Algoritmos y Programación Paralela

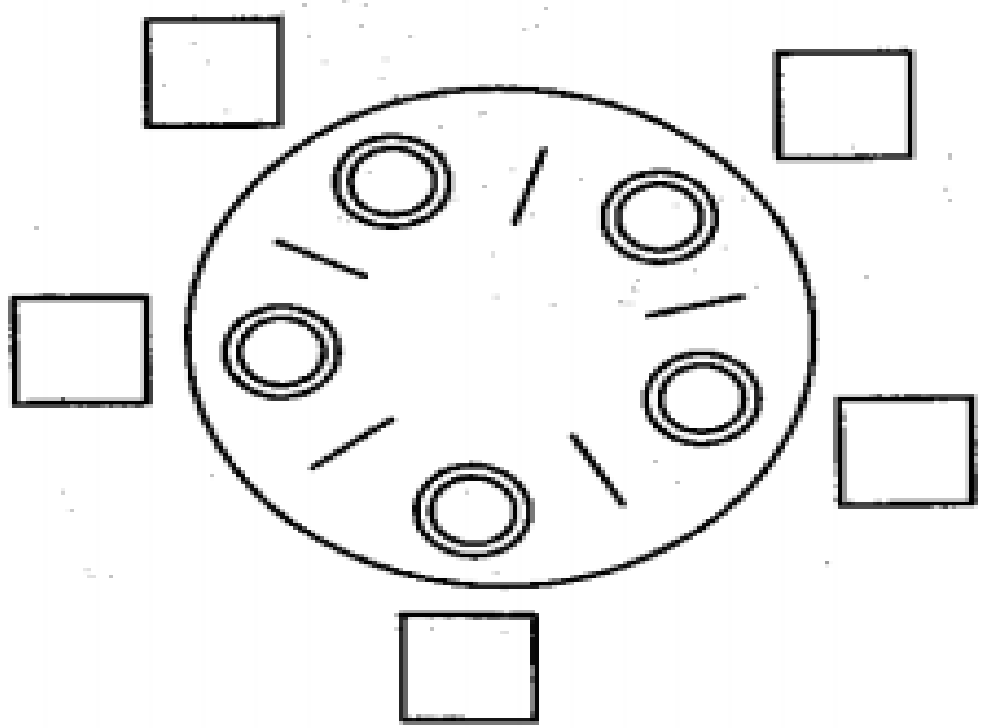
Dra. Ing. Ana Cori Morón

EL PROBLEMA DE LOS FILOSOFOS COMENSALES

- Problema propuesto por Edger Dijkstra, es un problema clásico porque sirve para ilustrar los problemas del interbloqueo ya que las condiciones para que se pueda producir interbloqueo están presentes en su enunciado, el cual dice:

EL PROBLEMA DE LOS FILOSOFOS COMENSALES

Cinco filósofos dedican su vida a pensar y comer, los filósofos comparten una mesa redonda de cinco sillas. En la mesa hay además cinco palillos y cinco platos. Un filósofo para comer hará uso de un plato y de dos palillos (el de su izquierda y el de su derecha).



La solución al problema consiste en elaborar el algoritmo que permita comer a los filósofos sin que se interrumpan entre ellos

CONDICIONES

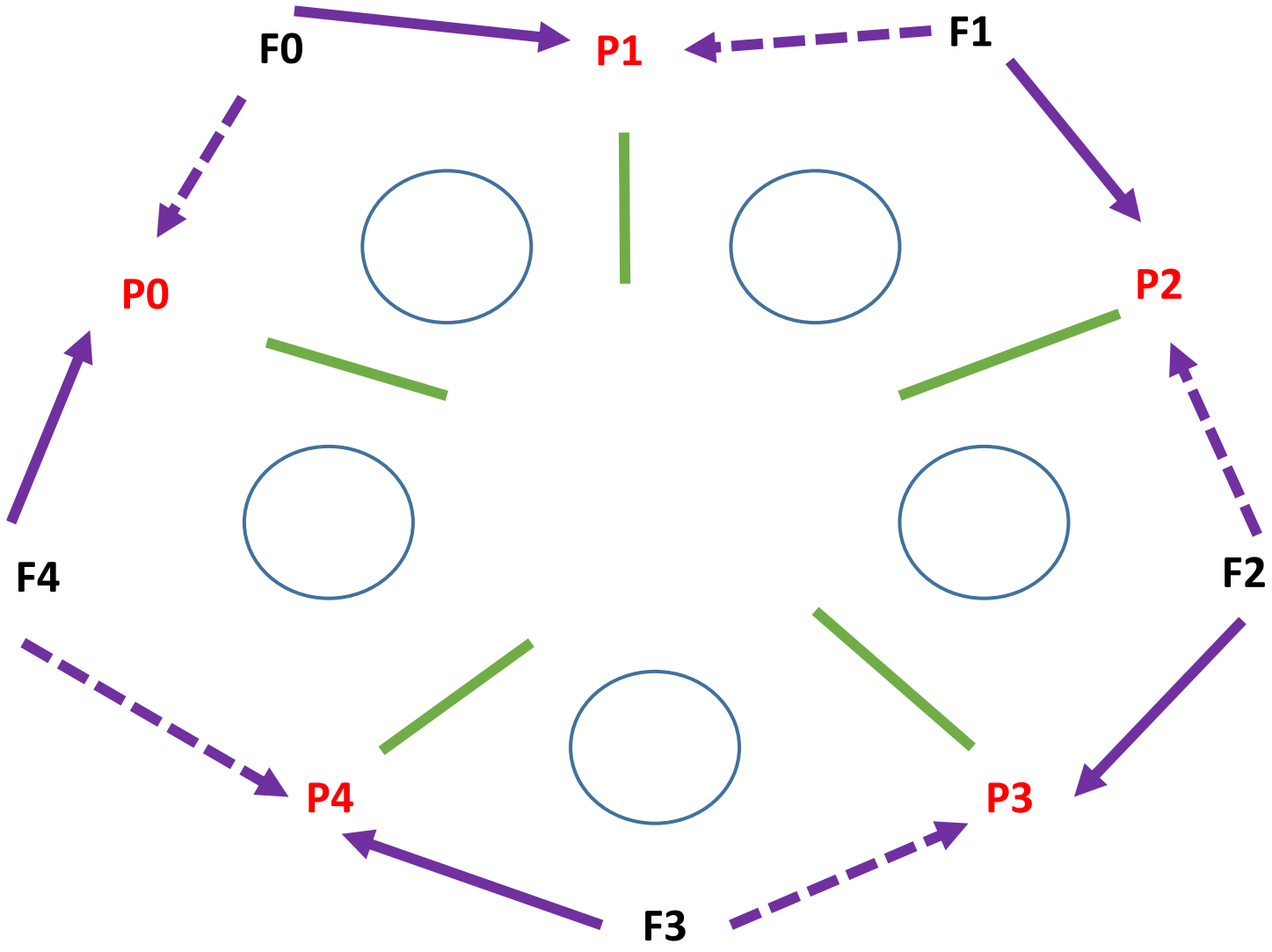
- EXCLUSIÓN MUTUA, dos filósofos contiguos no pueden comer a la vez
- SINCRONIZACION, si un filosofo esta comiendo, los contiguos no pueden hacerlo hasta que termine
- El filosofo que termina de comer debe ceder los palillos para su posterior utilización
- Si dos filosofos contiguos van a coger los palillos uno de ellos debe hacerlo.
- Todos los filósofos que quieran comer tienen que poder hacerlo en algún momento finito o morirán.

