Tarea 5

Clase: Computación distribuida
Profesora: Karla Vargas

Cruz Perez Ramon -315008148
Marco Antonio Orduña Avila -315019928

1. (2ptos) Considera el siguiente algoritmo distribuido para un sistema **síncrono** que calcula la la distancia entre la raíz de una gráfica y el nodo que se está visitando.

Algoritmo 1: Calcula distancia desde la raíz

```
neighbors_i = {Conjunto de vecinos de i};
distance_i;
inicio
    si p_s = p_i entonces
        distance_i = 0;
        para cada j en neighbors_i hacer
            send distance_i + 1 to p_j
        fin
    en otro caso
        distance_i = ∞;
    fin
fin
When: distance_j es recibido de p_j
inicio
    si distance_j < distance_i entonces
        distance_i = distance_j;
        para cada k en neighbors_i hacer
            send distance_i + 1 a p_k
        fin
    fin
fin
```

Demuestra por **inducción** que si un proceso está a distancia k de la raíz , entonces al finalizar la ronda k debe tener $distance_i = k$.
Considera los siguientes puntos para la demostración :

- Sobre qué estás haciendo la inducción.
- Cuál es tu hipótesis, base y paso inductivo.
- Argumentar correctamente en el paso inductivo.
- Al final escribir cuál es la conclusión de tu demostración.

Si alguno de los puntos anteriores no está en tu demostración se bajaran puntos.

Por induccion/ronda:
/ronda, Pues es la variable en la cual nos va a decir la distancia.
Caso base:

- ronda = 0,
  Si Ps esta solo, se cumple.

- Sea Ps y un vecino p_j .
  ronda 0,
  ps inicia:
  distancia = 0
  envia(0+1,p_j)
  $distancia_j = \infty$

  ronda = 1
  .- recibe distancia de Ps:
  $distancia_j < distancia_{ps}$ entonces $distancia_j = 1$
  $\implies distancia_j = 1 =$ ronda,
  $\implies$ se cumple.

H.I.
Si un proceso está a distancia k de la raíz , entonces al finalizar la ronda k debe tener $distance_i = k$.

P.I
Si un proceso está a distancia $k + 1$ de la raíz , entonces al finalizar la ronda $k + 1$ debe tener $distance_i = k + 1$.

Sea p_j,p_i quien envia y recibe respectivamente:
ronda = k,
p_j recibe y envia sus vecinos (distancia + 1),
Con distancia = k,
Por Hip. de induccion en la ronda k, se cumple que p_j esta a distancia k.

ronda= k+1,
.- p_i recibe distancia= k+1 de P_j:

$distancia_j < distancia_i$,
entonces $distancia_i$ = k+1
No puede tener otra distancia, pues como estamos en sistema sincrono todos los mesajes llegan en tiempo y forma.
Por Caso base, si mandan una distancia en la ronda anterior, le llega con una unidad mas.
Entonces la distancia de $distancia_i$ = k+1
$\Longrightarrow$ El proceso p en la ronda k+1 tiene distancia = k+1,

$\Longrightarrow$ Si un proceso está a distancia k de la raíz , entonces al finalizar la ronda k debe tener $distance_i = k$.

2. (3ptos) Escribe el protocolo de consenso para sistemas asíncronos sin fallas

Usaremos la primitiva de **wait()** until condition. Para esperar a que todos los mensajes lleguen, pues no hay fallas.
Usaremos **coord$_i$.** Para saber quien es el coordinador hasta el momento.

Algoritmo 2: Consenso asicrono sin fallas

```
[1] V_i = v_i
coord_i = 1
inicio
    para j in neighbors_i hacer
        send V_i to p_j
    let rec_i = wait() Hasta rec_i = ||neighbors_i|| //recibe todos los mensajes
    most_freq_i = mas frecuente en rec_i
    occ_nb_i = numero de occurencias de most_freq_i
inicio
    si i == coord_i entonces
        para j in neighbors_i hacer
            send most_freq_i to p_j
    si v = wait() Hasta coord_i == p_coord entonces
        coord_val_i = v
    en otro caso
        coord_va_i = v_i
    coord_i++ //para saber quien el siguiente coordinador
    si occ_nb_i > n/2 entonces
        V_i= most_freq_i
    en otro caso
        V_i =coord_va_i
    si coord_i == max(neighbors_i) entonces
        return V_i
return (V_i)
```

Si i es el ultimo coordinador(el que tenga el id maximo de los vecinos), entonces termina la ejecucion.

3. (2.5ptos) Explica con tus propias palabras por qué el consenso no puede

ser resuelto en sistemas asíncronos con al menos una falla

Como no se sabe que proceso va a fallar, no se sabe cuanto tiempo hay que esperar a cada proceso o que tan lento es. Es poco eficiente estar esperando a un proceso del cual nunca vamos a recibir un mensaje, por lo que si falla uno, solo nos quedariamos esperando la eternidad y nunca llegaría el mensaje.
Seria como predecir el futuro de todos los procesos para saber cual o cuales van a fallar.

4. (2.5ptos) Explica con tus propias palabras por qué el problema del ataque coordinado no puede resolverse con $n \geq 2$

Como los canales de comunicacion no son confiables, existen varios casos; Que cada proceso i envia un mensaje y no llegar al j , pero luego i envie otro mensaje a j, esto puede confundir al proceso j y no llegar a un acuerdo.
O puede que llegue el caso en que un proceso se quede esperando y nunca recibe el mensaje, y no se cumpla la Terminacion.
Lo dificil de esto es saber si el proceso tiene los mismos mensajes que el otro.