CSE 330: Operating Systems

Class: 23 **Date:** 4/5

Spring 2018

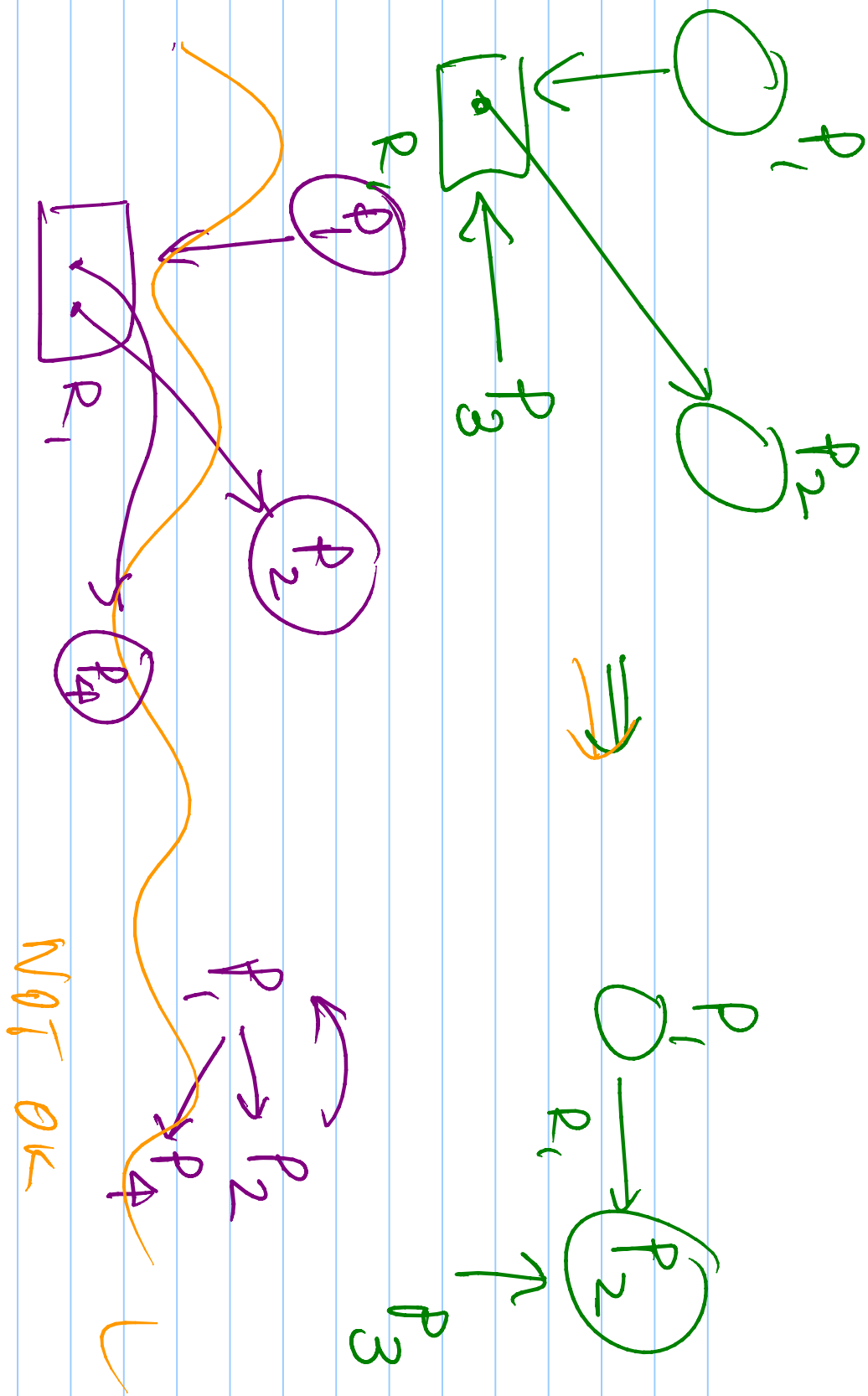
Note Title

Deadlocks

Detection

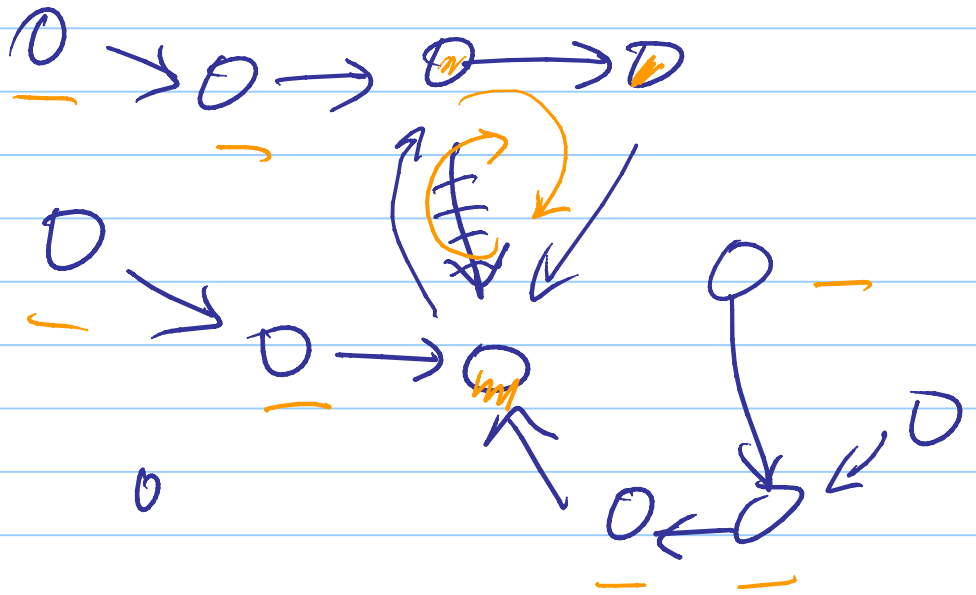
Convert RAG to WFG

↳ one instance of each type of resource


P1
P2
P3
P4
NOT OK

1 instance of each resource

→ maintain a WFG



