Estructura de Datos y Algoritmos

Grados en Ingeniería Informática, de Computadores y del Software

Examen Final Junio - 1 de junio de 2018 - Grupos C y F

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. (3.5 puntos) Sean dos cadenas de caracteres $V[0..N), W[0..M)$ con N, M arbitrariamente grandes. Se quiere desarrollar un programa que compruebe si la primera cadena es prefijo de la segunda sin tener en cuenta los espacios blancos que pueda haber en ambas. Para ello se pide:

- (0.5 puntos) Especifica una función que dadas dos cadenas de caracteres $V[0..N), W[0..M)$ tales que $N <= M$ y ninguna de ellas tiene ningún carácter blanco compruebe que la primera cadena es prefijo de la segunda.
- Implementa la función anterior, escribe el invariante (0.5 puntos) y la función cota (0.2 puntos) del bucle utilizado y comprueba el paso 2 de la verificación $\{I \wedge B\}S\{I\}$ (0.3 puntos)
- (2 puntos)Modifica la implementación de la función anterior para que resuelva el problema completo de comprobar que la primera cadena es prefijo de la segunda sin tener en cuenta los espacios en blancos que pueda haber en ambas y pudiendo ser la primera cadena más larga o más corta que la segunda. La implementación debe realizarse con complejidad $\mathcal{O}(N + M)$. No pueden utilizarse cadenas auxiliares, ni otras estructuras auxiliares, ni realizar copia de cadenas, ni modificar las cadenas dadas.

Ejemplos. Los blancos se indican con el carácter subrayado. ϵ es la cadena vacía. (Fichero Problema1Entrada.txt)

V	W	prefTrimSpaces
ada_cccb_b	a_dacc_cbb	TRUE
ada_ccc___	a_dacc_cbb	TRUE
aaa_cccb_b	a_dacc_cbb	FALSE
ada_ccbb_b	a_dacc__	FALSE
__________	a_dacc_cbb	TRUE
ϵ	ϵ	TRUE
a_da	ϵ	FALSE
ϵ	a_d	TRUE
a_d___	ad	TRUE

La entrada consta de una serie de casos de prueba. Cada caso de prueba tiene dos líneas, en la primera se indica la cadena de caracteres que debe ser prefijo de la que aparece en la segunda línea.

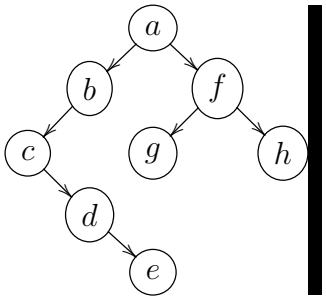
Para cada caso de prueba se escribirá en una línea `TRUE` si la primera cadena es prefijo de la segunda y `FALSE` en caso contrario.

Nota: La especificación y verificación la puedes realizar en papel o sobre el fichero de código.

2. **(2.25 puntos)** Implementa una función (externa al TAD List) que dada una lista enlazada de caracteres (`List<char>`) `l`, y dos números enteros `i` y `k`, invierta el segmento que comienza en la posición `i`-ésima de la lista `l` (se empieza desde la posición 1) y contiene `k` elementos. Por ejemplo, si la lista está formada por los elementos *a b c d e f*, e invertimos el segmento que comienza en la posición 3 y tiene longitud 3, entonces la lista resultante es *a b e d c f*. Si $i + k > l.size() + 1$ se invertirá el segmento que va desde la posición `i`-ésima hasta el final de la lista. Se valorará la complejidad en tiempo y en espacio de la función implementada, la cual debes indicar y justificar. ¿Qué ventaja(s) tendría una implementación interna al TAD List?

3. **(4.25 puntos)** En un árbol binario, se define el *factor de equilibrio* de un nodo como el valor absoluto de la diferencia entre las alturas de sus dos hijos. Decimos que un nodo es *equilibrado* si su factor de equilibrio es 0 o 1. Por otro lado decimos que un nodo es *padre* si al menos tiene un hijo (no vacío), y es un *buen padre*, si es padre y sus hijos son equilibrados. Se pide:

   a) **(2 puntos)** Implementar una función que dado un árbol binario de caracteres devuelva el porcentaje (como entero redondeado hacia abajo) de nodos padres que son buenos padres. En el árbol del ejemplo hay 5 padres y 3 de ellos (c, d y f) son buenos padres. Se debe devolver por tanto 60. Aclaración: dados dos enteros n y m ($n < m$) debes calcular el porcentaje mediante `int(double(n)*100/m)`. Si `m` = 0 debes devolver 100.



   b) **(2.25 puntos)** Implementar un programa que imprima un listado de todos los nodos del árbol que no son equilibrados, incluyendo para cada uno, el factor de equilibrio, el camino para llegar a él desde la raíz, y el valor del nodo. El listado debe ir ordenado de menor a mayor factor de equilibrio, y en caso de empate, por orden lexicográfico del camino. Para ello debes: (1) definir el tipo `TablaDesequilibrados` para almacenar el listado; (2) implementar una función que dado un árbol binario de caracteres devuelva una `TablaDesequilibrados`; y (3) implementar una función que dada una `TablaDesequilibrados` imprima el listado (ver plantilla y casos de prueba proporcionados). Aclaración: Si utilizas un `string` de 0's y 1's para representar el camino (0 significa ir por el hijo izquierdo y 1 ir por el derecho) el operador de orden del tipo `string` implementa precisamente el orden lexicográfico. En el ejemplo, los nodos desequilibrados son: a y c, con factor 2 y caminos "" y "00" resp., y b con factor 3 y camino "0". Justifica el coste de las funciones implementadas.

Instrucciones para la realización y entrega del examen

- Las plantillas y casos de prueba para poder probar vuestras soluciones, así como los TADs del 2º cuatr., se obtienen pulsando en el icono del Escritorio "Publicacion docente ...", después en "Alumno recogida docente", y en el programa que se abre, abriendo en la parte derecha la carpeta EDA-CF, arrastrando los ficheros a hlocal (en la izqda).
- Al principio del fichero principal de cada ejercicio debe aparecer, en un comentario, tu nombre completo, dni y puesto de laboratorio. También debes incluir unas líneas explicando lo qué has conseguido hacer.
- Todo lo que no sea código C++ (explicaciones, costes, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- La entrega se realiza por el juez online accesible en la dirección `http://exacrc/domjudge/team`. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. Tus soluciones serán evaluadas independientemente del veredicto del juez. Se tendrá en cuenta **exclusivamente** el último envío de cada ejercicio.