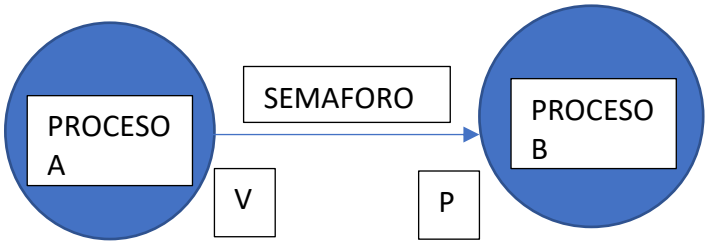
REPASO= Analogía semaforos – Grafos Dirigidos

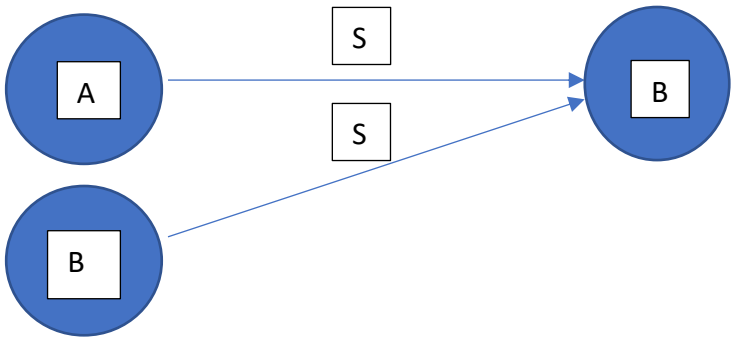
- Arista = Semaforo
- Nodo = Proceso
- Punto de partida arista = Libera Semaforo arista V(Semaforo)
- Destino arista = Pide Semaforo arista P(Semaforo)



PROCESO A
V(SEMAFORO)

PROCESO B
P(SEMAFORO)

Nota = Si a un nodo le llegan varias aristas todas representan el mismo semaforo



PROCESO A
V(S)

PROCESO B
V(S)

PROCESO B
P(S)
P(S)

Solucionar el siguiente problema:

En un evento deportista, existen 3 tipos de competidores, el primer obstáculo es una escalera que tiene la restricción que puede ser subido solamente por un tipo 3 a la vez, o un tipo 2 acompañado o no de un tipo 1, o hasta 3 del tipo 1 simultáneamente. Una vez arriba los corredores compiten por llegar a uno de los 4 botes disponibles. Los de tipo 3 pueden remar solos, los de tipo 1 y 2 necesitan de un remador (solo 2 disponibles). Los remadores la tienen complicada, deben llevar a los corredores y regresarlos mientras ellos descansan y se preparan para el resto del recorrido. Para no ser injusto con los de tipo 3, recién se retornarán los botes tomados por ellos cuando se acumulen 3. Finalizado el recorrido los de tipo 1 que fueron beneficiados se encuentran la sorpresa que deben llevar en sus espaldas a uno de tipo 3 para que también retome energía después de remar. Los de tipo 2 pagaron a los organizadores y hacen este último recorrido libremente.

Resolvamos:

Analicemos el problema

Se pueden identificar tres tramos de la competencia:

- Subir escalera
- Cruzar en bote
- Tramo Final

Trabajemos los tres sub-problemas por separado

Subir escalera

Primero tengo que preguntarme

- Que tengo que hacer?
- Cuales son los procesos?
- Cuales son los recursos que tengo que sincronizar?

Esta claro que se tiene que **subir** (función) de **tres formas distintas** (proceso) una escalera **que es compartida** (recurso).

Por lo tanto tenemos
Proceso= CompetidorTip1, CompetidorTip2, CompetidorTip3.
Semaforo= Escalera -> 3 (Igual a tres porque como mucho puden usarla simultaneamente tres competidores del tipo 1)
Función= subir()

Los competidores del tipo uno se fijan si tienen un lugar en la escalera y suben entonces: Pido lugar en la escalera, la subo y libero el lugar para otro.

CompetidorTip1
P(Escalera) Subir() V(Escalera)

Podemos ver que podrían subir la escalera hasta 3 del tipo 1, si hay más tendrán que esperar.

Ahora sigamos con tipo2, parece igual al tipo 1, pero ojo

- Si sube un tipo 2 solo podria a la vez acompañarlo un tipo 1
    - Esto implicaría reservar dos lugares de la escalera (Recordar que solo tiene tres lugares disponibles)

CompetidorTip1	CompetidorTip2
P(Escalera) Subir() V(Escalera)	P(Escalera) P(Escalera) Subir() V(Escalera)

	V(Escalera)

Finalmente deberíamos sincronizar el tipo tres, recordar que el tipo3 pasa solo (se reserva todos los lugares libres de la escalera)