Cycles in a WFG

- necessary → has to have a cycle to have deadlock (no cycle, no deadlock)
- sufficient → if there is a cycle there is a deadlock

deadlock detection in WFG

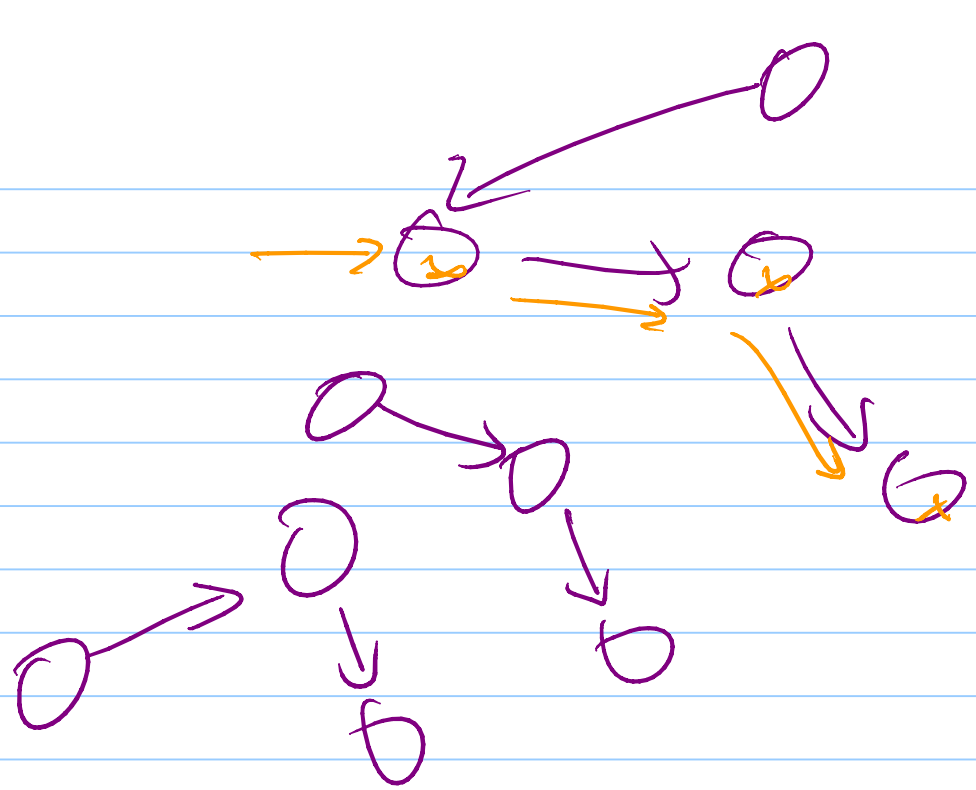
→ check for cycles

↳ YES → deadlock

↳ NO → no deadlock

how?

complexity?



