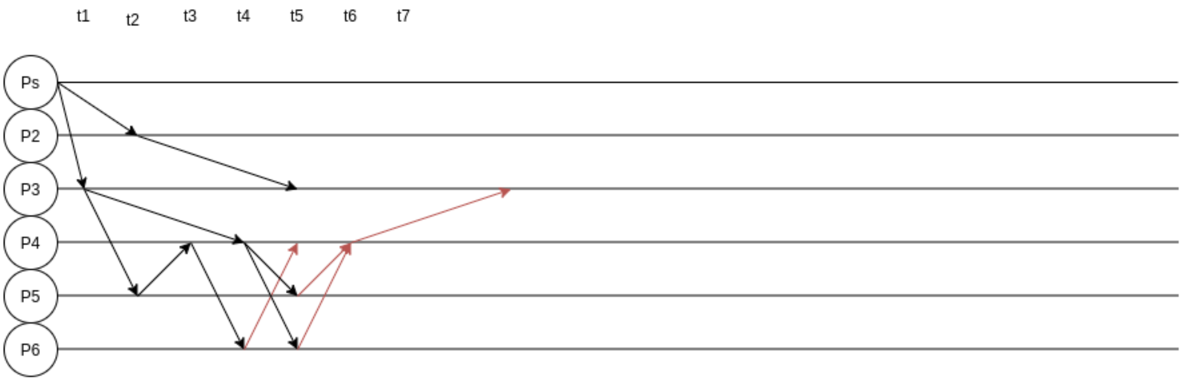
Tarea 3

Cruz Perez Ramon -315008148
Marco Antonio Orduña Avila -315019928

Octubre 2020

1. **Ejecuta el algoritmo BFS de la figura 1.11 [libro de M. Raynal] en la siguiente grafica**



Ps parent= ps, d = 0, msg = 2, Go(0),
P2 parent= ps, d = 1, msg = 1, Go(1)
P3 parent= p3, d = 1, msg = 3, Go(1)
P5 parent= p3, d = 2, msg = 1, Go(2)
P4 parent= p5, d = 3, msg= 2, Go(3)

P4 parent= p3, d = 2, msg= 1, Go(2)
P6 parent= p4, d = 4 back(yes,4)
P5 parent= p3, d = 2 back(no,2)

back(yes,4); P4 parent= p3, d = 2, msg= 1, ...

P6 parent= p4, d = 3 back(yes,3)

back(yes,3); P4 parent= p3, d = 2, msg= 0, child ={p6}
back(yes,2); P3 parent= ps, d = 2, msg= 1, child ={p4}

.... P5 se queda esperando un mensaje de 4.

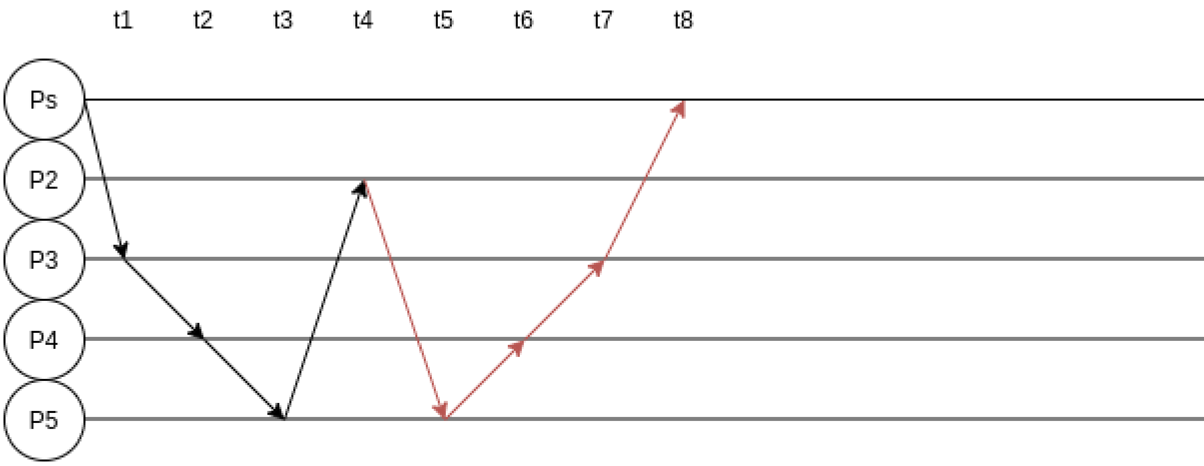
Debido a que en el tiempo t=4, P4 recibe un GO() de P3, con menor distancia, se actualiza su padre, y deja de ser P5, pero en ese momento P5 sigue esperando un mensaje de p4, y como nunca llega no es posible completar la grafica.
Para solucionar esto, solo falta que en la linea (19) el **if** necesita un **else**: que donde **if(resp = no)** Entonces seria igual a la lineas 21 - 24.

