Tarea 2: Computación distribuida

- Ángeles Sánchez Aldo Javier 320286144
- Sánchez Victoria Leslie Paola 320170513
- Sosa Romo Juan Mario - 320051926

1. **Determine si existe un algoritmo para el problema de elección de líder en un anillo sincrono en el que todos los procesadores son indistinguibles salvo por uno de ellos (digamos que este cuenta con un id distinto a los demás). Si existe, de el algoritmo y si no, argumente por qué.**

   Consideramos un anillo asíncrono con *n* procesadores, un procesador tendrá un ID distinto a los demás, mientras que los *n-1* procesadores restantes tendrán el mismo ID, lo que los hace indistinguibles.

   Para que exista el algoritmo, entonces el procesador diferente querrá y deberá proclamarse como líder, suponemos sin pérdida de generalidad que el ID del procesador único es mayor a los demás ID's. Entonces nuestro algoritmo elegirá como líder al procesador con ID mayor.

   Comenzamos desde el procesador distinto, éste envía su ID a su vecino derecho y el vecino que recibe el mensaje compara el ID recibido con su ID, como el recibido es mayor, entonces actualiza su líder y envía un mensaje a su vecino derecho con el ID mayor; al finalizar, el procesador con ID distinto recibirá nuevamente su ID y se proclamará como líder, de igual forma, todos los demás procesadores sabrán el ID de su líder.

   El caso con ID menor es análogo.

2. **Investiga y describe el algoritmo de consenso de tu criptomoneda favorita. El algoritmo debe estar redactado con tus propias palabras.**

   Algoritmo de consenso de Dogecoin:

   Para que todos los participantes de la red estén enterados de lo que sucede y verifiquen la información, DOGE utiliza un algoritmo de consenso llamado Proof of Work.

   Los participantes de la red compiten por encontrar un número que solucione problemas matemáticos, entonces aquel participante que solucione el problema debe compartir los resultados para que los demás participantes comprueben la solución. Cuando la solución es verificada, entonces el participante recibe las criptomonedas que generó resolviendo el problema. Los participantes de la red son muy activos y protegen la red, mientras más participantes haya, ésta será más fuerte porque se verificarán más veces las soluciones y evitarán ataques.

De la sección 3.3 del libro de Attiya y Welch:

1. **Demuestre que la complejidad del algoritmo de la sección 3.3.1 es O(n^2)**

Primero hay que recalcar que la complejidad del algoritmo en esta seccion es $O(n^2)$ sobre la cantidad de mensajes. Pero, primero veamos el algoritmo, este es una solución bastante inocente de un algoritmo para elegir un líder en un anillo no anónimo (con id's diferentes), la idea, como visto en clase, es que para cada nodo en el anillo, en el turno inicial mandan su id a su vecino izquierdo y recibe algo de su vecino derecho, checa el mensaje del vecino derecho, si este contiene un id menor al suyo, elimina el mensaje, si por el contrario el mensaje es mayor al suyo, lo pasa a su vecino izquierdo, finalmente si el id corresponde a su propio id, se declara el líder y le dice a su vecino izquierdo que el es líder. Este algoritmo funciona porque en un conjunto finito de números diferentes siempre va a existir un maximal, entonces para cada nodo con un id diferente al maximal (menor) su id se va a perder en alguna ronda antes de llegar a ellos y solo será el mensaje del máximo el que de toda la vuelta y regrese a el.

