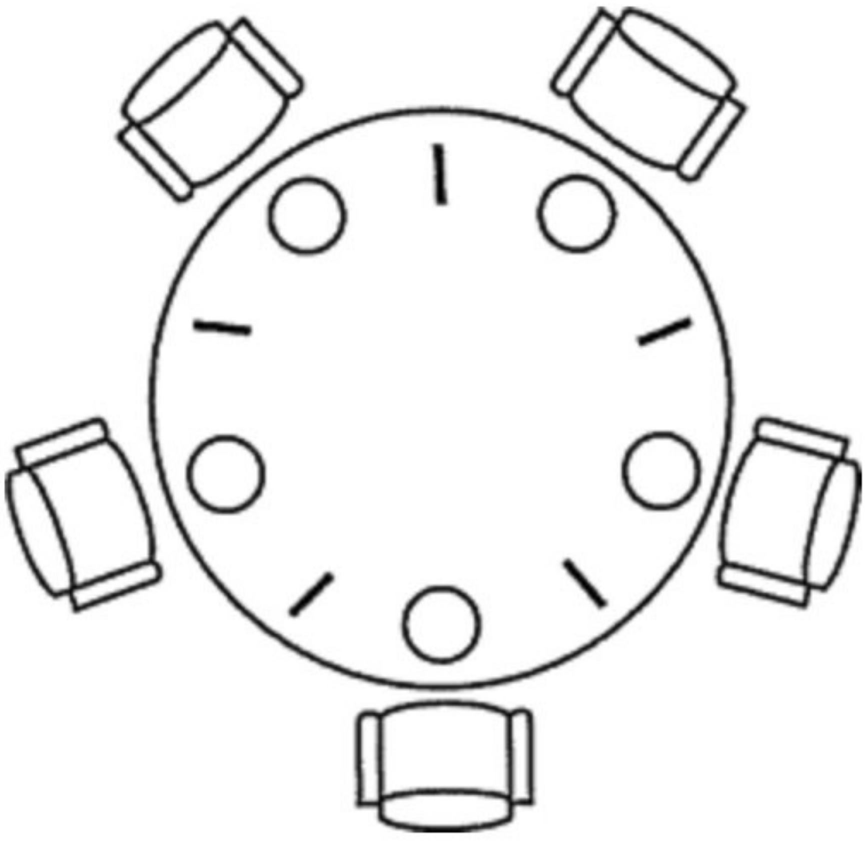
The Dining-Philosophers Problem

田野 2017329621125

1. Problem description

Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the centr of the table is a bowl of rice, and the table is laid with five single chopsticks. When a philosopher thinks, she does not interache with her colleagues. From time to time, a philosopher gets hungry and tries to pick uo the two chopsticks that are closest to her (the chopsticks that are between her and her left and right neighbors). A philosopher may pick up only one chopstick at a time. Obviously, she cannot pick up a chopstick that is already in the hand of a neighbor. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she is finished eating, she puts down both of her chopsticks and starts thinking again.



2. Some solution to the problem

solution1

Description

Suppose a philosopher is allowed to grab chopsticks when both sides of the chopsticks are available. And at the same time only one philosopher could eat it.

Code

```
def solution1(lock,phd):
    while True:
        phd.think()
        lock.acquire()
        flag = phd.left.acquire()
        if flag:
            print('phd%s takes left chopstick' % phd.name)
            flag = phd.right.acquire()
            if flag:
                print('phd%s takes right chopstick' % phd.name)
                phd.eat()
                phd.right.release()
            phd.left.release()
        lock.release()

def run1():
    lock1 = Lock()
    lock2 = Lock()
    lock3 = Lock()
    lock4 = Lock()
    lock5 = Lock()
    phd1 = phd(1, lock5, lock1)
    phd2 = phd(2, lock2, lock3)
    phd3 = phd(3, lock2, lock4)
    phd4 = phd(4, lock3, lock5)
    phd5 = phd(5, lock4, lock1)
    lock = Lock()
    p1 = Process(target = solution1, args=(lock,phd1,))
    p2 = Process(target = solution1, args=(lock,phd2,))
    p3 = Process(target = solution1, args=(lock,phd3,))
    p4 = Process(target = solution1, args=(lock,phd4,))
    p5 = Process(target = solution1, args=(lock,phd5,))
    p1.start()
    p2.start()
    p3.start()
    p4.start()
    p5.start()
```

```
run1()
```