2. **¿Se puede obtener mas de un arbol BFS en la grafica anterior?**

   Suponiendo que el algorimo esta arreglado:
   No, pues siempre que se encuentre un camino mas corto se va a actualizar, no importa si se cambian los pesos de las aristas.

3. **Ejecuta el algoritmo DFS de la figura 1.17 [libro de M. Raynal] en la siguiente grafica**

   Todo empieza en Ps, eligiendo k = P3.



Ps: k = P3, child={P3}
P3: parent = Ps, visted ={Ps}, k = P4, GO({Ps,P3}),child={P4}
P4: parent = P3, visted ={Ps,P3}, k = P5, GO({Ps,P3,P4}),child={P5}
P5: parent = P4, visted ={Ps,P3,P4}, k = P2, GO(visted U{P5}),child={P2}
P2: parent = P5, visted ={Ps,P3,P4,P5}, BACK(visted U{P2}), child= {0}
P5: parent = P4, visted ={Ps,P3,P4,P5,P2}, BACK(visted)
P4: parent = P3, visted ={Ps,P3,P4,P5,P2}, BACK(visted)
P3: parent = Ps, visted ={Ps,P3,P4,P5,P2}, BACK(visted)
Ps: parent = Ps, visted ={Ps,P3,P4,P5,P2}, BACK(visted)
fin de ejecucion

4. **Considera el algoritmo BFS que no detecta terminacion en un sistema sıncrono (Figura 3). Sea D la distancia mas grande de la raız a cualquier otro proceso. Demuestra que para cada 1 t D, despues de t rondas, cada vertice pi a distancia t ya ha recibido un mensaje con d = t − 1 de algun vecino pj y por lo tanto distancei = t y parenti = j tal que distancej = t**

— 1 (Hint: en la ronda 0 la raız comienza su ejecuci´on y env ´ıa su mensaje a sus vecinos y estos los reciben en la ronda 1).

Por Induccion sobre t:
D = Distancia mas grande.
1 ≤ t ≤ D, en t rondas, cada vertice p_i a t distancia ya recibio un mesaje d = t -1 para algun vecino p_j
Por lo tanto: D=t, parent = j, D_j = t-1

Caso Base:
G = raiz y un hijo:
t = 0, raiz, D = 0, send (D) to vecinos,
t = 1, como d+1 < distance
distance = 1, parent = raiz
$\Longrightarrow$ D = t, parent = raiz, $D_{raiz} = 0$

H.I.
1 ≤ t ≤ D, en t rondas, cada vertice p_i a t distancia ya recibio un mesaje d = t -1 para algun vecino p_j
Por lo tanto: D=t, parent = j, D_j = t-1

P.I.
1 ≤ t+1 ≤ D, en t+1 rondas, cada vertice p_i a $t+1$ distancia ya recibio un mesaje d = (t+1) -1 para algun vecino p_j
Por lo tanto: D=t+1, parent = j, D_j = (t+1) -1

ronda = t+1
Upon receiving (d) **from** p_j **do**
if (d < distance):
distance$_i$ = d+1, parent$_i$ = p_j,
como p_j es el padre y envio **send** en la ronda t.
Por HI. p_j, D_j= t , parent = x cualquiera, D_x= t -1
como j es el padre,
$\Longrightarrow$ distance$_i$ = t +1 y parent$_i$ = j, D_j = (t+1) -1