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

Práctica 7

1.

a) Definir y dar los costos de las siguientes operaciones del TAD Conjunto utilizando las demás operaciones del TAD:

- $\mathit{find} : \mathbb{S}_A \to A \to \mathit{Bool}$, tal que $\mathit{find}\ S\ e = e \in S$.
- $\mathit{insert} : \mathbb{S}_A \to A \to \mathbb{S}_A$, tal que $\mathit{insert}\ S\ e = S \cup \{e\}$.
- $\mathit{delete} : \mathbb{S}_A \to A \to \mathbb{S}_A$, tal que $\mathit{delete}\ S\ e = S - \{e\}$.

b) Dada la siguiente función

$$\begin{array}{lcl} \mathit{fromSeq} & : & \mathit{Seq}\ A \to \mathbb{S}_A \\ \mathit{fromSeq}\ s & = & \mathit{reduce}\ \mathit{union}\ \mathit{empty}\ (\mathit{map}\ \mathit{singleton}\ s) \end{array}$$

i. ¿A qué TAD pertenece cada operación en la definición?

ii. Suponiendo que la comparación es O(1), mostrar que *fromSeq* tiene trabajo $O(n \lg n)$ y profundidad $O(\lg^2 n)$.

2. Las siguientes funciones serán de utilidad para calcular el camino más corto de un vértice a otro en un grafo. Definirlas según las especificaciones dadas.
Suponer que los grafos están representados como tablas de adyacencias, es decir que $\mathit{Graph}\ V = \mathit{Table}\ V\ \mathbb{S}_V$, donde cada vértice del grafo está asociado en la tabla al conjunto de sus vecinos.

a) $\mathit{makeGraph} : \mathit{Seq}\ (a, a) \to \mathit{Graph}\ a$, que dada una secuencia de lados construya un grafo dirigido con los mismos. Definir *makeGraph* con los siguientes costos $W(n) \in O(n \lg n)$ y $S(n) \in O(\lg^2 n)$.

b) $\mathit{numEdges} : \mathit{Graph}\ a \to \mathit{Int}$ y $\mathit{numVertices} : \mathit{Graph}\ a \to \mathit{Int}$, que calculan la cantidad de lados y vértices de un grafo respectivamente.

c) $\mathit{outEdges} : \mathit{Graph}\ a \to a \to \mathbb{S}_{(a,a)}$, que dados un grafo g y un vértice v devuelve el conjunto de lados que contiene todos los lados que tienen como primer componente v, y el conjunto vacío si v no es un vértice de g.

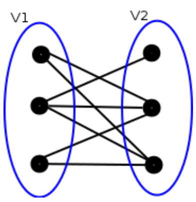
d) $\mathit{makeTree} : \mathit{Graph}\ a \to a \to \mathit{Seq}\ \mathbb{S}_a$, que dados un grafo g y un vértice v construye una secuencia con los vértices correspondientes al árbol de expansión con raíz v que contiene a todos los caminos simples desde v a cualquier vértice w para el cual existe un camino de v a w en g. La secuencia resultado está ordenada de acuerdo a los niveles del árbol, siendo el primer elemento de la secuencia el conjunto $\{v\}$.

Definir *makeTree* con los siguientes costos $W(n) \in O(|E| \lg |V|)$ y $S(n) \in O(D(t) \lg^2 |V|)$, donde E y V son los conjuntos de lados y vértices de g respectivamente, y D(t) es la longitud del más largo de los caminos más cortos.

e) $\mathit{shortestPath} : \mathit{Graph}\ a \to a \to a \to \mathit{Int}$, que dados un grafo y dos vértices v y w del mismo, calcule la longitud del camino más corto de v a w.

La profundidad de *shortestPath* debe ser igual a la de *makeTree*.

3. Un grafo es bipartito si sus vértices pueden separarse en dos conjuntos disjuntos V_1 y V_2 de manera que las aristas sólo conecten vértices de V_1 con vértices de V_2. Es decir, que no existe una arista entre dos vértices de V_1 o dos vértices de V_2. Por ejemplo, el grafo de la figura es bipartito.



Adaptar el algoritmo BFS para definir una función $bipartito : Graph\ V \to V \to Bool$ que dado un grafo conexo y un vértice del mismo determine si el grafo es bipartito o no.
Suponer que los grafos están representados como tablas de adyacencias, es decir que $Graph\ V = Table\ V\ \mathbb{S}_V$, y que se cuenta con una definición de la función $N_G : Graph\ V \to \mathbb{S}_V \to \mathbb{S}_V$ que calcula el conjunto de vecinos para un conjunto de vértices en un grafo (es decir, $N_G(S) = \cup_{v \in S}(getNbrs\ G\ v)$).

4. Dado un grafo g, una *componente conexa* de g es un subgrafo g' de g que contiene todos los lados que forman parte de algún camino que comienza en v, donde v es un vértice cualquiera de g'. De manera que un grafo es conexo si y solo si contiene una sola componente conexa.
Un algoritmo paralelizable para calcular las componentes conexas de un grafo consiste en:

1. Construir los conjuntos de vértices de cada componente conexa.

   Para ello definir una función $verticesComp : Graph\ a \to \mathbb{S}_{\mathbb{S}_V}$, que dado un grafo divida el conjunto de vértices del mismo en partes de manera que cada subconjunto sea el conjunto de vértices de una componente conexa.

2. A partir del grafo y de los vértices correspondientes a sus componentes construir las componentes conexas del grafo.

Definir una función que dado un grafo calcule sus componentes conexas utilizando el algoritmo dado.