Análisis de Parejas Iguales

En este análisis voy a comparar dos algoritmos que cuentan parejas de números en un array que se le pasa como parámetro. El tiempo de ejecución de estos algoritmos depende principalmente del tamaño de array, cuya longitud consideraremos n.

El primer algoritmo, ParejasIguales, usa una búsqueda simple, cogiendo cada término del array y comparándolo con cada término siguiente, utilizando dos bucles anidados. El primer bucle se ejecuta n veces, mientras que el segundo se ejecuta cada vez menos veces, empezando ejecutándose (n-1) veces la primera pasada hasta llegar a 1 vez en la última. Esto en teoría nos da (n-1) + (n-2) + (n-3) + … + 3 + 2 + 1 ejecuciones del bucle más interno, que puede ser simlificado a (n – 1) * (n – 2) / 2 iteraciones totales, una experesión que se aproxima al crecimiento cuadrático. En ejecuciones de DoublingRatio se obtienen los siguientes resultados:

n:	t:	ratio:
64000	1.0	3.2
128000	3.6	3.6
256000	14.3	4.0
512000	57.2	4.0

Se puede apreciar que conforme los valores de n aumentan, el ratio se va aproximando a 4, cuando n se dublica, indicativo de que el crecimiento es de n^2.

El segundo algoritmo, ParejasIgualesFast, ordena el array con Arrays.sort. Despúes, recorre el array comparando cada término con el siguiente hasta encontrar uno distinto, pues si el array está ordenado encontrar un término distinto garantiza que no quedan parejas. Por cada término igual que encuentra, aumenta el contador de parejas en j, siendo j el número de términos iguales después del primero. La lógica a seguir es que cada elemento nuevo genera una pareja con cada elemento anterior. La primera repetición genera 1 pareja, la segunda genera 2, la tercera genera 3…
Este algoritmo tiene 2 bucles, uno que itera tantas veces como números distintos haya en el array, y otro que itera n veces. Por tanto, en un array donde todos los números sean iguales, el primer bucle iterará 1 vez y el segundo n, mientras que si el array tiene todos los números distintos, ambos bucles iterarán n veces. Por tanto, sin contar la ordenación de array, el algoritmo tiene un rango entre n y 2n iteraciones, en ambos casos el crecimiento es lineal. Teniendo en cuenta que Arrays.sort tiene un crecimiento linearítmico, ese será el crecimiento del algoritmo. Los resultados de DoublingRatio:

n:	t:	ratio:
4096000	0.5	2.0
8192000	1.0	1.8
16384000	2.0	2.1
32768000	3.9	2.0
65536000	7.5	1.9
131072000	15.0	2.0
262144000	29.3	2.0

indican una clara tendencia a duplicarse a la vez que n, indicativos claros de un crecimiento aproximado al lineal. El crecimiento linearítmico es difícil de apreciar con Doublin Ratio, así que los resultados se adecúan a las expectativas de un crecimiento muy similar al lineal, una clara optimización frente al crecimiento cuadrático de ParejasIguales.