CSE 330: Operating Systems

Class: 21 **Date:** 3/29

Spring 2018

Note Title

Deadlocks

- Acquire → Request → wait → get
- Use → hold
- Release → done

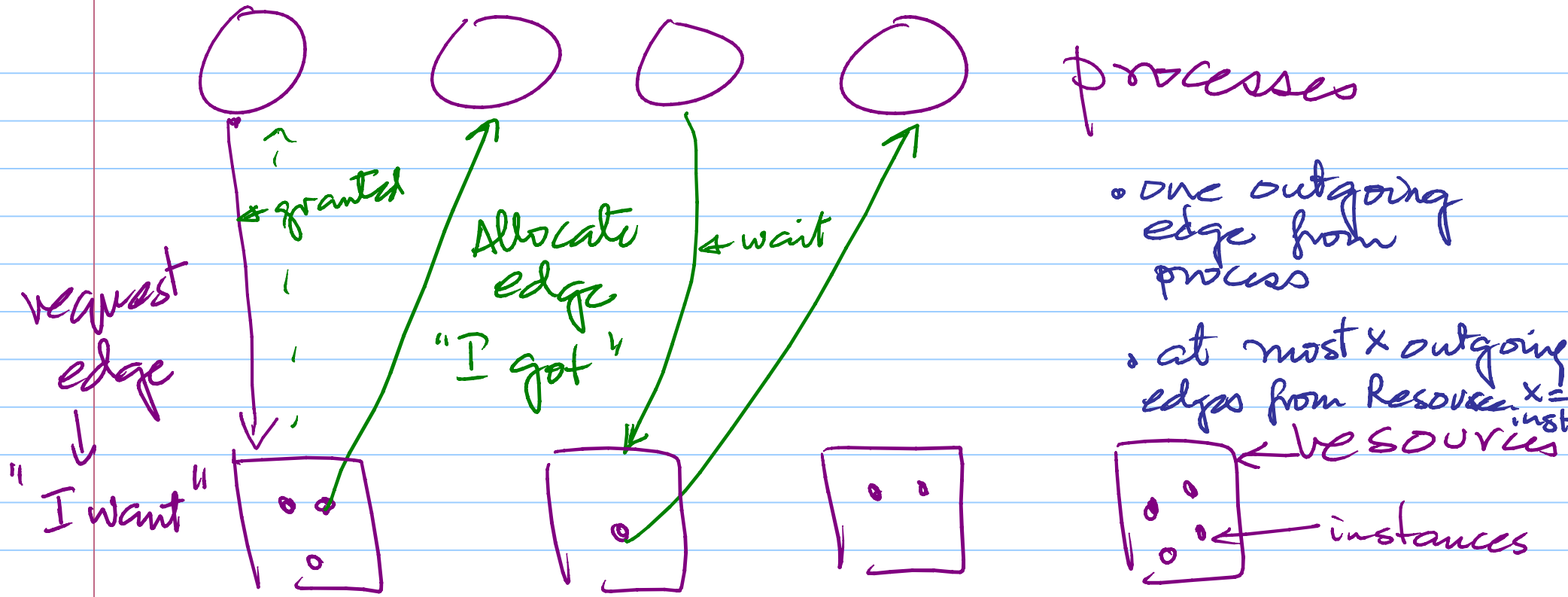
Processes & resources

Necessary Conditions

- mutual exclusion
- no preemption
- hold & wait
- circular wait

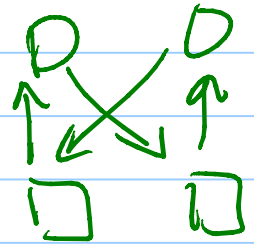
Resource Allocation graphs (RAG)

- 2 kinds of nodes – process
  – resource
- Resources can have multiple instances
  (Resource type, Resource instance)
- Edges can be 2 types – request edge
  – Allocated Edge


granted
Allocate edge
"I got"
wait
request edge
"I want"
processes
• one outgoing edge from process
• at most x outgoing edges from Resource. x = # instances
resources
instances

Cycle in a RAG ?