PRIMERA SOLUCIÓN

- Esta primera solución consiste en representar cada palillo como un semáforo, cuando un filosofo trata de coger un palillo ejecuta la operación WAIT sobre el semáforo.
- Estructuras:
  - P: array[0..4] of semaphore
- Inicialización
  - For k:=0 to 4 do
    - Initial (p[k],1);

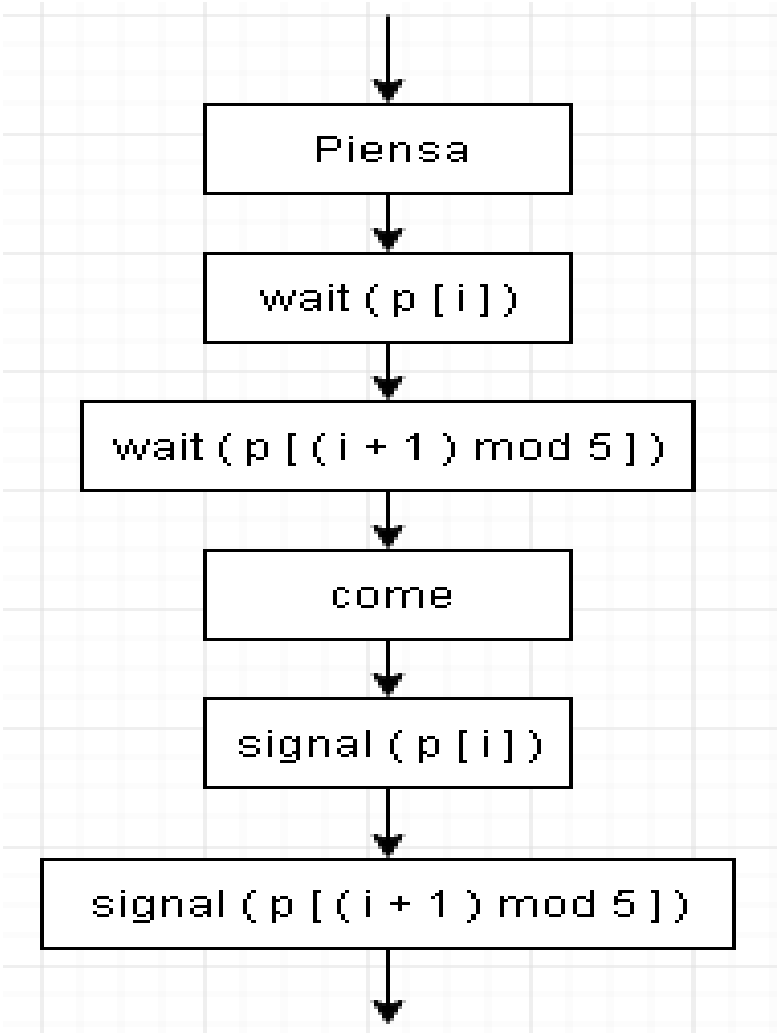
p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1

Cogen palillo de la izquierda
Luego el palillo de la derecha



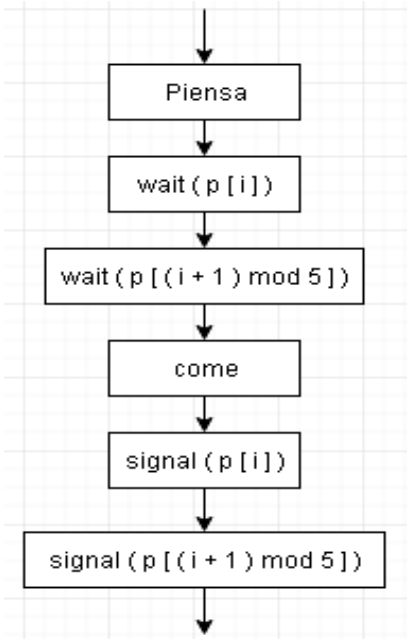
Sin embargo esta solución no es correcta, ya que si todos los procesos dejan de pensar al mismo tiempo, y cogen los palillos de su izquierda se produce un interbloqueo de todos los filósofos

process type filosofo[i]

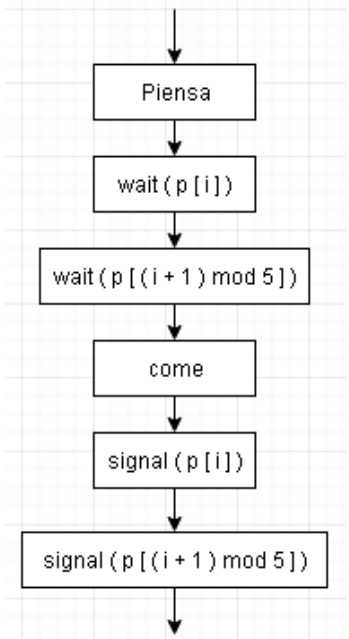


P[0]	P[1]	P[2]	P[3]	P[4]
0 ó 1	0 ó 1	0 ó 1	0 ó 1	0 ó 1
Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]

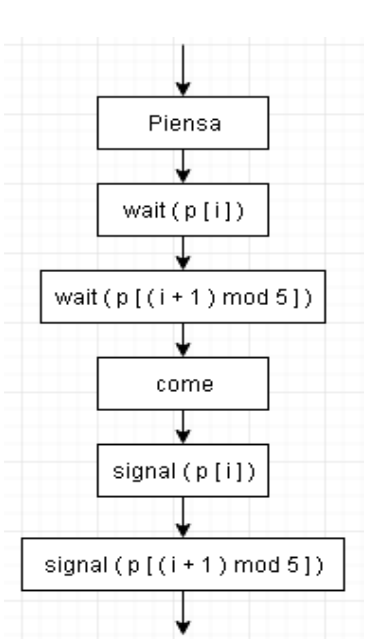
Filosofo[0]



