Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación distribuida

Práctica 3

Recorridos

Baños Mancilla Ilse Andrea - 321173988

Flores Arriola Edson Rafael - 423118018

Rivera Machuca Gabriel Eduardo - 321057608

Reporte de práctica

En esta práctica se implementaron los algoritmos distribuidos de Breadth-First Search (BFS) y Depth-First Search (DFS) utilizando la librería SimPy para la simulación de eventos discretos. Cada nodo del grafo fue modelado como un proceso independiente que recibe y envía mensajes a través de canales.

NodoBFS

Este algoritmo desarrollado está basado en el algoritmo proporcionado por el ayudante en el PDF, por lo que se dará un pequeño análisis.

Primero se simuló el proceso *START().* Para esto el nodo distinguido genera un mensaje especial *START_MSG* indicando que este nodo debe empezar la exploración y lo mete a su propio *canal_entrada* para simular que él mismo se da la señal de arranque.

Después se implementó la parte de recibir un mensaje *GO()*. Sea un nodo que recibe el mensaje, se tienen los siguientes casos:

- Si el padre es nulo, entonces el nodo va a tomar como padre al emisor del mensaje y se actualiza la distancia a *d+1.* El nodo calcula sus vecinos (sin contar al padre) y su cantidad. Si no tiene vecinos entonces regresa el mensaje *BACK_MSJ(YES)* al padre. Si no ocurre esto, entonces el nodo envía el mensaje que recibió a todos sus vecinos.
- Si el padre no es nulo, pero la distancia es menor, actualiza el padre anterior por el que mandó el mensaje, inicializa sus hijos , actualiza la distancia a *d+1* y hace el mismo proceso con sus vecinos que se hace en el caso anterior.
- Si no ocurre ninguno de los casos anteriores , envía el mensaje *BACK_MSJ(NO)* al nodo emisor.

Finalmente se implementó la parte de recibir un mensaje *BACK_MSJ().* Si la respuesta fue *YES* entonces agrega al nodo emisor a sus hijos y decrementa el contador de mensajes esperados.
Si ya no se están esperando mensajes

- Si tiene padre entonces envía el mensaje *BACK()* a su padre

- Si no tiene padre, entonces es la raíz y termina.

NodoDFS

Este algoritmo , al igual que el anterior, está basado en el algoritmo proporcionado por el ayudante en el PDF, por lo que se dará un pequeño análisis.

Primero se simuló el proceso *START().* Para esto el nodo distinguido genera un mensaje especial *START_MSG* indicando que este nodo debe empezar la exploración y lo mete a su propio *canal_entrada* para simular que él mismo se da la señal de arranque.

Después se implementó la parte de recibir un mensaje *START()*. Si un nodo recibe este mensaje, se reconoce a sí mismo como la raíz del árbol, inicializa su lista de hijos vacía y crea el conjunto *completed_children* vacío, donde luego irá guardando los vecinos que ya fueron explorados. Además el nodo raíz crea un mensaje de exploración y lo envía a su primer vecino para iniciar el recorrido.

Después se realizó la parte de recibir un mensaje *GO().*

- Si el padre es nulo, entonces el nodo va a tomar como padre al emisor del mensaje e inicializa su lista de hijos como el vacío y si lista de hijos completos como el nodo emisor. En caso de no tener vecinos no visitados, envía mensaje *BACK_MSG(YES)* al emisor, pero si aun tiene vecinos sin visitar, les envía mensaje *GO()*.
- Si el padre no es nulo, envía mensaje *BACK_MSG(NO)* al emisor.

Finalmente se implementó la parte de recibir un mensaje *BACK_MSJ().*

- Si la respuesta fue *YES* entonces agrega al nodo emisor a sus hijos.
- Si ya no tiene vecinos no visitados y si el nodo es la raíz entonces termina. Si ya no tiene vecinos no visitados pero no es la raíz entonces envía el mensaje *(BACK_MSG(YES)* a su padre.

Si ya no se están esperando mensajes

- Si no pasa alguno de los casos anteriores entonces se calculan los vecinos que no fueron visitados y se les envía el mensaje *GO()*