- deadlock → NO, maybe
- no cycle, no deadlock
- yes cycle, maybe deadlock



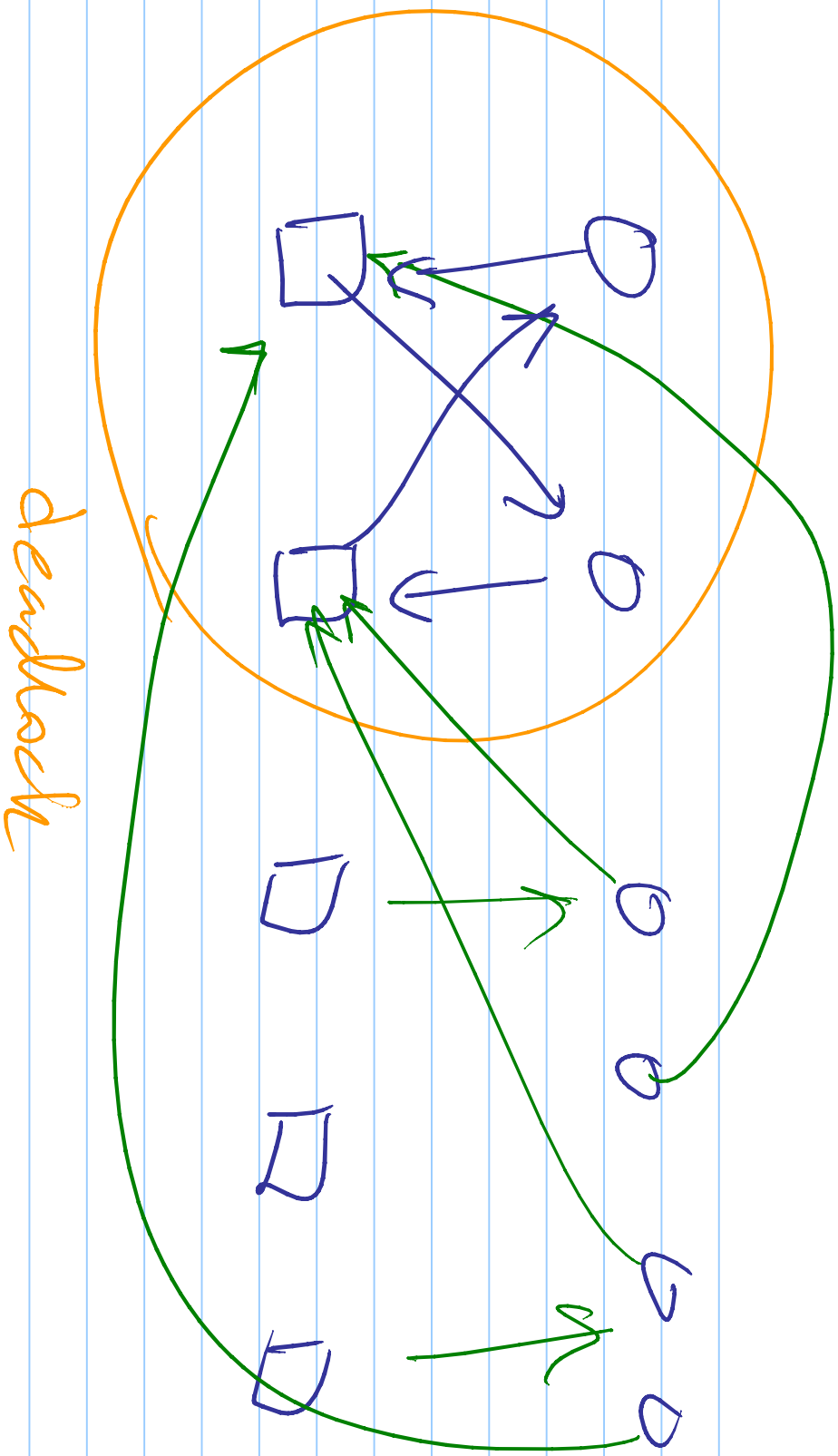
deadlock

If all resources have one instance

- cycle is a sufficient condition for deadlock

What to do about deadlocks?

1. prevention → deadlocks cannot happen
2. avoidance →
3. detection & recovery


deadlock

Deadlock prevention

- ensure at least 1 of the necessary conditions do not hold
  - mutex
  - no preempt
  - hold & wait
  - circular wait

Deadlock prevention

① [No] mutual exclusion

→ all resources are shareable

→ " " " read only

→ simultaneous usage possible

② [No] no preemption

→ all resources are preemptible

↑

OS follows this rule for

CPU, memory

- [no] hold & wait

→ good idea in programs

→ release a CS, before getting another one

e.g → do not do P(S1) P(S2)

