

## Filosofi a cena

```
while (true) {
    pensa();
    prendi_forchetta();
    prendi_forchetta();
    mangia();
    lascia_forchette();
```

Ci sono le condizioni per un deadlock.

## Soluzione di Dijkstra

Un mutex e  $n$  semafori

```
while (true) {
    pensa();

    lock()
    stato[i] = affamato;
    test(i);
    unlock()

    P(sem[i]);
    mangia();
    lascia_forchette(i);
}
```

Dove:

```
test(i) {
    if (stato[i] == affamato
        && stato[dx(i)] == pensa
        && stato[sx(i)] == pensa) {
        stato[i] = mangia;
        V(sem[i]);
    }
}

lascia_forchette(i) {
    lock();
    stato[i] = pensa;
    test(sx(i));
    test(dx(i));
    unlock();
}
```

Problema: un solo lock per tutti, non è efficiente per  $n$  grande.

## Ordinamento di risorse

Si numerano le forchette e si acquisiscono sempre in ordine. Problemi:

- starvation: un filosofo può riprendere la forchetta prima che ci riesca il vicino;
- inefficiente con acquisizioni condizionate: se ho 1 e 3 e scopro di aver bisogno di 2, devo rilasciare 3 e acquisire 2 e 3.

## Algoritmo del banchiere

Il banchiere concede la forchetta al filosofo solo se:

- ci sono almeno due forchette libere (stato safe: almeno una forchetta libera), o
- il filosofo ha già una forchetta.