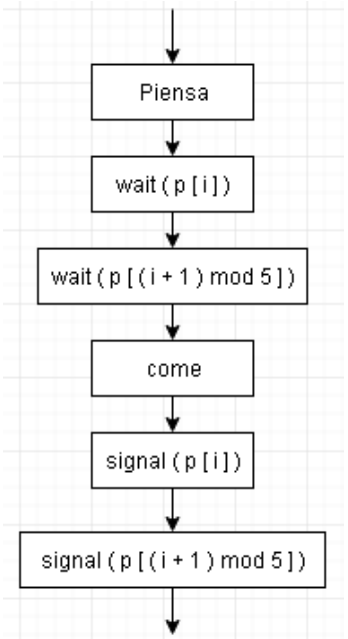
Filosofo[1]



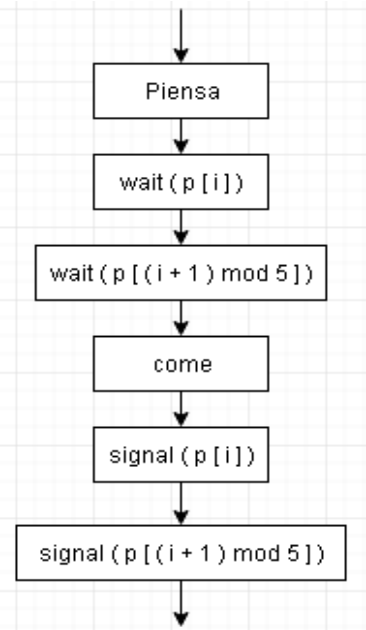
Filosofo[2]



Filosofo[3]



Filosofo[4]



p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1

EJECUCIÓN

T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
i													
P[0]													
P[1]													
P[2]													
P[3]													
P[4]													
F[0]													
F[1]													
F[2]													
F[3]													
F[4]													

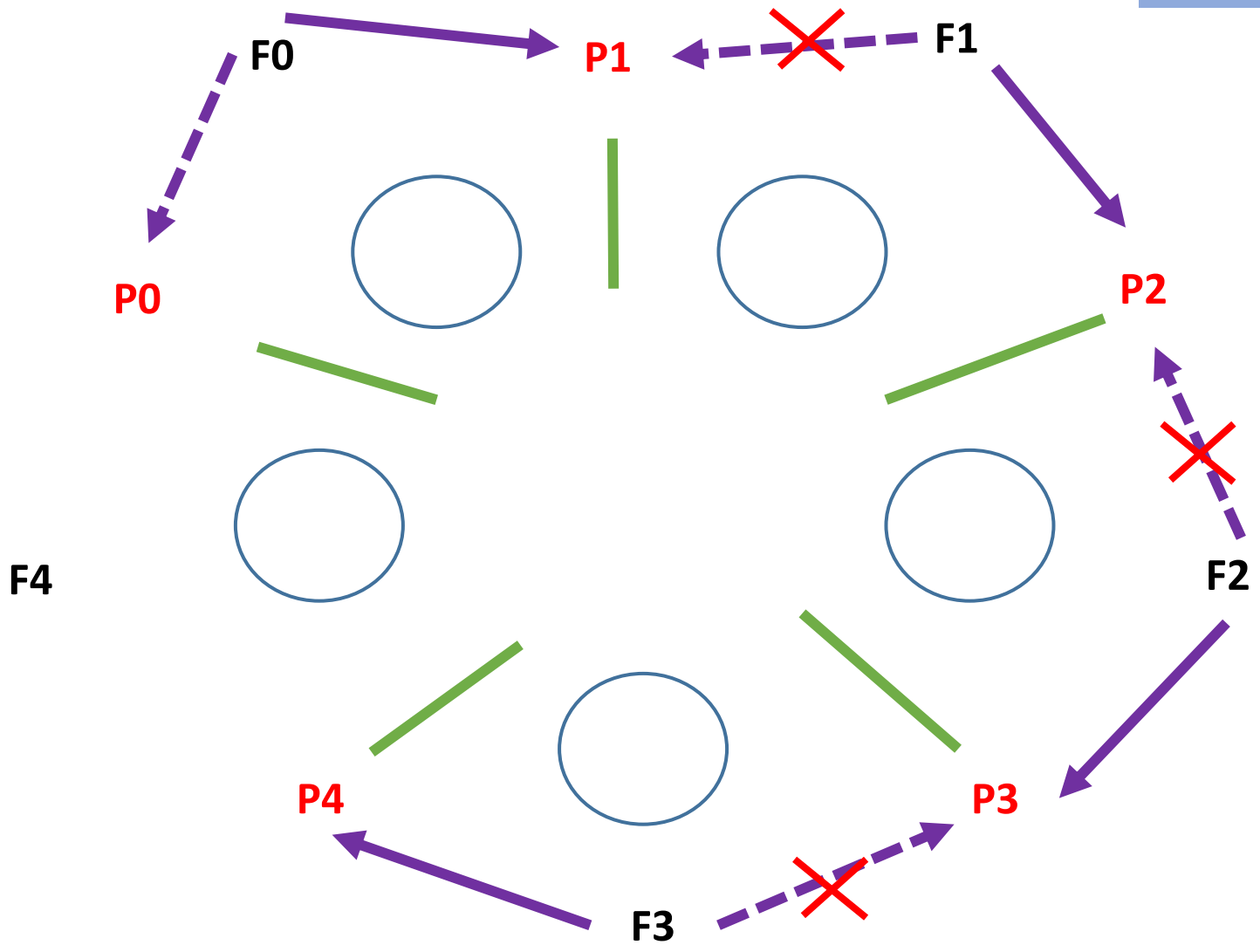
ELABORAR LA EJECUCIÓN MANUAL

SEGUNDA SOLUCION

- Consiste en inicializar la cantidad de filósofos que pueden comer simultáneamente a 4. Debido a que no se podrán ocupar los 5 palillos, no será posible un interbloqueo total.
- Estructuras:
  - P: array[0..4] of semaphore
  - sitios: semaphore
- Inicialización
  - Initial (sitios,4) {semáforo del recurso compartido}
  - For k:=0 to 4 do
    - Initial (p[k],1);