(no nested critical sections)

hold ↑ wait ↑

Database programs

- airline reservations

Book Phx to NYC via Chicago

① find flight Phx → NYC → lock

② find next flight → lock

book both

→ unlock both

[No] Circular wait

→ mutex OK, no preempt OK,
hold & wait OK → But NO circular

→ ordered resource allocation

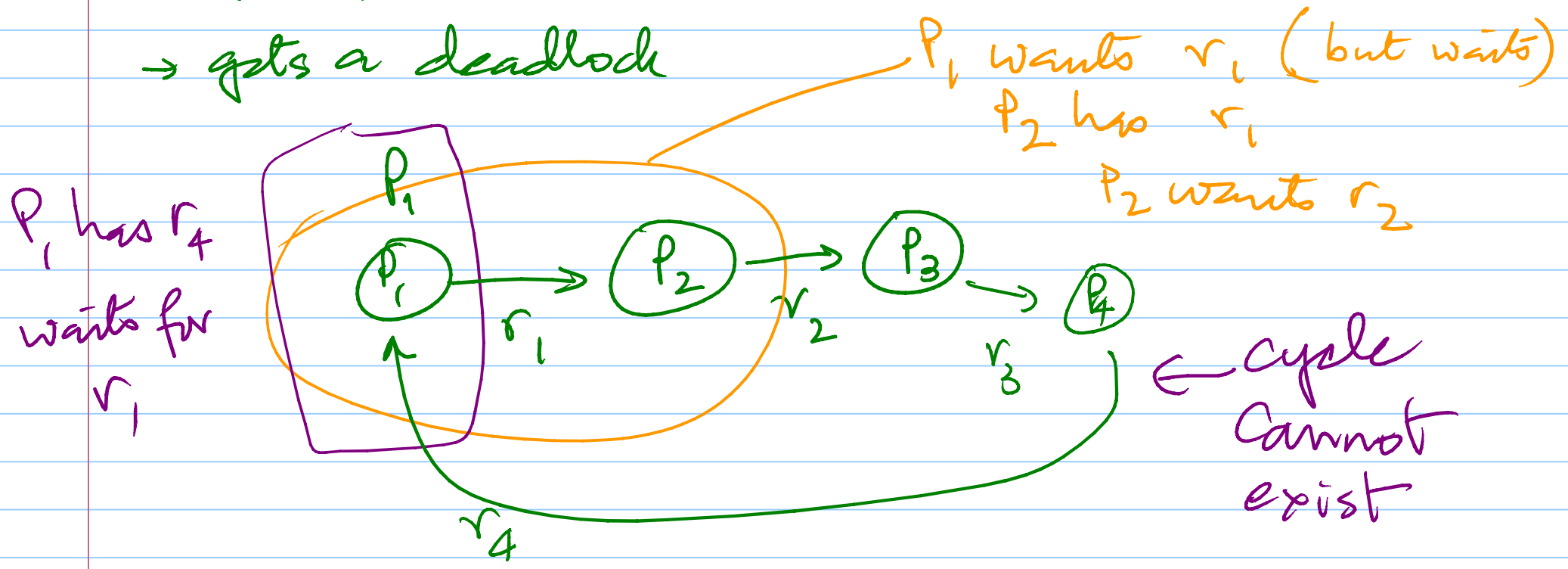
all resources are numbered

0 - - - - - - - (n-1) n resources

A process can request resource i if all the resources it has already holds have numbers < i

