Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Diseño de Algoritmos / Algoritmos II
Primer Cuatrimestre de 2018

Examen Parcial No. 1

Este examen debe ser resuelto en forma individual. No olvide poner su nombre y número de documento en el encabezado de la resolución.
Este examen consta de seis ejercicios. **Sólo cuatro de ellos deben resolverse**. No resuelva más de cuatro ejercicios, pues los ejercicios sobrantes no serán considerados.
Cada ejercicio tiene un valor de 2,5 puntos. La nota mínima de aprobación es de cinco puntos.

Ej. 1. Para disminuir el costo de reordenar una secuencia, se quiere invocar a un proceso de reordenamiento sólo cuando la lista a ordenar se considere "suficientemente desordenada". Para esto, se necesita determinar cuán desordenada está una lista dada, y para esto se define la siguiente métrica:

> El *grado de desorden* de una secuencia es el *mínimo* número de intercambios de valores en posiciones contiguas que se deben hacer para reestablecer el orden de la secuencia.

Por ejemplo, el grado de desorden de $[1, 2, 3, 4, 5, 6, 7]$ es 0, mientras que el grado de desorden de $[1, 2, 3, 4, 6, 5, 7]$ es 1, y el de $[1, 2, 5, 4, 4, 6, 7]$ es 2.

Diseñe e implemente en Java o Haskell un algoritmo que, dada una secuencia s, determine el grado de desorden de la misma. Qué complejidad tiene su solución? Justifique.

Ej. 2. Considere el problema de dar cambio por C centavos, con monedas de denominaciones $1 < d_2 < \ldots < d_k$. Se desea conocer, dados C y las denominaciones de las monedas, cuántas formas diferentes de dar cambio existen, suponiendo que se cuenta con una provisión infinita de monedas de cada una de las denominaciones. Diseñe un algoritmo que resuelva este problema, e implemente el mismo en Java o Haskell.

Ej. 3. Dada una secuencia s ordenada de enteros y un entero x, se desea encontrar el valor más cercano a x en la secuencia, es decir, el valor cuya diferencia en valor absoluto es la menor posible, entre todos los elementos de la secuencia. Considerando que la secuencia está almacenada en un arreglo, diseñe e implemente un algoritmo, en Java o Haskell, que resuelve este problema en $O(\log n)$.

Ej. 4. Dada una secuencia s ordenada de enteros y un entero x, se desea conocer el número de repeticiones de x en s. Considerando que la secuencia está almacenada en un arreglo, diseñe e implemente un algoritmo, en Java o Haskell, que resuelve este problema en $O(\log n)$.

Ej. 5. Es sabido que las *relaciones de equivalencia* inducen particiones en un dominio D, y que las particiones se corresponden con relaciones de equivalencia. Dado un dominio D, simplemente un conjunto finito de elementos, se desea construir *todas* las particiones posibles del conjunto D, es decir, todas las relaciones de equivalencia. Recordemos que una partición de un conjunto es una separación del mismo en subconjuntos no vacíos mutuamente disjuntos. Diseñe e implemente un algoritmo, en Java o Haskell, que resuelve este problema.

Ej. 6. Una terna pitagórica consiste en una tupla de tres enteros positivos a, b, c que cumplen que $a^2 + b^2 = c^2$. Se quiere construir un programa que, dado un valor n, genere *todas* las ternas pitagóricas con a, b y c acotados por n (es decir menores o iguales a n). Diseñe e implemente un algoritmo, en Java o Haskell, que resuelve este problema.