Universidad Nacional Autónoma de México
Facultad de Ciencias
Práctica 3 ReadMe
Cervantes Duarte Jose Fernando 422100827
Morales Chaparro Gael Antonio 320076972
Rivera Lara Sandra Valeria 320039823
11 de octubre de 2024

1 Explicación del Algoritmo BFS de la clase `NodoBFS`

Nuestra implementación del algoritmo BFS sin terminación utiliza mensajes para la comunicación entre nodos del ambiente. Describiremos los diferentes casos con los que se puede encontrar un nodo o proceso al momento de ejecutarse el algoritmo:

1.1 Inicialización del Nodo Raíz

El nodo raíz, el cual por convención es el nodo con `id_nodo = 0`, es el responsable de iniciar la ejecución del algoritmo. Este nodo envía un mensaje `GO()` a sí mismo con un valor inicial de distancia igual a `-1`.

1.2 Recepción de Mensajes `GO()`

Cuando un nodo recibe un mensaje `GO()` de uno de sus vecinos, tenemos dos casos:

- Si el nodo aún no ha sido asignado a ningún padre (es decir, `padre = -1`), significa que este es el primer mensaje `GO()` que recibe. El nodo que envió el mensaje se convierte en su *padre*, su lista de *hijos* se inicializa como vacía, y su nivel en el árbol BFS se establece como la distancia recibida más uno (`distancia = d + 1`). Además, el nodo determina cuántos mensajes `BACK()` debe esperar de sus vecinos restantes.
- Si el nodo ya tiene un padre pero recibe un mensaje con una distancia menor a la que actualmente tiene asignada, significa que ha encontrado un camino más corto hacia el nodo raíz. En este caso, el nodo actualiza su padre, su nivel de distancia, y reinicia su contador de mensajes `BACK()` esperados.

1.3 Envío de Mensajes `GO()` a los Vecinos

Una vez que un nodo se une al árbol, verifica si hay más vecinos a los cuales aún no ha enviado un mensaje `GO()`. Si existen vecinos pendientes, el nodo envía un nuevo mensaje `GO()` a esos vecinos con la distancia incrementada en uno (`d + 1`). Si no quedan más vecinos a los cuales enviar mensajes, el nodo envía un mensaje `BACK()` a su *padre* indicando que ha terminado la propagación.

1.4 Recepción de Mensajes `BACK()`

Cuando un nodo recibe un mensaje `BACK()` de uno de sus vecinos, debe determinar si ese vecino se ha unido al árbol BFS:

- Si el mensaje `BACK()` indica que el vecino es parte del árbol (`resp = yes`), el nodo lo agrega a su lista de *hijos*.
- En cualquier caso, el nodo disminuye su contador de mensajes `BACK()` esperados. Una vez que ha recibido todos los mensajes esperados, envía un mensaje `BACK()` a su *padre*.

1.5 Finalización del Algoritmo

El algoritmo finaliza cuando el nodo raíz ha recibido todos los mensajes `BACK()` esperados de sus vecinos. Aquí se considera que el árbol BFS está completamente construido y se notifica que el proceso ha terminado.

2 Explicación del Algoritmo DFS en la Clase `NodoDFS`

Nuestra implementación de DFS sin terminación simula una búsqueda en profundidad sobre una red de nodos. Lo hacemos de la siguiente forma:

2.1 Inicialización del Nodo Raíz

El nodo inicial o proceso distinguido, identificado por `id_nodo = 0` (esto por convención), es el responsable de iniciar la exploración. Este proceso hace:

- Envía un mensaje `VISITED()` a todos sus vecinos, informando que ha sido visitado.
- Selecciona el menor de sus vecinos no visitados mediante la función `menor_numero()` (esto es un detalle de implementación que es equivalente a elegir un vecino aleatorio cada vez), lo agrega como hijo y le envía un mensaje `GO()` para iniciar su exploración.

2.2 Recepción de Mensajes `GO()`

Cuando un nodo recibe un mensaje `GO()`, hace que el nodo que envió el mensaje sea su *padre* y después:

- Si todos sus vecinos ya han sido visitados, el nodo envía un mensaje `VISITED()` a todos sus vecinos y un mensaje `BACK()` a su *padre*, indicando que ha completado el recorrido.
- Si hay vecinos que no han sido visitados, selecciona el menor de esos vecinos utilizando `menor_numero()` y le envía un mensaje `GO()`. Luego, actualiza su lista de *hijos* para reflejar esta conexión.

2.3 Envío y Recepción de Mensajes `BACK()`

Cuando un nodo recibe un mensaje `BACK()`, verifica si todos sus vecinos han sido visitados:

- Si es así y el nodo no es el nodo raíz, envía un mensaje `BACK()` a su *padre*, indicando que ha terminado el recorrido.
- Si aún quedan vecinos por explorar, selecciona el menor de los vecinos no visitados, lo agrega como hijo, y le envía un mensaje `GO()`. El nodo también actualiza su lista de nodos visitados enviando un mensaje `VISITED()` a todos sus vecinos.

2.4 Recepción de Mensajes `VISITED()`

Los mensajes `VISITED()` permiten a los nodos actualizar su lista de vecinos ya visitados. Si un nodo recibe este mensaje de un vecino, lo agrega a su lista `visitados`, haciendo que no lo vuelva a explorar eventualmente.

2.5 Finalización del Algoritmo

El proceso DFS finaliza cuando el nodo raíz ha recibido mensajes `BACK()` de todos sus hijos. En este punto, se considera que todo el árbol DFS ha sido explorado. La terminación del algoritmo es señalada con un mensaje de finalización y la estructura completa del árbol ha sido descubierta.

2.6 Funciones Auxiliares

Nos apoyamos de varias funciones auxiliares en el algoritmo:

- `menor_numero(lista)`: Busca y devuelve el menor número de una lista dada.
- `eliminar_elementos(lista_principal, elementos_a_eliminar)`: Elimina de `lista_principal` todos los elementos presentes en `elementos_a_eliminar`.