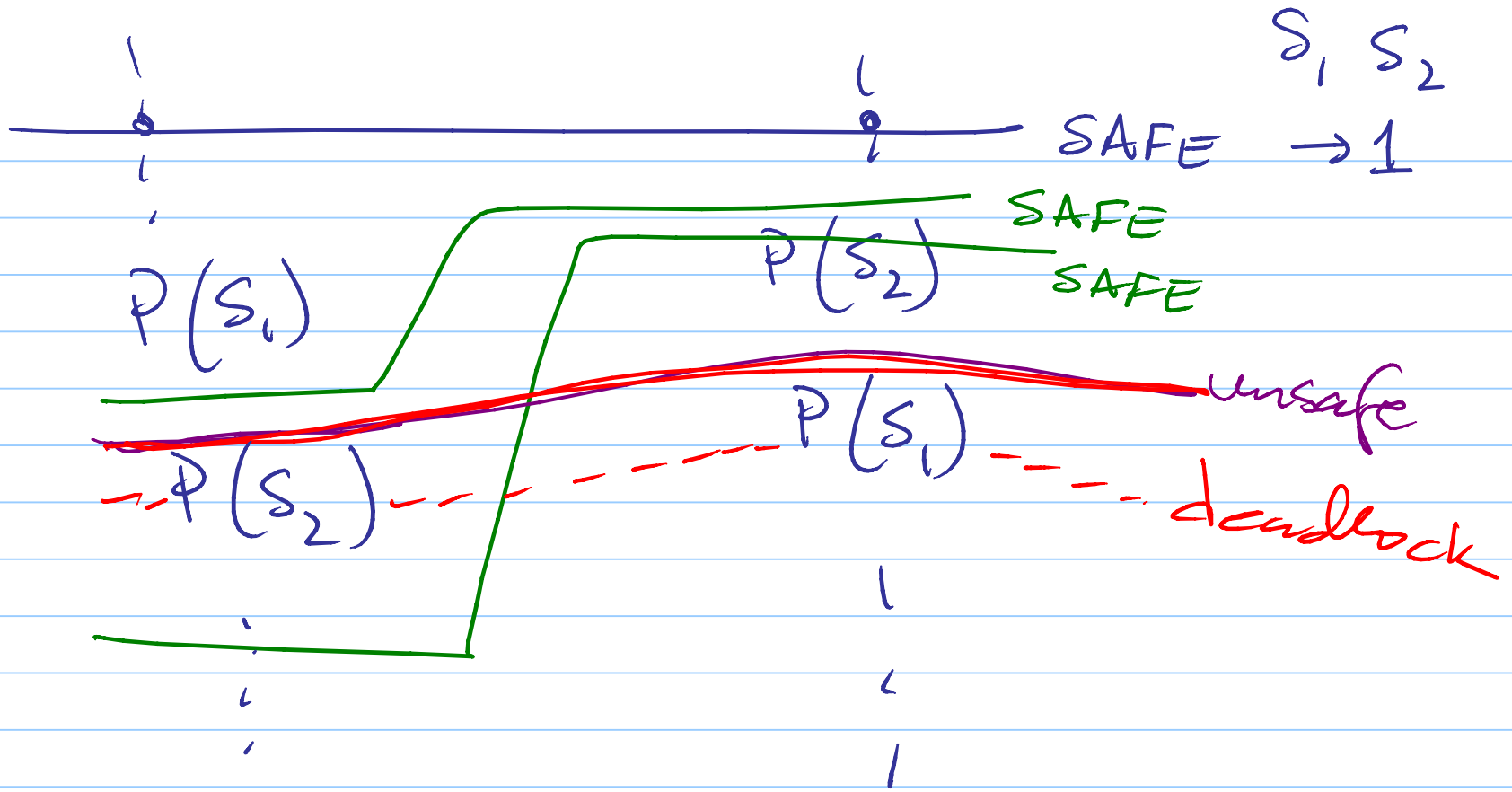
- Suppose ordered resource allocation does not work

