UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS
COMPUTACIÓN DISTRIBUIDA
TAREA 2

Nombre: 2025-1
Nombre:

EJERCICIO 1. Sea $G = (\Pi, E)$ un sistema distribuido síncrono. Sea τ un árbol BFS con raíz en el proceso $p_s \in \Pi$. Demuestra que existe una ejecución del algoritmo para crear árboles generadores que tiene como resultado τ con p_s el proceso que inicia la ejecución. Hint: Recuerda que en un árbol BFS, un proceso a distancia d de la raíz G, está a distancia d en τ.

EJERCICIO 2. Considere un sistema distribuido $G = (\Pi, E)$ representado como un árbol. Diseña un algoritmo distribuido que determine la altura del árbol, es decir, la longitud del camino más largo desde la raíz a cualquier hoja del árbol.

OBSERVACIÓN. Se puede suponer que cada proceso p_i conoce su proceso padre en el árbol, y el proceso raíz conoce su identificador único.

EJERCICIO 3. Sea $G = (\Pi, E)$ un sistema distribuido asíncrono. Considera un algoritmo para construir un árbol BFS en el que sólo la raíz sabe que ya terminó la construcción del árbol. Diseña un algoritmo que le permita a cada proceso saber que ya terminó la construcción del árbol y que la raíz sepa que ya todos saben que se construyó el árbol. Argumentar por qué es correcto y la complejidad del número de mensajes y tiempo.

EJERCICIO 4. Consideremos un sistema distribuido $G = (\Pi, E)$ representado como un árbol. Supongamos que cada proceso p_i tiene un identificador único. Diseña un algoritmo distribuido para determinar el proceso más alejado de la raíz del árbol y la distancia de la raíz a ese proceso.

OBSERVACIÓN. Puede asumir que el proceso raíz conoce su identificador único, y cada proceso p_i conoce su proceso padre en el árbol.

EJERCICIO 5. Sea $G = (\Pi, E)$ un sistema distribuido síncrono. Sea $M = \{m_1, m_2, \ldots, m_k\}$ un conjunto de mensajes que se quieren transmitir desde un proceso p_s hacia todos los demás. Debido a que el ancho de banda es muy limitado, no se puede enviar el conjunto M completo, por lo que tiene que enviarse cada m_i por separado. Cada proceso debe saber qué parte del mensaje está recibiendo, es decir que el i-ésimo mensaje es el m_i. Diseña un algoritmo de broadcast para diseminar M sin que se repita la recepción de los mensajes en los procesos, es decir, si p_i ya recibió m_j, no debe volver a recibirlo. Hint: broadcast sobre un árbol toma $O(n)$ mensajes porque no se repite la entrega de los mensajes.

EJERCICIO 6. Sea $G = (\Pi, E)$ un sistema distribuido síncrono y sea $p_s \in \Pi$ un proceso que empezará a ejecutar broadcast. Demuestra que broadcast no puede ejecutarse en menos de D

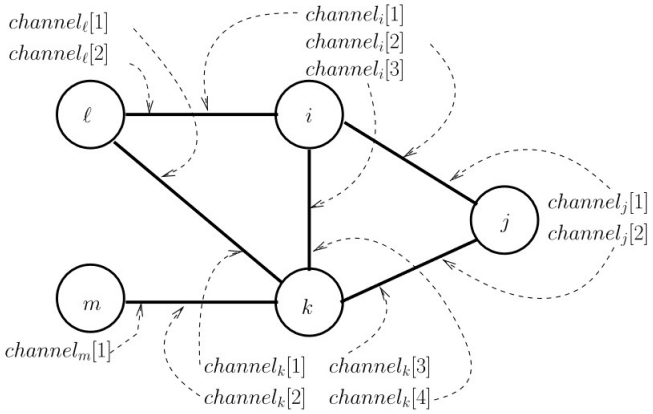
rondas, con D la distancia más grande entre p_s y cualquier otro proceso.

EJERCICIO 7. En la CDMX varios puntos de control (nodos) fueron agregados. Cada uno de estos puntos tiene información sobre el número de personas que viven dentro de cierto diámetro. Un punto (raíz) de la delegación Benito Juárez quiere recolectar la información de todos los puntos de control y determinar cuántas personas viven en la CDMX. Escribe un algoritmo distribuido para que cada nodo recolecte la información de sus hijos para enviársela al padre sin repetirla y que el padre reporte el total de pobladores viviendo en la CDMX.

EJERCICIO 8. Considera el algoritmo de convergecast sobre el árbol generador ya construido. Demuestra que cualquier nodo a altura h (distancia más corta desde la hoja hacia el nodo) envía un mensaje a más tardar en la ronda h.

