Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencia de la Computación
Estructura de Datos y Algoritmos II

Nombre y Apellido: Legajo:

Examen Parcial 2

1. Suponiendo una representación de secuencias con árboles binarios, definidos con el siguiente tipo de datos:

$$\textbf{data } \mathit{Tree}\ a = E \mid L\ a \mid N\ \mathit{Int}\ (\mathit{Tree}\ a)\ (\mathit{Tree}\ a)$$

donde se almacenan los tamaños de los árboles en los nodos y el recorrido inorder del árbol da el orden de los elementos de la secuencia, definir en Haskell, de manera eficiente, paralelizando cuando sea posible, las siguientes operaciones sobre secuencias:

a) La función $\mathit{concat} :: \mathit{Tree}\ (\mathit{Tree}\ a) \to \mathit{Tree}\ a$, que concatena una secuencia de secuencias. Por ejemplo,

$$\mathit{concat} < <5, 2, 3>, <6>, <8, 0> > = <5, 2, 3, 6, 8, 0>$$

b) La función $\mathit{subsequence} :: \mathit{Tree}\ a \to \mathit{Int} \to \mathit{Int} \to \mathit{Tree}\ a$, que dada una secuencia s y dos enteros i y j, devuelve la subsecuencia de s que está entre los índices i y j. Por ejemplo,

$$\mathit{subsequence} < 1, 2, 3, 4, 5, 6 > 2\ 4 = \cancel{<2, 3, 4>} = <3,4,5>$$ (indizaremos las seq en 0)

0 1 2 3 4 5

2. Usando las operaciones del TAD Secuencia definir las siguientes funciones:

a) $\mathit{uniquify} : \mathit{Seq\ Int} \to \mathit{Seq\ Int}$, que elimina los elementos duplicados de una secuencia de enteros. Esta función puede definirse con profundidad en $O((\lg n)^2)$, donde n es la longitud de la secuencia.

b) $\mathit{aa} : \mathit{Seq\ Char} \to \mathit{Int}$, que dada una secuencia de caracteres cuenta la cantidad de subsecuencias contiguas "aa" que contiene. La subsecuencia "aaa" debe sumar 2. Por ejemplo,

$$\mathit{aa} < a, c, d, e, a, a, f, a, a, a> = 3$$

La función puede definirse con profundidad en $O(\lg n)$, siendo n la longitud de la secuencia.

c) Calcular el trabajo y la profundidad de las funciones definidas en los apartados anteriores.

3. Sea $g = (V, E)$ un grafo simple, un lado $(v, w) \in E$ es llamado **puente** si (v, w) no está incluído en ningún ciclo de g, es decir que al eliminarlo del grafo no existe más un camino de v a w.
Suponiendo que los grafos están representados como tablas de adyacencias, es decir que $\mathit{Graph}\ V = \mathit{Table}\ V\ \mathbb{S}_V$, y que se cuenta con una definición de la función $N_G : \mathit{Graph}\ V \to \mathbb{S}_V \to \mathbb{S}_V$ que calcula el conjunto de vecinos para un conjunto de vértices en un grafo (es decir, $N_G(S) = \cup_{v \in S}(\mathit{getNbrs}\ G\ v)$), definir de manera eficiente las siguientes funciones:

a) $\mathit{path} : \mathit{Graph}\ v \to (v, v) \to \mathit{Bool}$, que determine si existe un camino entre dos vértices de un grafo.

b) $\mathit{elimEdge} : \mathit{Graph}\ v \to (v, v) \to \mathit{Graph}\ v$, que dados un grafo, un lado del mismo, elimine la arista del grafo.

c) $\mathit{bridge} : \mathit{Graph}\ v \to (v, v) \to \mathit{Bool}$, que dados un grafo y una arista del mismo determine si la arista es un puente del grafo.

Nota: El puntaje máximo para cada ejercicio será dado si el costo de la implementación dada coincide con el menor costo posible.