Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Diseño de Algoritmos / Algoritmos II
Primer Cuatrimestre de 2017

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. Consideremos las siguientes operaciones de edición de cadenas. Dada una cadena s de caracteres:

- Se puede pasar un caracter de s de minúscula a mayúscula,
- Se puede eliminar todos los caracteres en minúscula de s.

Dadas dos cadenas s_1 y s_2, se quiere saber si la primera cadena se puede convertir en la segunda, usando exclusivamente las operaciones admitidas anteriormente. Implemente, usando Java o Haskell, un algoritmo que resuelva este problema.

Ej. 2. Dada una cadena s de caracteres sobre el alfabeto $\{a, b, c\}$, se permite realizar sobre la misma sólo un tipo de modificación: un par de caracteres adyacentes distintos se puede reemplazar por el caracter restante.

Se desea conocer, dada una cadena inicial s, cuál es la cadena más pequeña a la que se puede llegar mediante la aplicación repetida de la modificación anterior. Diseñe un algoritmo que resuelva este problema, e implemente el mismo en Java o Haskell.

Su algoritmo puede retornar sólo la longitud de la cadena más chica obtenida, en lugar de la cadena en sí.

Ej. 3. Un agente inmobiliario ha estimado el precio de una propiedad para los próximos n años, y en base a esa información quiere aconsejar a un cliente cuál es el momento óptimo de compra y de venta de la misma. Más precisamente, lo que se desea maximizar es la ganancia obtenida por el cliente, es decir, la diferencia (positiva) obtenida entre el precio de compra y el precio de venta. Diseñe un algoritmo, e implemente el mismo en Java o Haskell, que resuelva este problema en tiempo polinomial.

Ej. 4. Un banco monitorea la actividad de las cuentas de sus clientes, en la búsqueda de acciones fraudulentas. La política que utiliza para advertir a sus clientes es muy simple: cuando la suma gastada por un cliente en una fecha d particular es mayor o igual a la mediana de los últimos k días (k días previos a n), el banco envía una notificación de potencial fraude. El banco no envía ningún tipo de notificación hasta no tener al menos k fechas previas.

Dados los gastos de un cliente en n fechas, y un valor para k, diseñe un algoritmo e impleméntelo en Java o Haskell, que imprima las fechas en las cuales enviaría las notificaciones al cliente.

Ej. 5. El algoritmo de Knuth-Morris-Pratt resuelve de manera eficiente el problema de decidir si una cadena es subcadena de otra. Este algoritmo requiere un algoritmo auxiliar para computar los *bordes* de cada uno de los sufijos de una cadena. El borde de una cadena s es, simplemente, un prefijo de s (de longitud menor a s) que también es sufijo de s.

Diseñe e implemente en Java o Haskell un algoritmo que, dada una cadena s, calcule el borde de cada sufijo de s.

Ej. 6. Un elemento es *mayoría* en una secuencia s de tamaño n si el mismo se repite más de $n/2$. Se desea determinar si existe en una secuencia un elemento que sea mayoría en $O(n \log n)$, con la restricción adicional de que el algoritmo debe ser *in place*, es decir, el mismo puede utilizar una cantidad constante de espacio de memoria, además del espacio ocupado por la secuencia de entrada. Además, la secuencia de entrada no puede modificarse (es de sólo lectura).

Diseñe un algoritmo que resuelva este problema con las restricciones indicadas, e implemente el mismo en Java o Haskell.