CompetidorTip1	CompetidorTip2	CompetidorTip3
P(Escalera) Subir() V(Escalera)	P(Escalera) P(Escalera) Subir() V(Escalera) V(Escalera)	P(Escalera) P(Escalera) P(Escalera) Subir() V(Escalera) V(Escalera) V(Escalera)

*Parece funcionar pero…. **NO***
Fijarse que pasa si esta ejecutando el competidor de tipo 3 y antes de ejecutar el tercer P(escalera) se pasa el proceso a la cola de listo y el competidor de tipo 2 entra en ejecución….. Ambos procesos se consumen los semaforos que requieren mutuamente y no hay forma que continue la ejecución. Entonces debemos controlar que una vez que entre en ejecución el proceso del tipo dos no puedan empezar un tipo 3 o viceversa. Esto se soluciona con control mutuo osea un semaforo mutex

CompetidorTip1	CompetidorTip2	CompetidorTip3
P(Escalera) Subir() V(Escalera)	P(M) P(Escalera) P(Escalera) Subir() V(Escalera) V(Escalera) V(M)	P(M) P(Escalera) P(Escalera) P(Escalera) Subir() V(Escalera) V(Escalera) V(Escalera) V(M)
Escalera=3 ->Acumulador M=1 ->Mutex		

Primer subProblema resulto

Cruzar en bote

En el enunciado se puede notar que los competidores luchan por conseguir alguno de los 4 botes disponibles (recursos).

Competidor tipo 1	Competidor tipo 2	Competidor tipo 3
P (bote) RemarYDevolver() V(bote)	P (bote) RemarYDevolver() V(bote)	P(bote) Remar() //Lo deja en el muelle

El enunciado tambien aclara que el tipo 1 y tipo 2 necesitande uno de los dos remadores disponibles (recurso).

Competidor tipo 1	Competidor tipo 2	Competidor tipo 3
P (remador) P (bote) RemarYDevolver() V(bote) V(remador)	P (remador) P (bote) RemarYDevolver() V(bote) V(remador)	P(bote) Remar() //Lo deja en el muelle

Notar el orden, se pide primero el remador, para no impedir que otro no use el bote si no conseguis un remador.

Por último el enunciado dice que el competidor del tipo 3 se lleva un bote y lo deja en un muelle con lugar para tres (recurso), y cuando este se llena un encargado retorna los botes. Podemos identificar otro proceso, el encargado del muelle.

Competidor tipo 1	Competidor tipo 2	Competidor tipo 3	Encargado muelle

P (remador)
P (bote)
Remar()
V(bote)
V(remador)
	P (remador)
	P (bote)
	Remar()
	V(bote)
	V(remador)
		P(LugarMuelle)
		P(bote)
		Remar()
		//Lo deja en el muelle
		V(BoteEnMuelle)
			P(BoteEnMuelle)
			P(BoteEnMuelle)
			P(BoteEnMuelle)
			devolverBotes()
			V(Bote)
			V(Bote)
			V(Bote)
			V(LugarMuelle)
			V(LugarMuelle)
			V(LugarMuelle)

Bote = 4 Remador = 2 BoteEnMuelle = 0 LugarMuelle = 3			

Ultima etapa

En esta parte del problema tenemos que controlar que un competidor del tipo1 pase acompañado de un tipo3, osea tenemos dos procesos CompetidorTipo1 y CompetidorTipo3, el tipo1 espera de un tipo3 y el tipo3 una vez en upa del tipo1 le dice empezemos a correr.

CompetidorTipo1	CompetidorTipo3
V(Esperando) P(Listo) Correr()	V(Listo) P(Esperando) Cavalgar()
Esperando=0 ->Acum Listo=0 ->Acum	

Resumen

Competidor tipo 1	Competidor tipo 2	Competidor tipo 3	Encargado muelle

P(Escalera)
Subir()
V(Escalera)
P (remador)
P (bote)
Remar()
V(bote)
V(remador)
V(Esperando)
P(Listo)
Correr()
	P(M)
	P(Escalera)
	P(Escalera)
	Subir()
	V(Escalera)
	V(Escalera)
	V(M)
	P (remador)
	P (bote)
	Remar()
	V(bote)
	V(remador)
	Correr()
		P(M)
		P(Escalera)
		P(Escalera)
		P(Escalera)
		Subir()
		V(Escalera)
		V(Escalera)
		V(Escalera)
		V(M)
		P(LugarMuelle)
		P(bote)
		Remar()
		//Lo deja en el muelle
		V(BoteEnMuelle)
		V(Listo)
		P(Esperando)
		Cavalgar()
			P(BoteEnMuelle)
			P(BoteEnMuelle)
			P(BoteEnMuelle)
			devolverBotes()
			V(Bote)
			V(Bote)
			V(Bote)
			V(LugarMuelle)
			V(LugarMuelle)
			V(LugarMuelle)

Bote = 4 Remador = 2 BoteEnMuelle = 0 LugarMuelle = 3 Escalera=3 M=1 Esperando=0 Listo=0			