Laboratorio 1: Análisis de algoritmos

Importar a Eclipse el proyecto Lab1.

Ejercicio 1.

En este ejercicio se trabaja con arrays de números enteros que están **ordenados ascendentemente, y que no tienen elementos repetidos**. Los enteros pueden ser tanto positivos como negativos.

-7	-4	-1	0	1	2	3	7

Responde a las siguientes cuestiones:

a. ¿Qué hace el método *cuantosContrarios1* de la clase Contrarios? ¿Cuál es su orden temporal?

Cuenta cuántos números contrarios hay en el array ordenado. Orden temporal N^2

b. Ejecuta la clase *PruebaContrarios*. ¿Hasta qué tamaño del array se consigue ejecutar en menos 12 segundos?

256000 en 9.702 segundos

c. Diseña e implementa un algoritmo *cuantosContrarios2* que haga lo mismo que *cuantosContrarios1* en O(N), siendo N el nº de elementos del array. Escribe un programa de prueba que compruebe que el algoritmo es correcto.

d. Ejecuta la clase *PruebaContrarios* pero sustituyendo la llamada a *cuantosContrarios1* por *cuantosContrarios2*. ¿Hasta qué tamaño del array se consigue ejecutar en menos de 12 segundos?

1048576000 en 3.4 segundos antes de que salte java.lang.OutOfMemoryError

Ejercicio 2.

Un grupo de amigos está haciendo una colección de cromos. El grupo está compuesto por MAXAMIGOS amigos y la colección completa está compuesta por MAXCROMOS cromos diferentes. Tanto los amigos como los tipos de cromo se identificarán mediante enteros consecutivos comenzando en el 0 (es decir, si hubiera 3 amigos, se identificarían mediante 0, 1 y 2, y análogamente para los cromos).

Se os entregan dos representaciones posibles:

Cromos1

En la clase ***Cromos1*** se utiliza una matriz donde las filas representan a los amigos y las columnas a los diferentes cromos de la colección. El valor de cada casilla *cromos[i][j]* indica cuántos cromos de tipo j tiene el amigo i. Ejemplo:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	MAXCROMOS-1
0	0	0	0	2	1	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1
2	0	2	0	0	0	1	1	1	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
5	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0
MAXAMIGOS-1	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0

En este ejemplo se puede ver que el amigo 2 tiene dos cromos de tipo 1, y un cromo de cada uno de los tipos 5, 6 y 7.

Se entrega ya implementado el método que carga desde un fichero la información de los amigos y los cromos a la estructura. Se os entregan también varios ficheros de prueba, cuyo formato está explicado al final de este ejercicio.

Debes responder a las siguientes cuestiones:

a. Completa los métodos *cuantosNoTieneNadie()* y *cuantasApariciones(int numCromo)*.

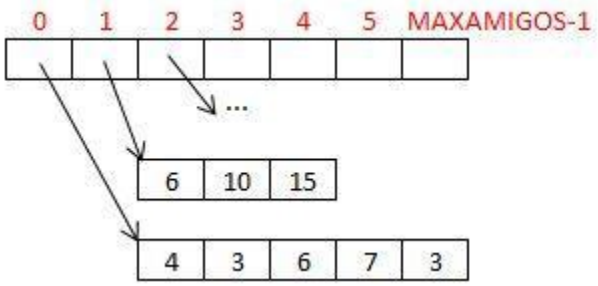
   El método *cuantosNoTieneNadie* devuelve cuántos cromos hay que no los tenga ningún amigo.

   El método *cuantasApariciones* devuelve cuántas veces está un determinado cromo (contando las repeticiones).

b. ¿Cuál es el orden temporal de dichos métodos?

Cromos2

En la clase ***Cromos2*** se utiliza una representación diferente de la colección de cromos. En concreto se utiliza el array *cromos*, donde en *cromos[i]* se guarda un ArrayList que contiene los cromos del amigo i. Si tiene algún cromo repetido, ese cromo aparecerá varias veces en el ArrayList. Supondremos que cada amigo como mucho tendrá 10 veces el mismo cromo.



En el ejemplo podemos ver que el amigo 0 tiene dos veces el cromo 3 y una vez cada uno de los cromos 4, 6 y 7.

c. ¿Cuál es el orden temporal del método *cuantosNotieneNadie1* que se entrega implementado?

d. Piensa un algoritmo *cuantosNoTieneNadie2* más eficiente e impleméntalo en la clase *Cromos2*. Implementa también el método *cuantasApariciones(int numCromo)*.

e. ¿Cuál es el orden temporal de dichos métodos?

Formato de los ficheros de prueba

En la primera línea: nº de amigos y nº de tipos de cromos.
Después, una línea para cada amigo. El primer número de cada línea indica el identificador del amigo y el resto de números los identificadores de los cromos que tiene dicho amigo.