EJERCICIO 9. Consideremos un gráfica de comunicación en el que los procesos no tienen identidad y cada proceso p_i conoce su posición en la red de comunicación con la ayuda de una matriz local $channel_i[1 \dots c_i]$ (donde c_i es el número de vecinos de p_i). En la siguiente imagen se muestra un ejemplo. Como podemos ver en la imagen, el canal que conecta p_i y p_j es localmente conocido por p_i como $channel_i[2]$ y localmente conocido por p_j como $channel_j[1]$. Utilizando el algoritmo DFS en el que un proceso distinguido recibe un mensaje START(), diseñe un algoritmo que asocie a cada proceso p_i una identidad id_i y una tabla de identidad $neighbor\_name_i[1 \dots c_i]$ tal que:

- $\forall i : id_i \in \{1, 2, \dots, n\}$
- $\forall i, j : id_i \neq id_j$
- Si p_i y p_j son vecinos y el canal que los conecta se denota $channel_i[k]$ en p_i y $channel_j[l]$ en p_j, y tenemos que $neighbor\_name_i[k] = id_j$ y $neighbor\_name_j[l] = id_i$.



EJERCICIO 10. Consideremos el caso de una gráfica de comunicación dirigida donde el significado de 'dirigida' es el siguiente. Un canal de p_i a p_j permite

- Que p_i solo envíe mensajes $GO()$ a p_j
- Que p_j solo envíe mensajes $BACK()$ a p_i

Dos procesos p_i y p_j son entonces de tal manera que o bien no hay un canal de comunicación que los conecte, o hay un canal de comunicación dirigido que conecta a uno con el otro, o hay dos canales de comunicación dirigidos (uno en cada dirección).
Diseña un algoritmo distribuido que construya un árbol generador con una raíz distinguida p_a y compáralo con el algoritmo BFS. Se asume que hay un camino dirigido desde la raíz distinguida p_a a cualquier otro proceso.

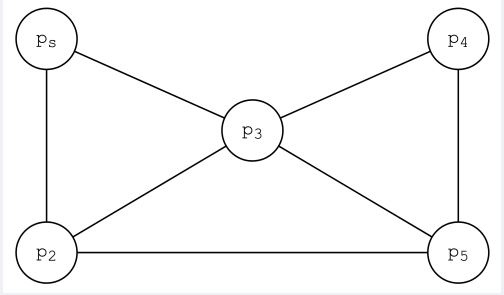
EJERCICIO 11. Considera el algoritmo BFS que no detecta terminación en un sistema síncrono (Figura 3). Sea D la distancia más grande de la raíz a cualquier otro proceso. Demuestra que para cada $1 \leq t \leq D$, después de t rondas, cada vértice p_i a distancia t ya ha recibido un mensaje con $d = t - 1$ de algún vecino p_j y por lo tanto $distance_i = t$ y $parent_i = j$ tal que $distance_j = t - 1$.

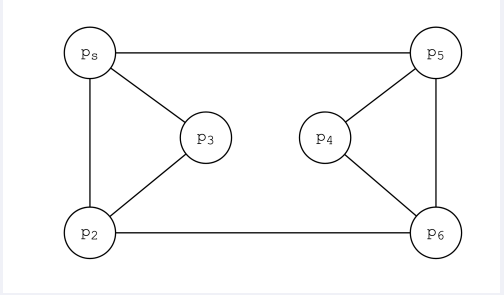
```
1  initially do
2      if pid = initiator then
3          distance ← 0
4          send distance to all neighbors
5      else
6          distance ← ∞
7  upon receiving d from p do
8      if d + 1 < distance then
9          distance ← d + 1
10         parent ← p
11         send distance to all neighbors
```

Figura 3: Algoritmo BFS que no detecta terminación

HINT. En la ronda 0 la raíz comienza su ejecución y envía su mensaje a sus vecinos y estos los reciben en la ronda 1.

EJERCICIO 12. Ejecuta el algoritmo DFS en las siguientes gráficas.





EJERCICIO 13. Consideremos una red lineal, es decir, una colección lineal de n procesos $1, \ldots, n$, donde cada proceso está conectado bidireccionalmente con sus vecinos. Supongamos que cada proceso i puede distinguir su izquierda de su derecha y sabe si es o no un punto final. Supongamos que cada proceso i tiene inicialmente un valor entero muy grande v_i y que puede mantener en memoria sólo un número constante de tales valores en cualquier momento. Diseñe un algoritmo para ordenar los valores entre los procesos, es decir, para hacer que cada proceso i devuelva un valor de salida o_i, donde el subconjunto de salidas es igual al subconjunto de entradas y $o_1 \leq \cdots \leq o_n$. Intenta diseñar el algoritmo más eficiente que puedas tanto en número de mensajes como en número de rondas. Demuestra tus afirmaciones.

EJERCICIO 14. Considere un sistema distribuido $G = (\Pi, E)$ representado como un árbol. Diseña un algoritmo distribuido que determine el diámetro del árbol, es decir, el camino más largo entre dos nodos cualesquiera del árbol.

OBSERVACIÓN. Puede asumir que cada proceso p_i conoce su proceso padre en el árbol, y el proceso raíz conoce su identificador único.

EJERCICIO 15. Demuestra el siguiente lema.

> **LEMA 1.** Si todas las aristas de una gráfica G tienen pesos distintos, entonces existe exactamente un árbol generador mínimo (MST) en G.