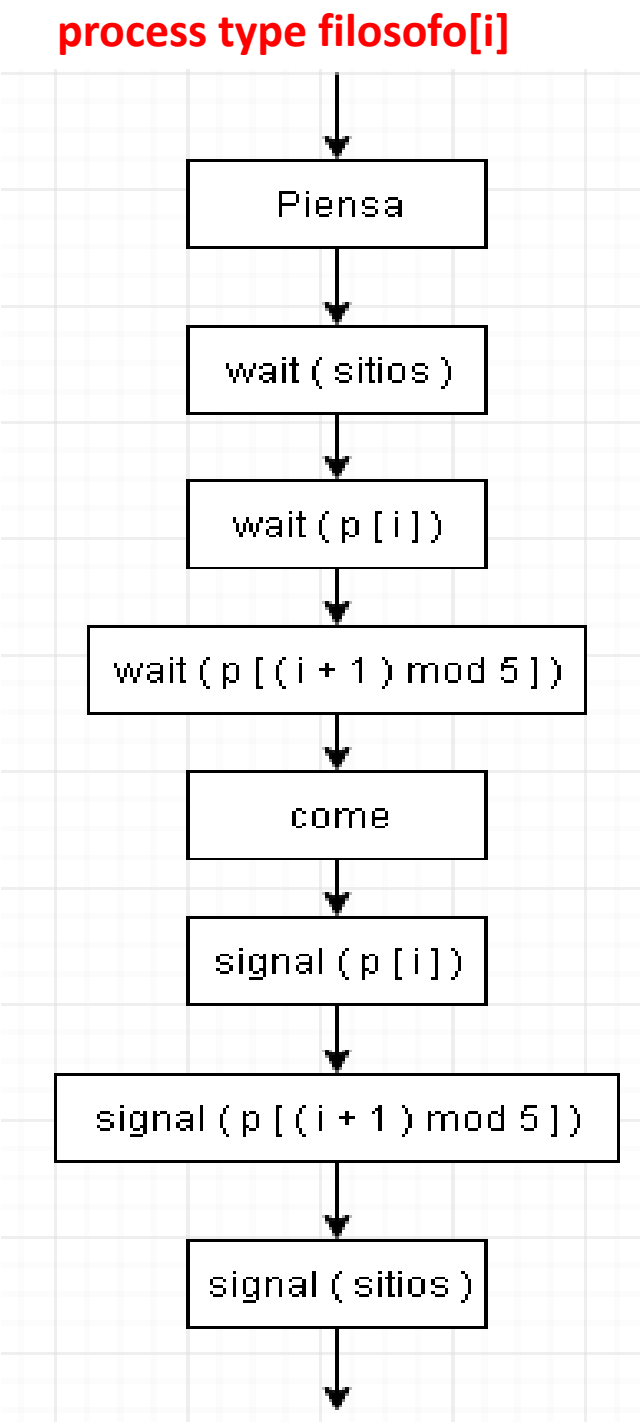
Sitios=4
p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1

Cogen palillo de la izquierda
Luego el palillo de la derecha

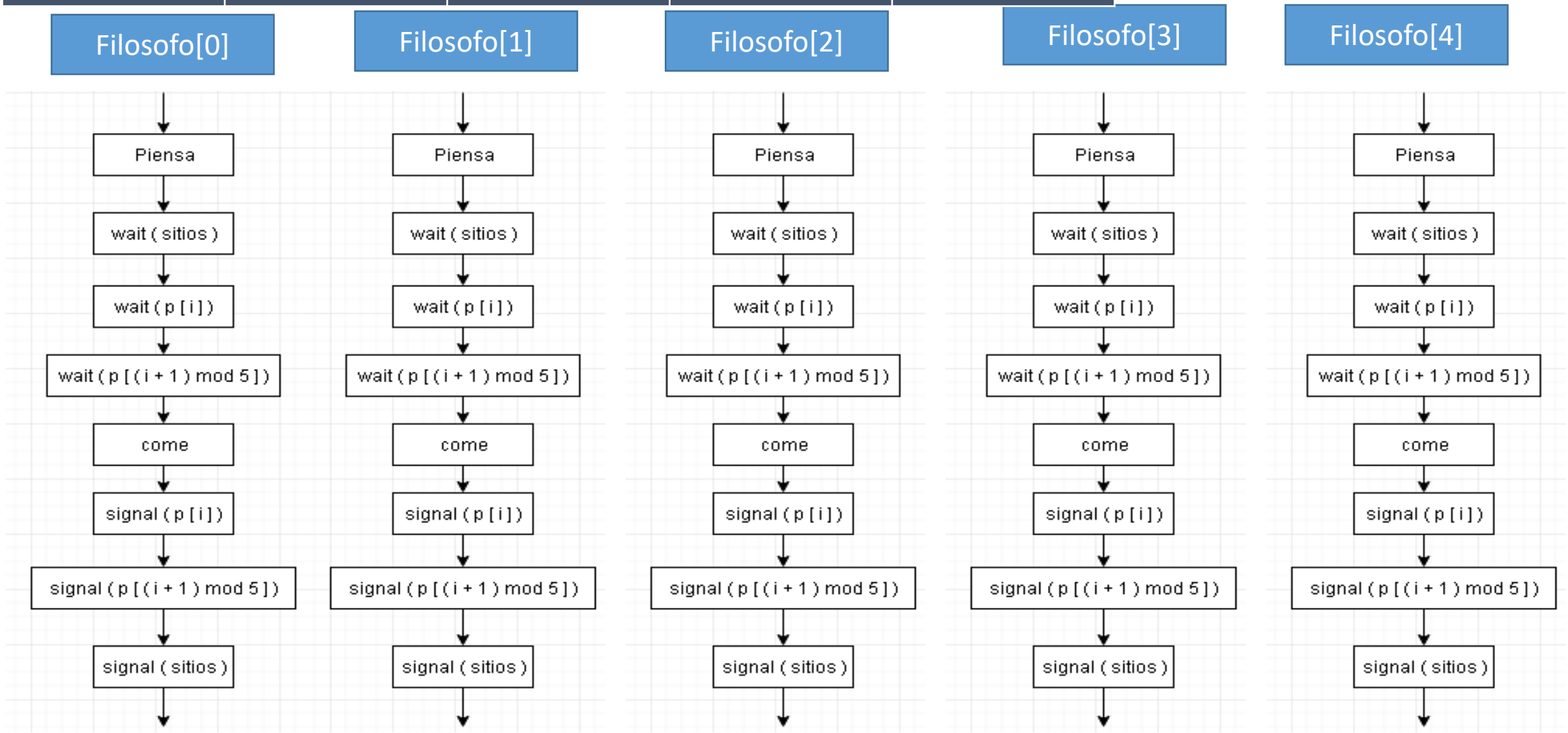


Este algoritmo hace que sea vuelva mas lento el proceso, pero garantiza que al menos un filosofo puede comer.