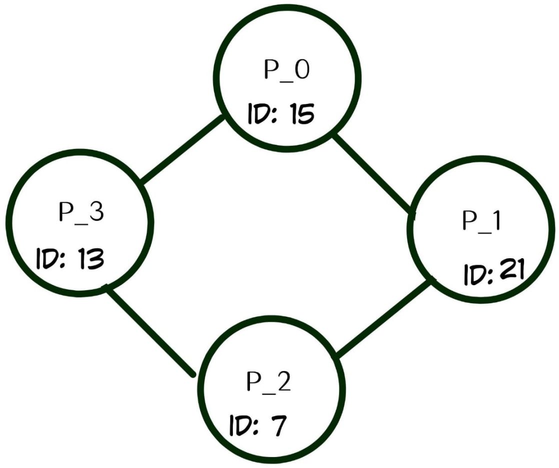
Entonces, en el peor caso, los mensajes van a viajar un montón de veces o lo que es lo mismo reenviarse hasta que un id mayor, que este muy a su izquierda los cancele; el libro ve el caso en donde los nodos están acomodados de manera que todos menos el nodo 0 y el nodo n-1 tengan un nodo con id menor a si mismos a su izquierda y mayor a si mismos a su derecha, de esta forma, el nodo con id n-1 manda su mensaje a su izquierda, este va por el n-2,n-3,...,0 y después le llega por su derecha, pero ahí esta el problema, porque para el id n-1 se ocuparon n mensajes, pero el id n-2 también estaba viajando y como están en orden descendiente recorre n-3,...,0 y cuando llega al n-1 entonces se lo traga, utilizando n-1 mensajes, esto como ya nos dijo Euler es:

$$S = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \in O(n^2)$$

Lo que resulta natural pues se ve como que cada mensaje recorre hacia su izquierda hasta llegar a un nodo mayor a ellos pero están en orden descendente; añadir los n mensajes de terminación que pasa desde el mayor a su izquierda a todos los demás mantiene la complejidad pero cambia un poco las constantes ocultas.

2. **Realice una ejecución del algoritmo de la sección 3.3.2 con 4 procesos.**

Supongamos que tenemos los siguientes procesos:



La k-vecindad de los procesos es k=1. Inicialmente asleep=true para cada proceso. Como aún no se envían mensajes, entonces:

P_0 asleep = false y envía <probe,15,0,1> a P_1 y P_3

P_1 asleep = false y envía <probe, 21,0,1> a P_0 y P_2

P_2 asleep = false y envía <probe,7,0,1> a P_1 y P_3

P_3 asleep = false y envía <probe,13,0,1> a P_0 y P_2

Fase 0

P_0 recibe de la derecha <probe,13,0,1>

¿13=15? No

¿13>15 y 1<2º? No

¿13>15 y 1≥2º? No

El mensaje no se transmite

P_0 recibe de la izquierda <probe,21,0,1>

¿21=15? No

¿21>15 y 1<2º? No

¿21>15 y 1≥2º? Sí

Enviamos <reply,21,0> a la izquierda

P_1 recibe de la derecha <probe,15,0,1>

¿15=21? No

¿$15>21$ y $1<2^0$? No

¿$15>21$ y $1\geq 2^0$? No

El mensaje no se transmite

P_1 recibe de la izquierda <probe,7,0,1>

¿7=21? No

¿$7>21$ y $1<2^0$? No

¿$7>21$ y $1\geq 2^0$? No

El mensaje no se transmite

P_2 recibe de la derecha <probe,21,0,1>

¿21=7? No

¿$21>7$ y $1<2^0$? No

¿$21>7$ y $1\geq 2^0$? Sí

Enviamos <reply,21,0> a la derecha

P_2 recibe de la izquierda <probe,13,0,1>

¿13=7? No

¿$13>7$ y $1<2^0$? No

¿$13>7$ y $1\geq 2^0$? Sí

Enviamos <reply,13,0> a la izquierda

P_3 recibe de la derecha <probe,13,0,1>

¿7=13? No

¿7>13 y 1<2º? No

¿7>13 y 1≥2º? No

El mensaje no se transmite

P_3 recibe de la izquierda <probe,15,0,1>

¿15=13? No

¿15>13 y 1<2º? No

¿15>13 y 1≥2º? Sí

Enviamos <reply,15,0> a la izquierda

P_1 recibió <reply,21,0> de la derecha

21=21 pero aún no se ha recibido mensaje del lado izquierdo

P_1 recibió <reply,21,0> de la izquierda

21=21 y ya se recibió un mensaje del lado derecho entonces enviamos <probe,21,1,1> a sí mismo

P_3 recibió <reply,13,0> de la derecha

13=13 pero aún no se ha recibido mensaje del lado izquierdo

P_0 recibió <reply,15,0> de la derecha

15=15 pero aún no se ha recibido mensaje del lado izquierdo

Fase 1

P_1 recibió <probe,21,1,1>

21=21 entonces P_1 es líder del anillo.