→ gets a deadlock



Prevention

- 4 rules, choose 1
  - ↳ not always practical


S1 S2
SAFE → 1
SAFE
SAFE
P(S1)
P(S2)
P(S1)
P(S2)
unsafe
deadlock

safe
safe
safe
unsafe
unsafe
deadlock
all possible
states

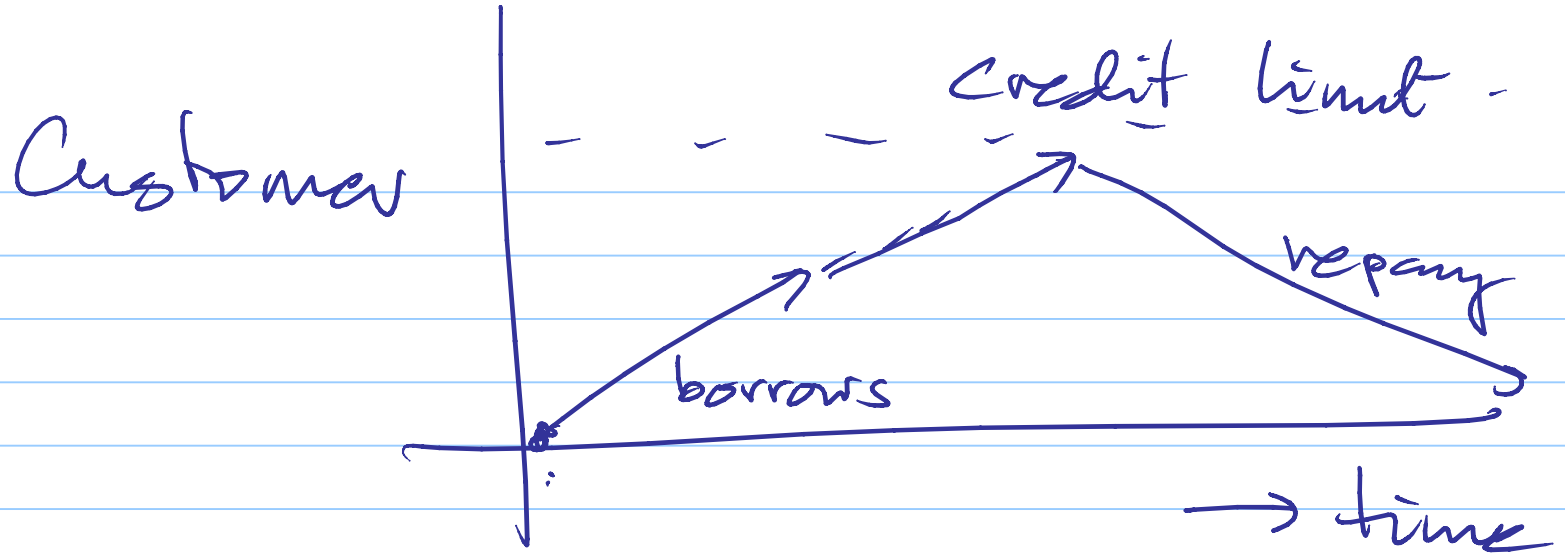
Bankers Algorithm

[simplified version]

Bank has $N on deposit

" " customers $C_1, C_2 \ldots$

each customer wants to borrow money


Customer
credit limit
repay
borrows
time

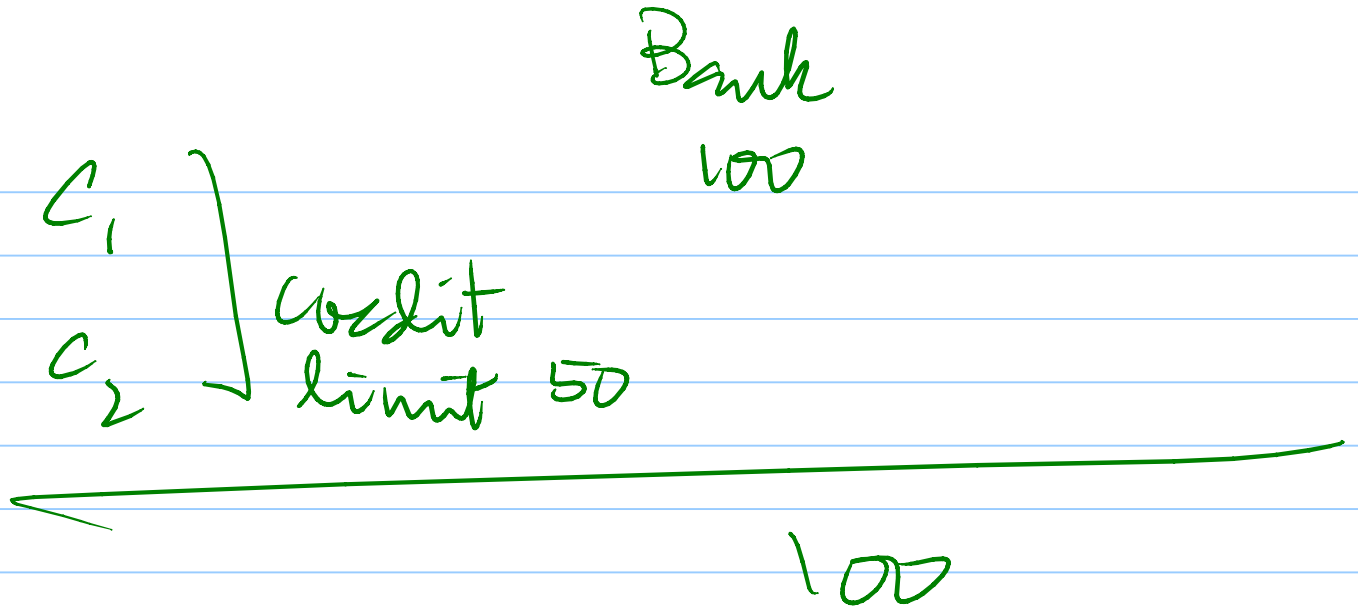
Currency → type of resource

$x → x instances of $

borrow → acquire

repay → release

Credit limit → predeclared resource acquisition plan



70 → each borrow $50'
70