Práctica 1

Parcial 1. Temas 1, 2 y 3

Objetivo

Resolver de forma individual y original los ejercicios planteados, para demostrar que se alcanza los siguientes resultados de aprendizaje de la asignatura:

- **RA1.** Realizar cálculos de la complejidad teórica de un algoritmo y su orden de magnitud, para poder argumentar la elección de una solución frente a otra.
- **RA2.** Emplear estrategias algorítmicas clásicas (divide y vencerás, avance rápido y vuelta atrás), y ser capaz de utilizarlas a la hora de implementar soluciones a problemas concretos.
- **RA3.** Utilizar las principales estructuras de datos lineales asociativas (tablas hash), su utilidad, su funcionamiento y su implementación. Ser capaz de utilizarlas para la resolución de problemas concretos y como mecanismo para obtener soluciones más óptimas.

Ejercicio 1

Sean A y B dos matrices cuadradas NxN de enteros, se dice que A "es neutralizada" por B, si A - B = I, donde I es la matriz identidad.

En el fichero Java adjunto a esta actividad figura una clase que contiene tres métodos:

- `esNeutralizada_v1  (int[][] m1, int[][] m2): boolean`
- `esNeutralizada_v2  (int[][] m1, int[][] m2): boolean`
- `esNeutralizada_DyV (int[][] m1, int[][] m2): boolean`

El primero de ellos plantea un algoritmo para determinar si una matriz cuadrada NxN m1 es neutraliza por la también matriz NxN m2 (siendo N un número potencia de 2). Esta primera versión en realidad no es correcta, y además tampoco es del todo eficiente.

En el método main de la clase se han creado varias matrices y llamadas a los métodos para hacer pruebas. Como puede verse, el resultado de esNeutralizada_v1 no es el esperado para todos los casos, y falta por completar el código de los otros dos métodos.

Se pide realizar las implementaciones siguientes, respetando siempre las cabeceras propuestas, pero pudiendo añadir los métodos auxiliares (privados) que se consideren oportunos:

a) Manteniendo los dos bucles **for**, realizar las modificaciones **mínimas** necesarias en el código de **esNeutralizada_v1** para que simplemente funcione de manera correcta.

b) Tras los cambios anteriores, esNeutralizada_v1 será correcta, pero no será una versión eficiente. Se pide entonces implementar en **esNeutralizada_v2** una versión más eficiente en tiempo para resolver el problema.

c) Implementar en **esNeutralizada_DyV** una versión que siga la estrategia "Divide y Vencerás" para resolver el problema.

d) Calcular la complejidad de las tres funciones. Aplicar los correspondientes sumatorios y reducciones para argumentar y demostrar el resultado alcanzado.

e) Comparar las complejidades obtenidas, y argumentar cuál de las tres versiones sería la mejor.

NOTA: No se aceptarán entregas donde los cálculos de las complejidades no estén desarrollados y razonados, ni los realizados "a mano alzada" (fotografías, escaneado de resoluciones hechas a bolígrafo, etc.)

Ejercicio 2

Implementar una función que utilizando la estrategia de **"Divide y Vencerás"**, determine si dos arrays de números enteros son uno el inverso del otro.

Por ejemplo: [1,2,3,4,5,6,7,8] y [8,7,6,5,4,3,2,1] devolverían true, pero [1,2,3,4] y [4,3,2,2] devolverían false.

Calcular razonadamente su complejidad, detallando las reducciones o sumatorios necesarios para su desarrollo, y comparar el resultado con el que obtendríamos si aplicásemos una estrategia iterativa tradicional.

NOTA: No se aceptarán entregas donde los cálculos de las complejidades no estén desarrollados y razonados, ni los realizados "a mano alzada" (fotografías, escaneado de resoluciones hechas a bolígrafo, etc.)

Ejercicio 3

Se tiene la tabla hash de la figura, para la que se conocen las siguientes restricciones:

a) Se utiliza doble hashing a partir de las siguiente expresión para gestionar las colisiones:
$H(k) = H1(k) + (c \cdot H2(k))$, donde c representa el número de colisiones.
$H1(k) = k \bmod N$
$H2(k) = 3 - (k \bmod 3)$

b) No se quiere superar el 80% del factor de carga de la tabla.

Se pide desarrollar (explicar y representar gráficamente) paso a paso cada operación de inserción de las siguientes claves, que llegarían una a una en el mismo orden en que se listan a continuación: 16, 12, 27, 29.

0	
1	
2	30
3	
4	2
5	
6	

Normas de entrega

La entrega consistirá en **2 ficheros**: un archivo **.java** para el código y un documento **PDF** con la descripción de todo el trabajo y resolución de todos los ejercicios.

Ambos archivos se comprimirán en **formato ZIP** (no RAR, ni 7Z u otros). No respetar los formatos descritos supondrá la no aceptación de la entrega.

Los diferentes ejercicios de entregarán de la siguiente manera:

- **Ejercicio 1:**
  - Se entregará en el propio fichero **Neutraliza.java**. Es condición necesaria que el código compile.
  - Se completarán los apartados (comentarios iniciales y salidas por consola) donde se hace referencia a los datos personales del estudiante.
  - Se entregará adicionalmente en PDF todos los apartados (a, b, c, d y e). Para los tres primeros simplemente será necesario "copiar y pegar" (ajustando formatos) el código de cada función tal y como se ha incluido en el código fuente.
  - Los apartados d y e deberán argumentarse convenientemente. Deberán desarrollarse los sumatorios y reducciones resultantes de los códigos, utilizando algún editor de fórmulas para los sumatorios (ni fotos, ni escaneos, etc.)
- **Ejercicio 2:**
  - Se adjuntará en PDF. No es necesario adjuntar fichero JAVA para este ejercicio.
  - Se espera encontrar únicamente el código de la función DyV pedida.
  - Se debe desarrollar, siguiendo las mismas pautas del ejercicio 1-d, la complejidad pedida, y comparar el resultado con la versión tradicional. Esta comparación implica razonar cuál sería mejor opción y por qué.
- **Ejercicio 3:**
  - Se incluirá en el PDF.
  - Deben razonarse, desarrollarse e ilustrarse (mostrarse gráficamente) todos los pasos de manera clara, paso a paso.

- De nuevo, sólo admiten desarrollos realizados con el propio editor de texto: no se deben incluir dibujos hechos a mano, capturas de pantalla, escaneos, etc.

Antes de subir la entrega, seguir los siguientes cuatro pasos:

1. Crear una carpeta con el siguiente formato de nombre:

   Apellido1_Apellido2_Nombre.TPA.P1

2. En dicha carpeta, incluir los dos ficheros indicados:
   a. El código fuente del ejercicio 1: Neutraliza.java
   b. El **PDF** con todos los detalles de la actividad. Este documento seguirá el siguiente formato a la hora de ser nombrado:

   Apellido1_Apellido2_Nombre.TPA.P1.**PDF**

3. Comprimir en formato **ZIP** (no RAR, no 7Z, etc.) la carpeta creada en el punto 1, lo que generará un fichero que seguirá el siguiente formato:

   Apellido1_Apellido2_Nombre.TPA.P1.**ZIP**

4. Entregar el fichero ZIP **antes de las 23:55h** de la fecha marcada como límite.