Universidad Nacional de Cuyo
Facultad de Ingeniería
Licenciatura en Ciencias de la Computación

Trabajo Práctico Nº 5

Algoritmos y Estructuras de Datos

Grafos

2024

Gonzalo Padilla Lumelli

Mayo 2024

Parte 1

A partir de la siguiente definición:

Graph = Array(n, LinkedList())

Donde Graph es una representación de un grafo simple mediante listas de adyacencia resolver los siguiente ejercicios.

Ejercicio 1

Implementar la función crear grafo que dada una lista de vértices y una lista de aristas cree un grafo con la representación por Lista de Adyacencia.

def createGraph(List, List)

Descripción: Implementa la operación crear grafo.
Entrada: LinkedList con la lista de vértices y la LinkedList con la lista de aristas donde por cada par de elementos representa una conexión entre dos vértices.
Salida: Retorna el nuevo grafo.

Solución

Ejercicio 2

Implementar la función que responde a la siguiente especificación.

def existPath(Grafo, v1, v2)

Descripción: Implementa la operación existe camino que busca si existe un camino entre los vértices v1 y v2.
Entrada: Grafo con la representación de Lista de Adyacencia, v1 y v2 vértices en el grafo.
Salida: retorna True si existe camino entre v1 y v2, False en caso contrario.

Solución

Ejercicio 3

Implementar la función que responde a la siguiente especificación.

def isConnected(Grafo)

Descripción: Implementa la operación es conexo.
Entrada: Grafo con la representación de Lista de Adyacencia.
Salida: retorna True si existe camino entre todo par de vértices, False en caso contrario.

Solución

Ejercicio 4

Implementar la función que responde a la siguiente especificación.

def isTree(Grafo)

Descripción: Implementa la operación es árbol.
Entrada: Grafo con la representación de Lista de Adyacencia.
Salida: Retorna True si el grafo es un árbol.

Solución

Ejercicio 5

Implementar la función que responde a la siguiente especificación.

def isComplete(Grafo)
Descripción: Implementa la operación es completo.
Entrada: Grafo con la representación de Lista de Adyacencia.
Salida: Retorna True si el grafo es completo.

Nota: tener en cuenta que un grafo es completo cuando existe una arista entre todo par de vértices.

Solución

Ejercicio 6

Implementar una función que dado un grafo devuelva una lista de aristas que si se eliminan el grafo se convierte en un árbol. Respetar la siguiente especificación.

def convertTree(Grafo)
Descripción: Implementa la operación convertir a árbol.
Entrada: Grafo con la representación de Lista de Adyacencia.
Salida: LinkedList de las aristas que se pueden eliminar y el grafo resultante se convierte en un árbol.

Solución

Parte 2

Ejercicio 7

Implementar la función que responde a la siguiente especificación.

def countConnections(Grafo)
Descripción: Implementa la operación cantidad de componentes conexas.
Entrada: Grafo con la representación de Lista de Adyacencia.
Salida: Retorna el número de componentes conexas que componen el grafo.

Solución

Ejercicio 8

Implementar la función que responde a la siguiente especificación.

def convertToBFSTree(Grafo, v)
Descripción: Convierte un grafo en un árbol BFS.
Entrada: Grafo con la representación de Lista de Adyacencia, v vértice que representa la raíz del árbol.
Salida: Devuelve una Lista de Adyacencia con la representación BFS del grafo recibido usando v como raíz.

Solución

Ejercicio 9

Implementar la función que responde a la siguiente especificación.

def convertToDFSTree(Grafo, v)

Descripción: Convierte un grafo en un árbol DFS.
Entrada: Grafo con la representación de Lista de Adyacencia, v vértice que representa la raíz del árbol.
Salida: Devuelve una Lista de Adyacencia con la representación DFS del grafo recibido usando v como raíz.

Solución

Ejercicio 10

Implementar la función que responde a la siguiente especificación.

def bestRoad(Grafo, v1, v2)

Descripción: Encuentra el camino más corto, en caso de existir, entre dos vértices.
Entrada: Grafo con la representación de Lista de Adyacencia, v1 y v2 vértices del grafo.
Salida: Retorna la lista de vértices que representan el camino más corto entre v1 y v2. La lista resultante contiene al inicio a v1 y al final a v2. En caso que no exista camino se retorna la lista vacía.

Solución

Ejercicio 11 (Opcional)

Implementar la función que responde a la siguiente especificación.

def isBipartite(Grafo)

Descripción: Implementa la operación es bipartito.
Entrada: Grafo con la representación de Lista de Adyacencia, v1 y v2 vértices del grafo.
Salida: Retorna True si el grafo es bipartito.

NOTA: Un grafo es bipartito si no tiene ciclos de longitud impar.

Solución

Ejercicio 12

Demuestre que si el grafo G es un árbol y se le agrega una arista nueva entre cualquier par de vértices se forma exactamente un ciclo y deja de ser un árbol.

Solución

Ejercicio 13

Demuestre que si la arista (u,v) no pertenece al árbol BFS, entonces los niveles de u y v difieren a lo sumo en 1.

Solución

Parte 3

Parte 4

Parte 5