UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Tarea 4

INTEGRANTES

Torres Valencia Kevin Jair - 318331818
Aguilera Moreno Adrián - 421005200
Natalia Abigail Pérez Romero - 31814426

PROFESOR

Miguel Ángel Piña Avelino

AYUDANTE

Pablo Gerardo González López

ASIGNATURA

Computación Distribuida

8 de octubre de 2022

1. Describe un algoritmo distribuido basado en DFS que cuente el número de procesos en un sistema distribuido cuya gráfica G es arbitraria. Al terminar de contar, debe informar a todos los procesos el resultado del conteo. Muestra que es correcto.
Caso base. Sea G una gráfica tal que $V_G = \{p_1\}$, por la linea 8 $contarProcesosDFS() = 1$ lo cual es correcto.
Hipótesis de inducción Para cualquier gráfica G con n vértices `contarProcesosDFS()` cuenta la cantidad procesos de G
Al inicio de la ejecución **Paso inductivo** Por demostrar que dada una gráfica G con $n+1$ vértices `asignarEtiquetasDFS()` asigna etiquetas únicas en el rango $[1, \ldots, n+1]$.

Algorithm 1 contarProcesosDFS(ID,soyLider)

```
Padre = ⊥
Hijos = ∅
numProcesos = 0
SinExplorar = todos los vecinos
Si no he recibido algún mensaje
if soyLider and Padre == ⊥ then
    Padre = ID
    numProcesos = 1
    explore(numProcesos)
end if
Al recibir < numP > desde el vecino p_j:
if Padre == ⊥ then
    Padre = j
    numProcesos = numP + 1
    elimina p_j de SinExplorar
    explore(numProcesos)
else
    if numProcesos < numP then
        numProcesos = numP
    end if
    send(< already >) a p_j:
    elimina p_j de SinExplorar
end if
Al recibir < already > desde p_j
explore(numProcesos)
Al recibir < parent, numP >
if numProcesos < numP then
    numProcesos = numP
end if
Hijos ∪p_j
explore(numProcesos)
procedure EXPLORE(numP)
    if SinExplorar ≠ ∅ then
        elegir p_k en SinExplorar
        eliminar p_k de SinExplorar
        send (< numP >) a p_k
    else
        if Padre ≠ ID then
            send(< parent, numP >) a Padre
        end if
        if Padre == ID and numProcesos < numP then
            numProcesos = numP
        end if
    end if
end procedure
```

2. Describe un algoritmo distribuido basado en DFS que, en una gráfica arbitraria G con n vértices anónimos, asigne etiquetas únicas en el rango $[1, ..., n]$ a los vértices de G. Muestra que es correcto.
Hint: Puedes suponer que cada proceso conoce a sus vecinos aunque estos no tengan una etiqueta explicita.

Algorithm 2 asignarEtiquetasDFS()

```
1: Al inicio seleccionar un nodo al azar para empezar a enviar mensajes
2: Hijos = ∅
3: ID = 1
4: Padre = ID
5: SinExplorar = todos los vecinos
6: etiqueta = ID
7: EXPLORE(< etiqueta, 1 >)
8: Al recibir < etiqueta, i > desde el vecino p_j:
9: if Padre == ⊥ then
10:     Padre = j
11:     ID = i
12:     elimina p_j de SinExplorar
13:     EXPLORE(etiqueta,i)
14: else
15:     send(< already >) a p_j:
16:     elimina p_j de SinExplorar
17: end if
18: Al recibir < already > desde p_j
19: EXPLORE(etiqueta)
20: Al recibir < parent, i > desde p_j
21: Hijos ∪p_j
22: EXPLORE(etiqueta,i)
23: procedure EXPLORE(etiqueta,i)
24:     if SinExplorar ≠ ∅ then
25:         k = i, j = 0
26:         while j < |SinExplorar| do
27:             k+ = 1
28:             send (< etiqueta, k >) a p_j
29:             eliminar p_j de SinExplorar
30:             j+ = 1
31:         end while
32:     else
33:         if Padre ≠ ID then
34:             send(< parent, etiqueta >) a Padre
35:         end if
36:     end if
37: end procedure
```

Por demostrar que el algoritmo es correcto, es decir, dada una gráfica arbitraria G con n vértices anónimos, asigne etiquetas únicas en el rango $[1, \ldots, n]$ a los vértices de G Demostración por inducción:
Demostración por inducción sobre n número de vértices anónimos en G.

Caso base. Sea G una gráfica tal que $V_G = \{p_1\}$, por la linea 3 el $ID(p_1) = 1$ por lo tanto todos los vértices de G tiene una etiqueta única y conoce las etiquetas de sus vecinos.

