Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Diseño de Algoritmos / Algoritmos II
Primer Cuatrimestre de 2014

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. Dada una secuencia de n valores enteros, se desea ordenar la misma eficientemente. Más precisamente, sabiendo que los valores de la secuencia se encuentran en el rango $1..k$, se quiere ordenar la secuencia en $O(n \log k)$. Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.

Ej. 2. Dada una secuencia de n valores enteros, se desea calcular el menor entero positivo que no pertenece a la secuencia, en $O(n)$. Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.

Ej. 3. Dada una secuencia ordenada de enteros distintos $\{a_1, a_2, \ldots, a_n\}$, se requiere determinar, en $O(\log n)$, si existe un índice i tal que $i = a_i$. Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.

Ej. 4. Los *bags* son estructuras de datos similares a las listas y los conjuntos, que pueden pensarse en algún sentido "intermedias" entre estas dos estructuras: en los bags "no importa el orden", como en los conjuntos, aunque importa el número de repeticiones de cada elemento, como en las secuencias. Suponiendo que un bag está representado mediante una secuencia ordenada, se desea construir un algoritmo que calcule todos los "sub-bags" del mismo, bajo la misma representación y asegurando que ningún sub-bag se repite más de una vez. Por ejemplo, si el bag de entrada es $[1, 2, 2, 3]$, el resultado debería ser

$$\{[1, 2, 2, 3], [1, 2, 3], [1, 3], [1], [1, 2, 2], [1, 2], [2], [2, 2], [2, 2, 3], [2, 3], [3]\}.$$

Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.

Ej. 5. Dadas dos secuencias s_1 y s_2, y un natural k, se desea determinar si s_1 se puede obtener a partir de s_2, realizando a lo sumo k inserciones de caracteres, en cualquier posición, en s_2. Por ejemplo, si $s_1 = [a, c, d, b]$ y $s_2 = [b, b]$, entonces claramente no es posible convertir s_2 en s_1 mediante inserciones (cualquiera sea el valor k); en cambio, si $s_1 = [a, c, d, b]$ y $s_2 = [a, b]$, entonces podemos obtener s_1 realizando dos inserciones, insertando c después de a, y d después de c. Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.

Ej. 6. Dado un conjunto de letras proposicionales $\{p_1, p_2, \ldots, p_k\}$, se quiere construir el conjunto de todas las *valuaciones* posibles, es decir, todas las formas posibles de combinar las letras proposicionales y sus negaciones. Por ejemplo, si las letras proposicionales son $\{p, q\}$, entonces debe producirse como resultado el conjunto $\{[p, q], [\neg p, q], [p, \neg q], [\neg p, \neg q]\}$. Diseñe e implemente un algoritmo que resuelva este problema, usando Java o Haskell según su conveniencia.