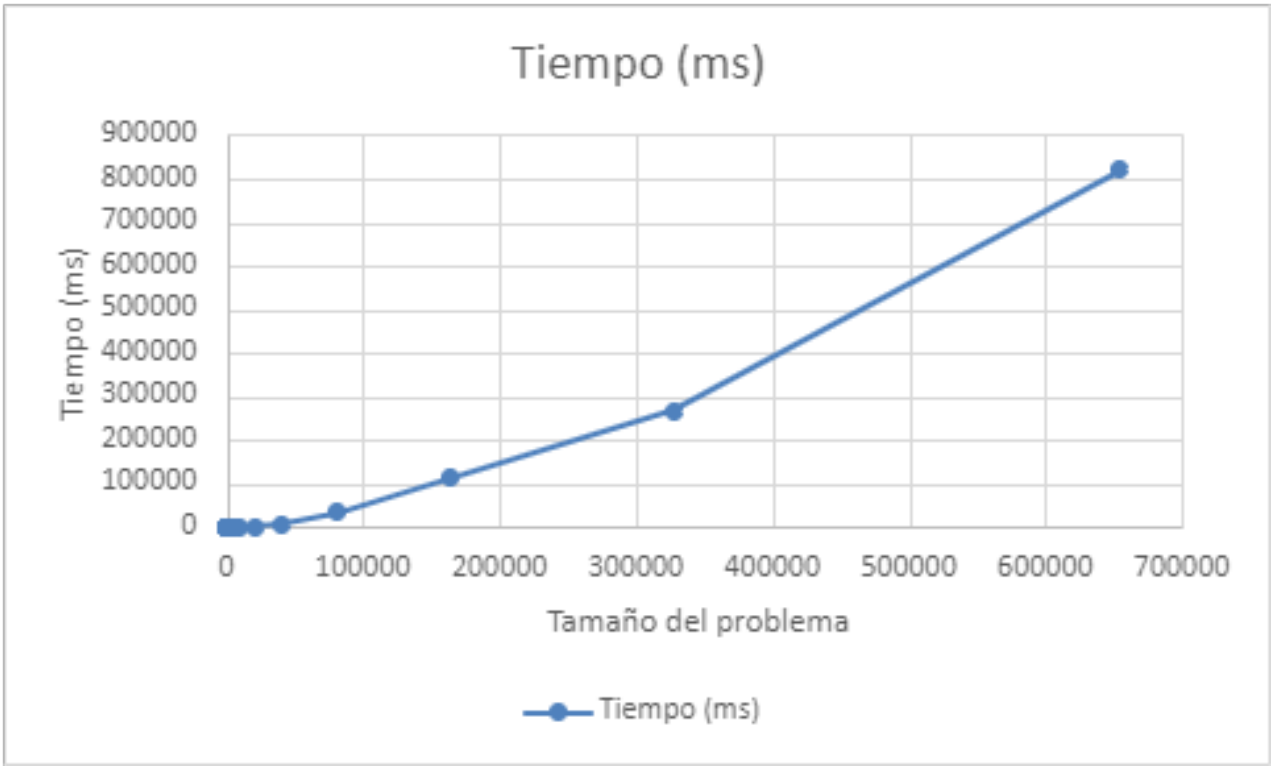
Para realizar las mediciones de tiempo de lo que tarda el algoritmo explicado anteriormente, se desarrolló una nueva clase que medía los tiempos para cada tamaño del problema, es decir, para un número de enemigos y de defensas que había que emparejar y que iba creciendo.

Los tiempos obtenidos fueron los siguientes:

Tamaño del problema	Tiempo (ms)
10	0
20	0
40	1
80	0
160	1
320	3
640	5
1280	19
2560	24

Miguel Méndez Murias
UO287687 - Algoritmia

5120	89
10240	346
20480	1413
40960	5140
81920	34830
163840	111002
327680	265582
655360	818447



Como se puede observar en los tiempos, la complejidad del algoritmo se cumple, ya que estos van aumentando de forma cuadrática.

Parte opcional: versión mejorada del algoritmo

Si bien no se ha programado al algoritmo correspondiente a esta parte, si que se podría mejorar el algoritmo para solucionar el problema explicado al principio de la práctica. Esto se conseguiría ordenando los enemigos y las defensas según su fuerza, lo cuál tendría una complejidad O(n * log n). Posteriormente, solo quedaría asignar las defensas de la forma más óptima posible. Para ello, una de las opciones sería escoger el enemigo más fuerte que no se ha defendido y asignarle la defensa más fuerte disponible si es capaz de ganarle o la defensa más débil posible si la más poderosa cae derrotada. De esta forma se consigue que las mejores defensas siempre ganen, mientras que se sacrifican a las más débiles, lo que permitiría conseguir el máximo número de victorias posibles.
La complejidad final de este algoritmo coincide con la complejidad de ordenar las colecciones, es decir, tendría una complejidad O(n * log n), lo que mejoraría la complejidad anterior, que es cuadrática.