DIAGRAMA DE FLUJO



P[0]	P[1]	P[2]	P[3]	P[4]
0 ó 1	0 ó 1	0 ó 1	0 ó 1	0 ó 1
Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]



Sitios=4
p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1

EJECUCIÓN

T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
i													
sitios													
P[0]													
P[1]													
P[2]													
P[3]													
P[4]													
F[0]													
F[1]													
F[2]													
F[3]													
F[4]													

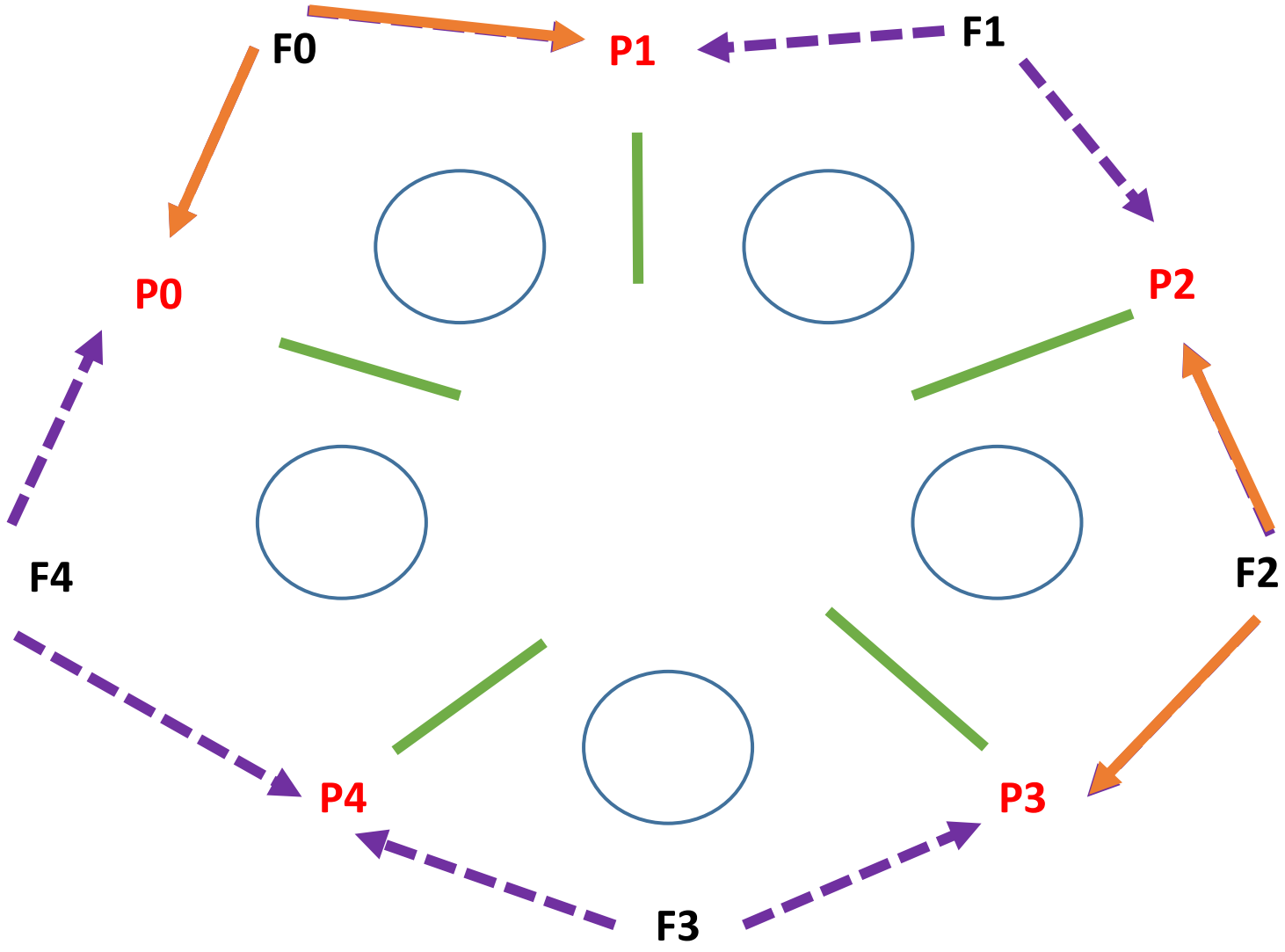
ELABORAR LA EJECUCIÓN MANUAL

TERCERA SOLUCION

- Esta solución consiste en que cada filósofo tome sus palillos solamente si ambos están libres.
- Estructuras:
  - P: array[0..4] of integer {puede tomar dos valores los palillos; 1=libre y 0=ocupado}
  - mutex: semaphore
- Inicialización
  - Initial (mutex,1)
  - for k:=0 to 4 do
    - p[k]=1;

mutex=1
p[0]=1 {libre}
p[1]=1 {libre}
p[2]=1 {libre}
p[3]=1 {libre}
p[4]=1 {libre}

