Deadlock Avoidance

safe state - no deadlock
unsafe - deadlock could occur
Single instance of each resource
Banker's Algorithm

Data:

- `int Available[]` - for each res.
- `int max[i][j]` - max res. of type `j` used by proc. `i`
- `int allocation[i][j]` - # of res. `j` alloc to proc `i`
- `int need[i][j] = max - allocation`
Safety Algorithm

boolean finish[] - for each proc
work[] = available

- find process i s.t.
  finish[i] = false
  and need[i][j] <= work[j]
    for all j
  if no such i, safe if all are finished, otherwise unsafe
else
  finish[i] = true
  work[j] += allocation[i][j]
    for all j
when proc i makes a request request[j] = # of res.

1. request[j] ≤ need[i][j] for all i
   (error if not)

2. request[j] ≤ avail[j] ∀j
   if not, wait

3. Simulate granting the request
   avail[j] -= request[j] ∀j
   alloc[i][j] += request[j]
   need[i][j] -= request[j]

   check if safe
   uses - allocate
no-deny, fix arrays
Detection
- allow deadlock
- recognize it
- fix it

Check for deadlock (cycles in resource graph)

have locks expire
Live lock
Recovery

1. Kill all involved: $P_1, P_2, P_3$

2. Kill one at a time

3. $\varnothing$