Universidad Rey Juan Carlos

Prueba de Diseño de Algoritmos – Junio – Curso 2016-2017

Diseño y Análisis de Algoritmos

Valor: 35 % de la nota final. Duración: **1 hora y 45 minutos**

Ejercicio 1 Dígitos comunes [4 puntos]

Sea **a** un array o lista de números naturales. Se quiere hallar el conjunto de dígitos comunes a todos los elementos de **a**. Por ejemplo, para $\mathbf{a} = [2348, 1349, 7523, 3215]$, la solución es $\{3\}$.

Para representar el conjunto de dígitos del resultado se usará un array o lista de booleanos de tamaño 10, que expresará si un elemento es común o no. De esta manera, la solución del ejemplo anterior se representará con el array o lista [false, false, false, true, false, false, false, false, false, false].

Se pide implementar un algoritmo basado en la técnica de **divide y vencerás** que resuelva el problema.

Ejercicio 2 [2 puntos]

Sea un vector **v** formado por un número n par de enteros. Hay que agrupar a los elementos de **v** en $n/2$ parejas de forma que si se halla la suma de los dos miembros de cada pareja y se toma el máximo de estas sumas, el número resultante sea el menor posible.

Por ejemplo, dado el vector $\mathbf{v} = [-1, 5, 12, 7, 0, 2, -3, 9]$, una partición óptima de **v** es el conjunto de pares {(0,7),(-1,9),(2,5),(-3,12)}, donde la suma máxima de sus pares es 9, del par $(-3, 12)$. Puede comprobarse que cualquier otra partición produce una suma máxima mayor o igual que 9.

Se pide describir una **estrategia voraz** que obtenga una solución óptima al problema propuesto. En este ejercicio no se pide código.

Ejercicio 3 [4 puntos]

Sea una lista S de números enteros. Se desea hallar todas las formas de dividir S en dos listas A y B, tal que las sumas de sus elementos sean iguales. Por ejemplo, dado $S = [1, 2, 3, 4, 5, 9]$, dos posibles particiones son $(A = [1, 2, 4, 5], B = [3, 9])$ y $(A = [1, 2, 9], B = [3, 4, 5])$. Se pide implementar un algoritmo **eficiente** (que realice podas en el árbol de recursión) para resolver el problema basado en la estrategia de **backtracking**. Para ello se deberá partir de alguno de los esquemas explicados en las transparencias para generar subconjuntos. El algoritmo imprimirá por pantalla las parejas de listas cada vez que las encuentre.