Cogen palillo de la derecha
Luego el palillo de la izquierda

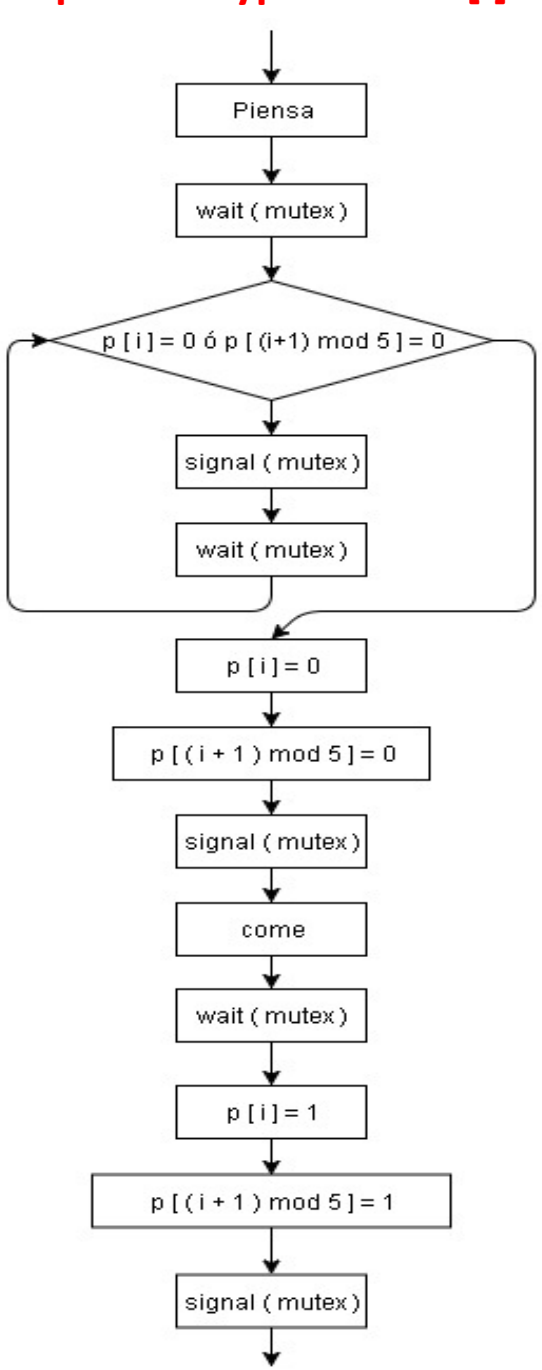


P[0]=0 {ocupado}
P[1]=0

P[2]=0
P[3]=0

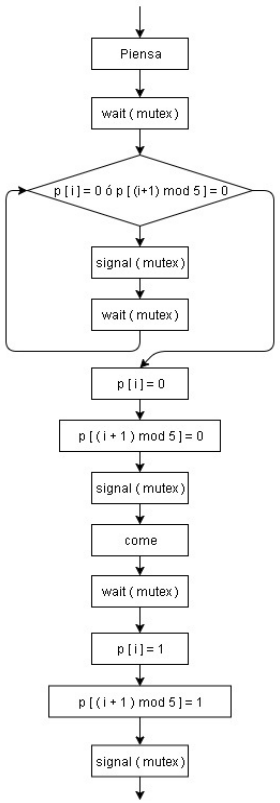
Este algoritmo se basa en espera ocupada, no es eficiente

DIAGRAMA DE FLUJO

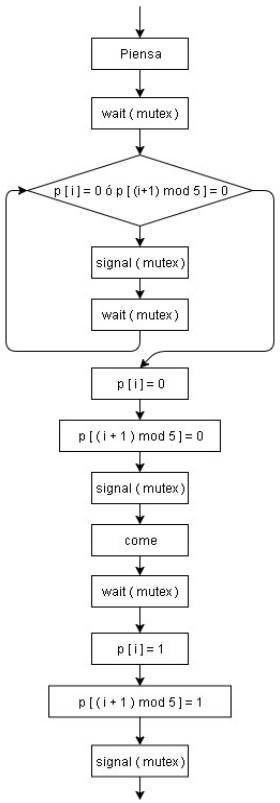


P[0]	P[1]	P[2]	P[3]	P[4]
0 ó 1	0 ó 1	0 ó 1	0 ó 1	0 ó 1

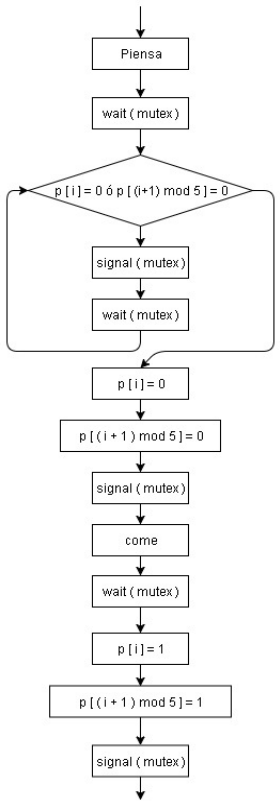
Filosofo[0]



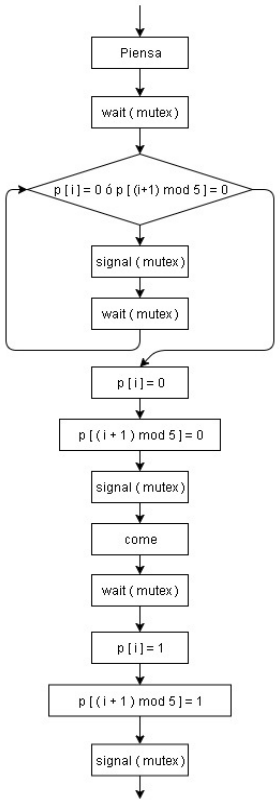
Filosofo[1]



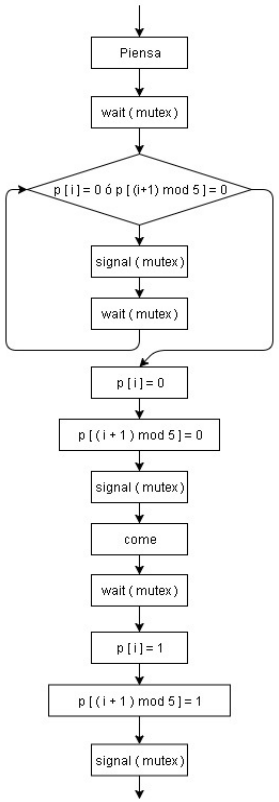
Filosofo[2]



Filosofo[3]



Filosofo[4]



EJECUCIÓN

mutex=1
p[0]=1 {libre}
p[1]=1 {libre}
p[2]=1 {libre}
p[3]=1 {libre}
p[4]=1 {libre}

T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
i													
mutex													
P[0]													
P[1]													
P[2]													
P[3]													
P[4]													
F[0]													
F[1]													
F[2]													
F[3]													
F[4]													

