# Problema de los filósofos

Cota Guillermo
Barriga Rosales Alan
Macias Gomez Jorge

January 5, 2021

## 1 Relato breve del problema

Cinco filósofos se encuentran sentados alrededor de una mesa. Cada filósofo tiene un plato de comida y un tenedor a la izquierda de éste. Para comer necesitan ocupar dos tenedores. Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará esperando con el tenedor en la mano, para luego empezar a comer.

El problema consiste en encontrar un algoritmo que permita que los filósofos nunca se mueran de hambre, el cual lo veremos para un número $n > 1, n \in N$ .

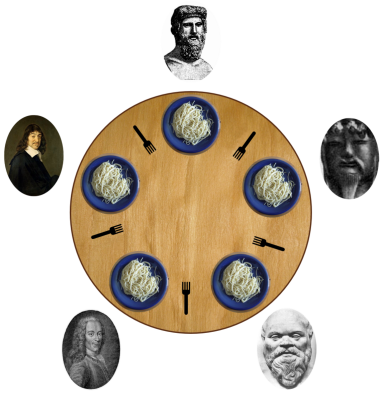

**Figura 1: Representación del problema de los 5 filósofos.**

## 2 Relato de la solución aproximada

Para nuestra solución tomaremos un filosofo, y numeraremos a los filósofos de forma consecutiva, todos los filósofos pares tomaran su tenedor derecho primero, y luego el izquierdo, y los impares harán lo contrario. Esto nos garantiza que en el caso de que todos los filósofos quieran comer, lo puedan hacer en a lo más 3 turnos. Esto debido a que si solo un subconjunto de filósofos con números pares asignados quieren comer, lo pueden hacer, al igual que si solo quieren comer los impares. En caso de que quieran comer todos al mismo tiempo, y son un número par de comensales, todos los pare podrán comer primero y enseguida los impares. En caso de que sean un número impar, primero comerán los pares, luego los impares excepto el primero o el último y por último el que no comió en el segundo turno.

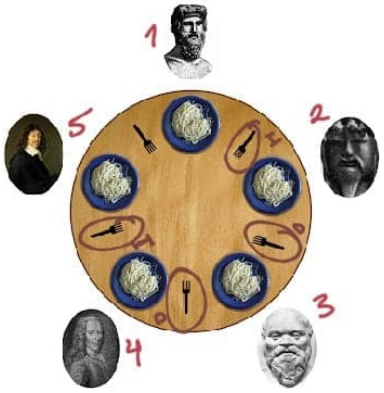


**Figura 2: Primer paso en la solución propuesta para el problema de 5 filósofos donde primero comen los pares, tal que I representa izquierda y D derecha.**

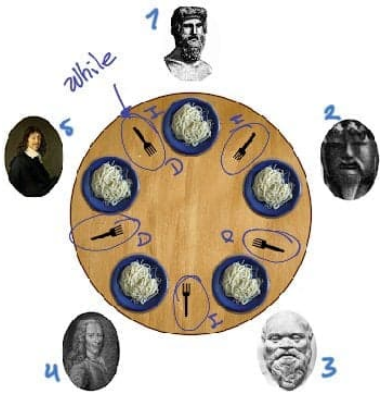


**Figura 3: Segundo paso para la solución propuesta donde el primer filósofo se espera a que el n-ésimo filósofo coma, mediado por un ciclo while donde I representa izquierda y D derecha.**

# 3 Pseudocódigo

El pseudocódigo de la solución descrita en el párrafo anterior es como el que sigue para el caso de una `mesa` con un número `n` de `filosofo`:

```
def filosofo_comer(i):
    pname = currentProcessName()
    if (pname%2):
        try:
            mesa[i].agarrarTenedor(izquierda)
            while (!mesa[i].tenedorLibre(derecha)):
                wait()
            mesa[i].agarrarTenedor(derecha)
            mesa[i].comer()
            mesa[i].dejarTenedor(derecha)
            mesa[i].dejarTenedor(izquierda)
    else:
        try:
            mesa[i].agarrarTenedor(derecha)
```

```
while (!mesa[i].tenedorLibre(izquierda)):
    wait()
mesa[i].agarrarTenedor(izquierda)
mesa[i].comer()
mesa[i].dejarTenedor(derecha)
mesa[i].dejarTenedor(izquierda)
```

# 4 Aplicación para solucionar la exclusión mutua

Los whiles esperan hasta la disponibilidad del recurso, ya que solo una persona puede tomar el tenedor al mismo tiempo. Tenemos que usar un candado para poder solucionar que solo un proceso se apropie del recurso.