Hipótesis de inducción Para cualquier gráfica G con n vértices `asignarEtiquetasDFS()` asigna etiquetas únicas en el rango $[1, \ldots, n]$. Al inicio de la ejecución el nodo elegido tiene $ID(p_1) = 1$ luego envía $< etiqueta, 1 >$ a uno de sus vecinos p_j este asigna $ID(p_j) = 2$ si tiene vecinos diferentes del líder, es decir $SinExplorar \neq \emptyset$ de manera similar asigna un nuevo ID una unidad mayor, es decir $ID(p_{j+1}) = ID(p_j) + 1$, cuando el proceso p_j ya no tenga vecinos por explorar le envía el mensaje $< parent, i >$ donde la i es la última etiqueta asignada, cuando los vértices reciben $< parent, i >$ desde p_j guardan este vecino en el conjunto de sus hijos y envían la etiqueta i a su padre, cuando $Padre == ID$ termina la asignación de ID de los vecinos de uno de los hijos del proceso con $ID = 1$ y procede a enviar $< etiqueta, i >$, donde i es el último ID asignado, a otro de los hijos del proceso con $ID = 1$, de manera similar recorre los vecinos asignando un ID mayor al que recibieron. Hasta que el proceso con $ID = 1$ no tenga vértice por explorar el algoritmo termina.

Paso inductivo

Por demostrar que dada una gráfica G con $n + 1$ vértices `asignarEtiquetasDFS()` asigna etiquetas únicas en el rango $[1, \ldots, n + 1]$.

Sea un vértice p_{j+1} que no tiene etiqueta, pero sabemos que es vecino de al menos un vértice p_j, que por hipótesis inductiva tiene una etiqueta al igual que sus vecinos excepto p_{j+1}, por lo tanto SinExplorar de p_j es $\{p_{j+1}\}$ y por la linea 28 en el procedimiento `EXPLORE` P_{j+1} recibe el mensaje:$< etiqueta, i >$, asigna a p_j como su padre y $ID(p_{j+1}) = ID(p_j) + 1$, por lo tanto todos los vértices $n + 1$ de G tienen una etiqueta única en el rango $[1, \ldots, n + 1]$.

3. Modifica el algoritmo DFS para que se ejecute en tiempo a lo más $2|V|$ y no mande más de $2|E|$ mensajes, suponiendo que las aristas son bidireccionales.
Hint: Cuando un proceso recibe el mensaje `M` por primer vez, este notifica a todos sus vecino pero envía el mensaje a sólo uno de ellos.

4. Considera el algoritmo 1, que calcula una $\Delta + 1$ coloración, donde Δ es el grado máximo en la gráfica. Muestra una gráfica G con al menos 10 vértices y una asignación de IDs, donde el algoritmo coloree todos los procesos (el primer momento en el que todas las variables c son distintas de $\perp$) en tiempo $diam(G)$. Muestra otra asignación de IDs para las que el algoritmo coloree en tiempo a los más $diam(G)/2$.

Pseudocódigo 1: $\Delta + 1$ coloración

```
Algoritmo coloring(ID):
  c = ⊥

Ejecutar inicialmente:
  send(⟨ID,c⟩)

Al tener todos los mensajes de todos mis vecinos en t ≥ 1:
  Mensajes = ⟨ID_1,c_1⟩,...⟨ID_j,c_j⟩  \(j\) = grado maximo en la grafica
  A = {ID_i | c_i = ⊥}
  if c == ⊥ ∧ ID = máx(A ∪ {ID}) then
    c = mín({1,...,Δ+1} \ {c_i ≠ ⊥})
  send(⟨ID,c⟩) a todos los vecinos
```

▷ Para este problema dividamos la solución en 3 respectivas soluciones:

1. Mostrar una ejecución en tiempo `diam`(G), con G una gráfica tal que $|V_G| = 10$.

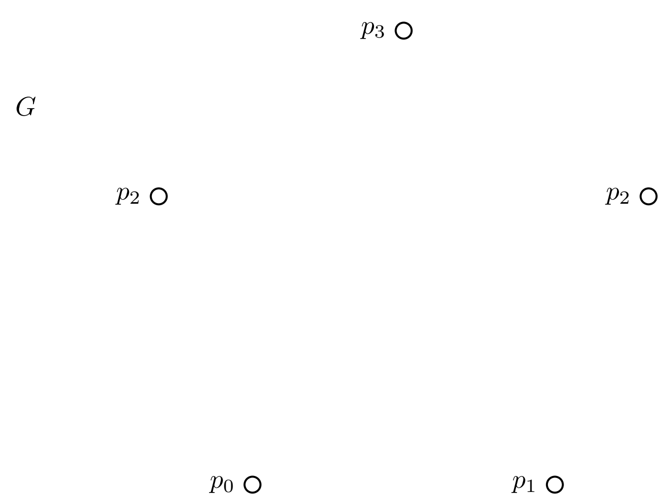

Figura 1: Gŕagica G.

◁

5. Un toro $n \times m$ es una versión dos dimensional de un anillo, donde un nodo en la posición (i, j) tiene un vecino hacia el norte en $(i, j-1)$, al este en $(i+1, j)$, al sur en $(i, j+1)$ y al oeste en $(i-1, j)$. Esos valores se calculan módulo n para la primera coordenada y módulo m para la segunda; de este modo $(0, 0)$ tiene vecinos $(0, m-1)$, $(1, 0)$, $(0, 1)$ y $(n-1, 0)$. Supongamos que tenemos una red síncrona de paso de mensajes en forma de un toro $n \times m$, consistente de procesos anónimos idénticos, los cuáles no conocen n, m o sus propias coordenadas, pero tienen sentido de la dirección (es decir, puede decir cual de sus vecinos está al norte, este, etc.). **Pruebe o refute**: Bajo estas condiciones, ¿existe un algoritmo determinista que calcule cuando $n > m$?