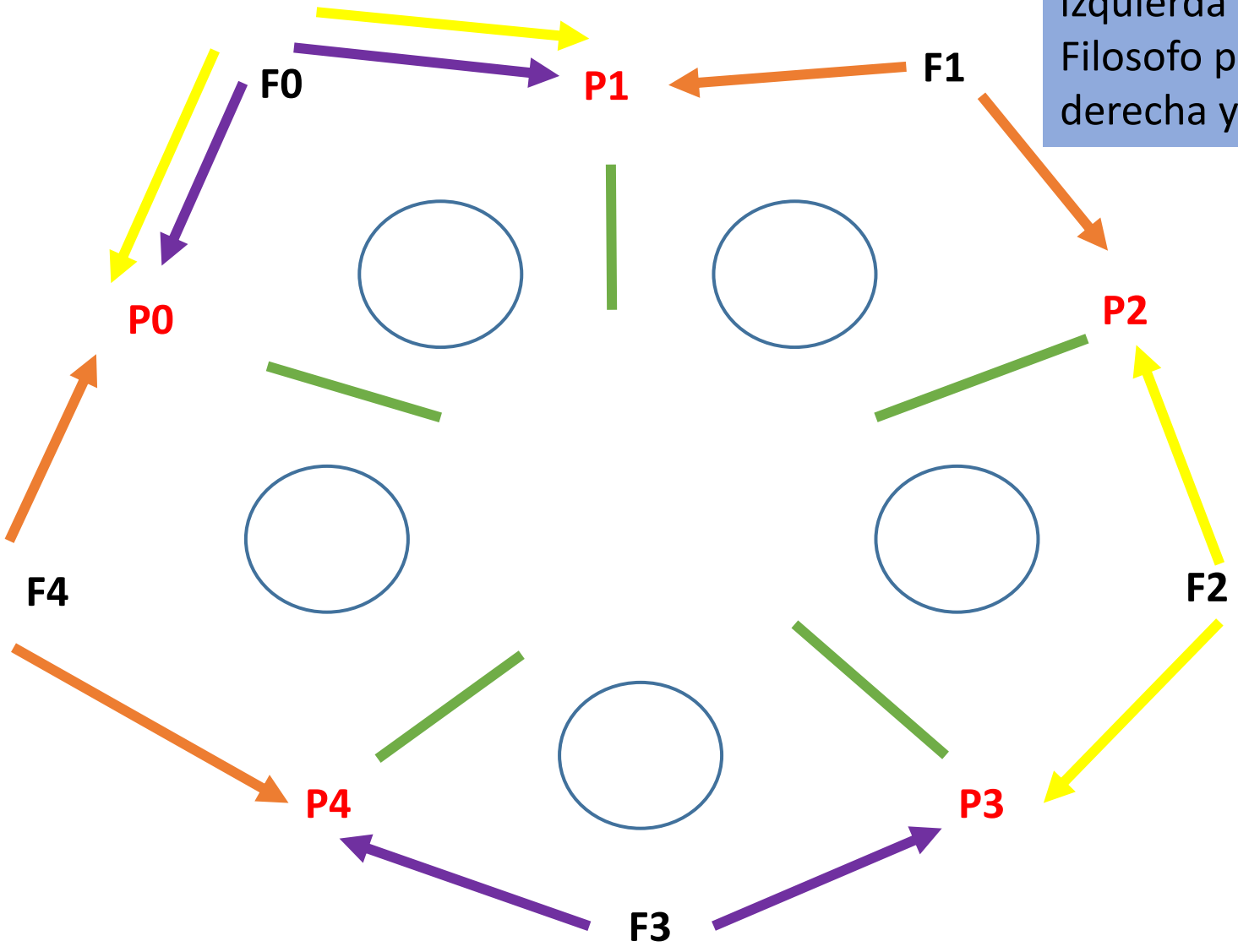
ELABORAR LA EJECUCIÓN MANUAL

CUARTA SOLUCION

- Solución asimétrica, es decir un filosofo impar toma primero su palillo de la izquierda y luego el palillo de la derecha, mientras que un filosofo par elige su palillo de la derecha y luego el de la izquierda .
- Estructuras:
  - P: array[0..4] of semaphore
- Inicialización
  - for k:=0 to 4 do
    - Initial(p[k],1);

p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1

Filosofo impar cogen palillo de la izquierda y luego el de la derecha
Filosofo par coge el palillo de la derecha y luego el de la izquierda

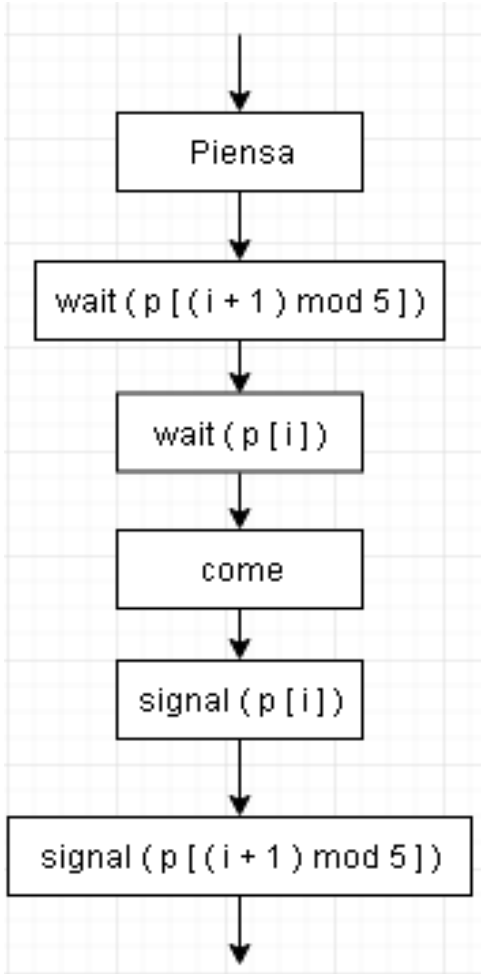


F0 y F1 son considerados en el grupo de "filósofo impar"

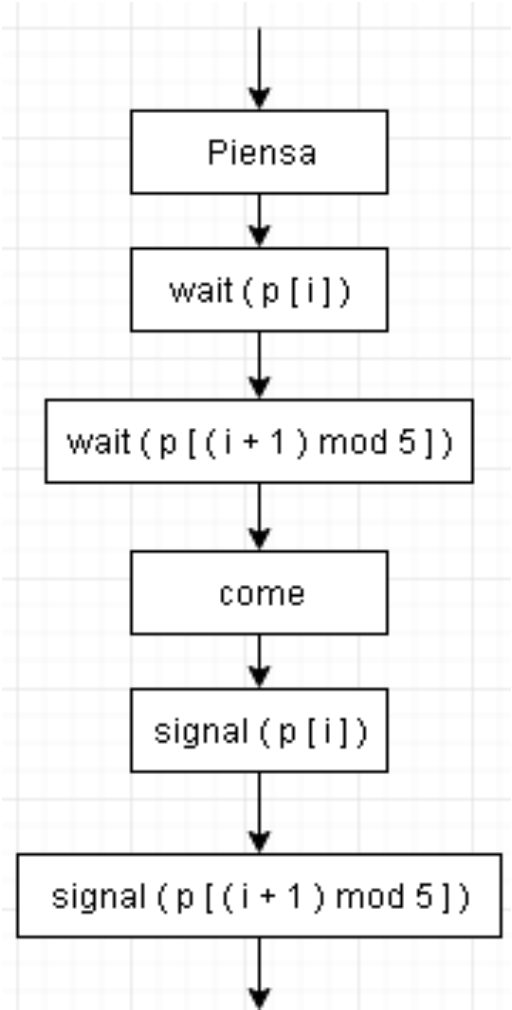
F2 , F3 , F4 son considerados en el grupo de "filósofo par"