Output :

```
1 is thinking
2 is thinking
3 is thinking
4 is thinking
5 is thinking
phd3 takes left chopstick
phd3 takes right chopstick
3 is eating
3 is thinking
phd4 takes left chopstick
phd4 takes right chopstick
4 is eating
4 is thinking
phd5 takes left chopstick
phd5 takes right chopstick
5 is eating
phd1 takes left chopstick
5 is thinking
phd1 takes right chopstick
1 is eating
1 is thinking
phd2 takes left chopstick
phd2 takes right chopstick
2 is eating
2 is thinking
phd3 takes left chopstick
phd3 takes right chopstick
3 is eating
phd4 takes left chopstick
3 is thinking
phd4 takes right chopstick
4 is eating
4 is thinking
phd5 takes left chopstick
phd5 takes right chopstick
5 is eating
phd1 takes left chopstick
5 is thinking
....
```

Solution2

Description:

At most four philosophers are allowed to pick up the chopsticks on the left side at the same time, which ultimately guarantees that at least one philosopher will be able to eat and release the two chopsticks he used when he finished, thus enabling more philosophers to eat.

Code:

```
from multiprocessing import Semaphore
import time
import random
from multiprocessing import Lock, Process
class phd:
    def __init__(self, name, left, right):
        self.name = name
        self.left = left
        self.right = right


    def eat(self):
        print("%s is eating" % self.name)
        time.sleep(random.randint(1,3))

    def think(self):
        print("%s is thinking" % self.name)
        time.sleep(random.randint(2,4))

se = Semaphore(4)

def solution2(se,phd):
    while True:
        phd.think()
        se.acquire()
        phd.left.acquire()
        print('phd%s takes left chopstick' % phd.name)
        flag = phd.right.acquire()
        if flag:
            print('phd%s takes right chopstick' % phd.name)
            phd.eat()
            phd.right.release()
            phd.left.release()
        se.release()

def run2():
    lock1 = Lock()
    lock2 = Lock()
    lock3 = Lock()
    lock4 = Lock()
    lock5 = Lock()
    phd1 = phd(1, lock5, lock1)
    phd2 = phd(2, lock2, lock3)
    phd3 = phd(3, lock2, lock4)
    phd4 = phd(4, lock3, lock5)
    phd5 = phd(5, lock4, lock1)
    lock = Lock()
    p1 = Process(target = solution2, args=(lock,phd1,))
    p2 = Process(target = solution2, args=(lock,phd2,))
    p3 = Process(target = solution2, args=(lock,phd3,))
    p4 = Process(target = solution2, args=(lock,phd4,))
    p5 = Process(target = solution2, args=(lock,phd5,))
    p1.start()
    p2.start()
    p3.start()
    p4.start()
    p5.start()

run2()
```

Output:

```
1 is thinking
2 is thinking
3 is thinking
4 is thinking
5 is thinking
phd3 takes left chopstick
phd3 takes right chopstick
3 is eating
3 is thinking
phd1 takes left chopstick
phd1 takes right chopstick
1 is eating
1 is thinking
phd2 takes left chopstick
phd2 takes right chopstick
2 is eating
2 is thinking
phd4 takes left chopstick
phd4 takes right chopstick
4 is eating
4 is thinking
phd5 takes left chopstick
phd5 takes right chopstick
5 is eating
```

```
phd3 takes left chopstick
5 is thinking
phd3 takes right chopstick
3 is eating
3 is thinking
phd1 takes left chopstick
phd1 takes right chopstick
1 is eating
1 is thinking
phd2 takes left chopstick
phd2 takes right chopstick
2 is eating
2 is thinking
```