DIAGRAMA DE FLUJO

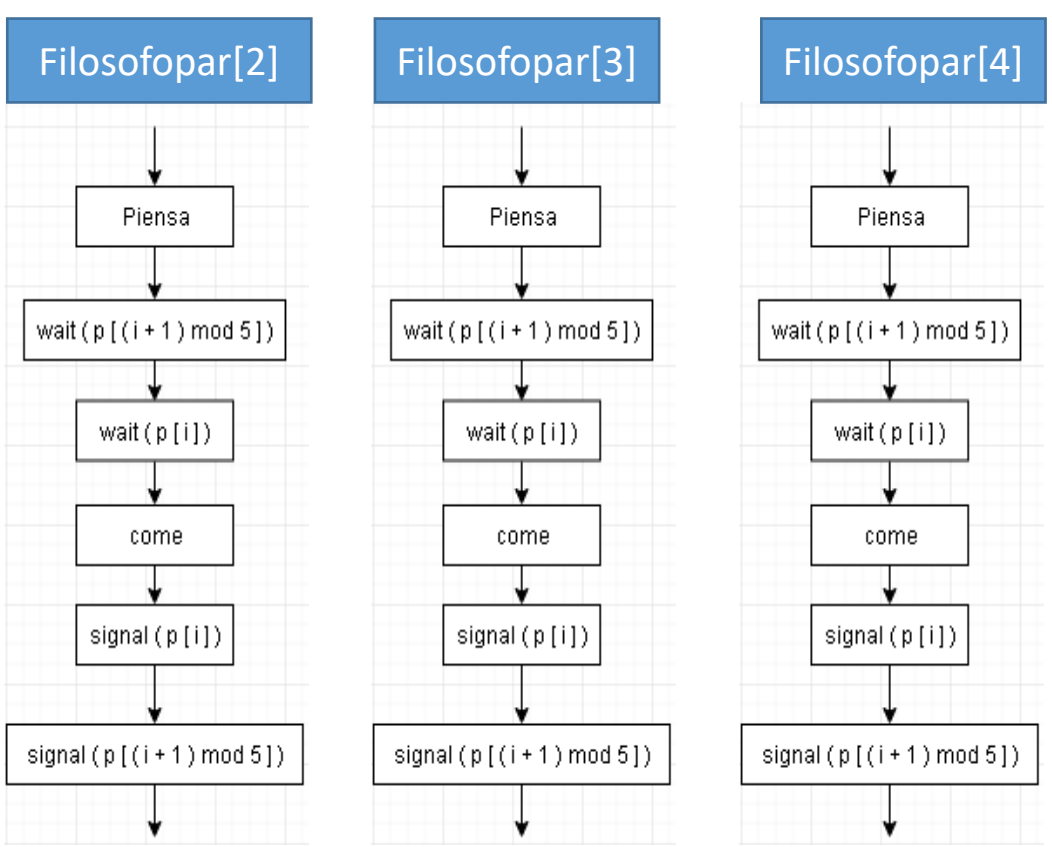
process type filosofopar[i]

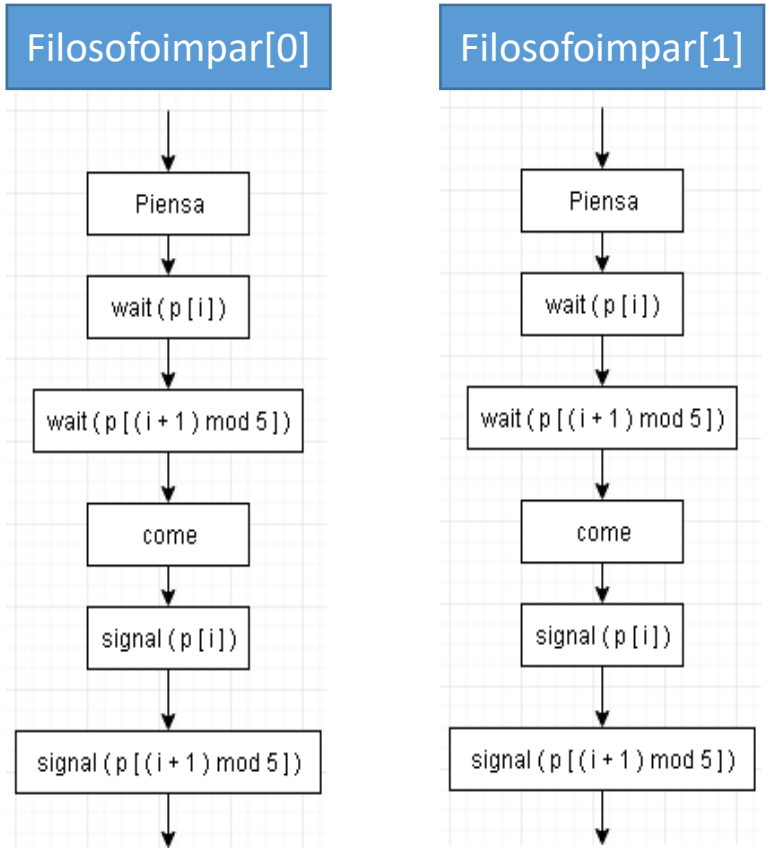


process type filosofoimpar[i]



P[0]	P[1]	P[2]	P[3]	P[4]
0 ó 1	0 ó 1	0 ó 1	0 ó 1	0 ó 1
Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]	Filosofo[0 ó 1 ó 2 ó 3 ó 4]





EJECUCIÓN

p[0]=1
p[1]=1
p[2]=1
p[3]=1
P[4]=1
F[0] y F[2]
F[0] y F[3]
F[1] y F[4]
F[1] y F[3]

T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
i													
P[0]													
P[1]													
P[2]													
P[3]													
P[4]													
F[0]													
F[1]													
F[2]													
F[3]													
F[4]													

ELABORAR LA EJECUCIÓN MANUAL

- https://www.youtube.com/watch?v=trdXKhWAGdg