- depth first search

→ all nodes marked → no cycle

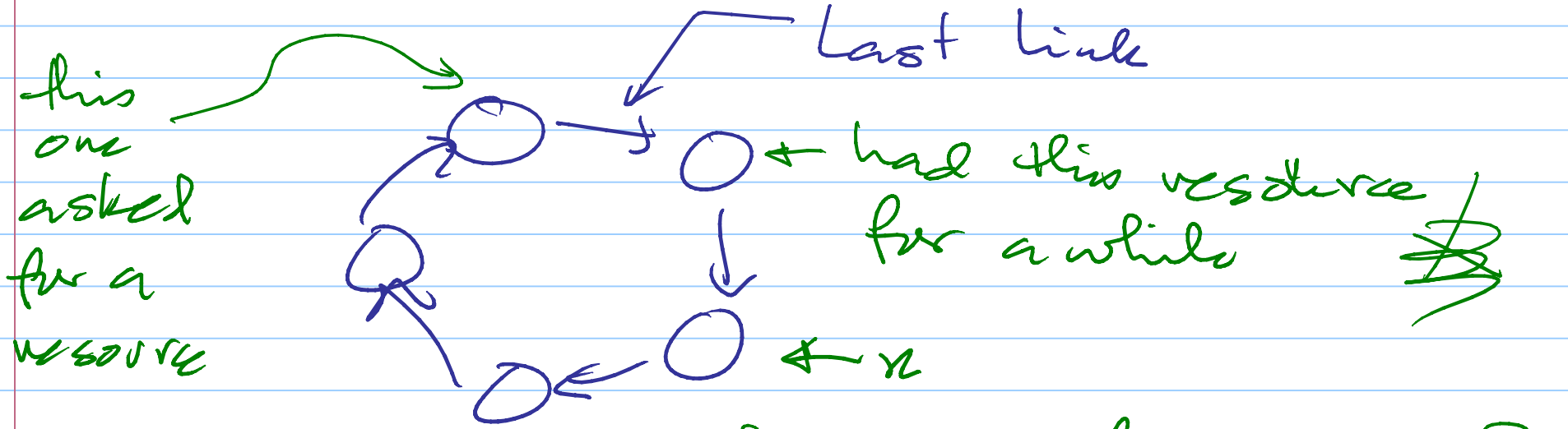
find traversal goes to beginning / cycle

Complexity → # of nodes → n per detection

when to do cycle detection?

- eager → every time a new edge is added
  - ↳ optimization
    - → follow only 1 path.
- lazy → "sometimes"
  - → full cycle detection

Suppose a cycle is detected



deadlock recovery

→ get rid of the cycle

→ kill one process, reclaim all its resources & allocate them

→ which one

one → least priority

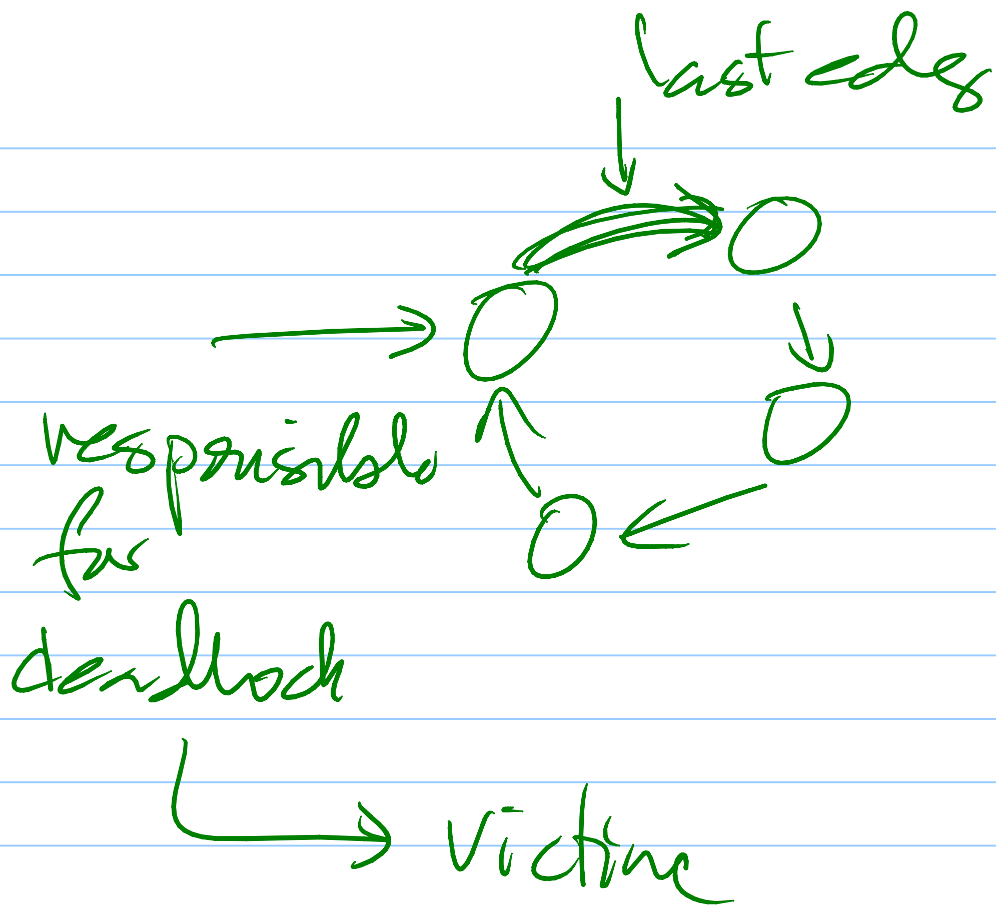
→ oldest

→ youngest

→ with most resources

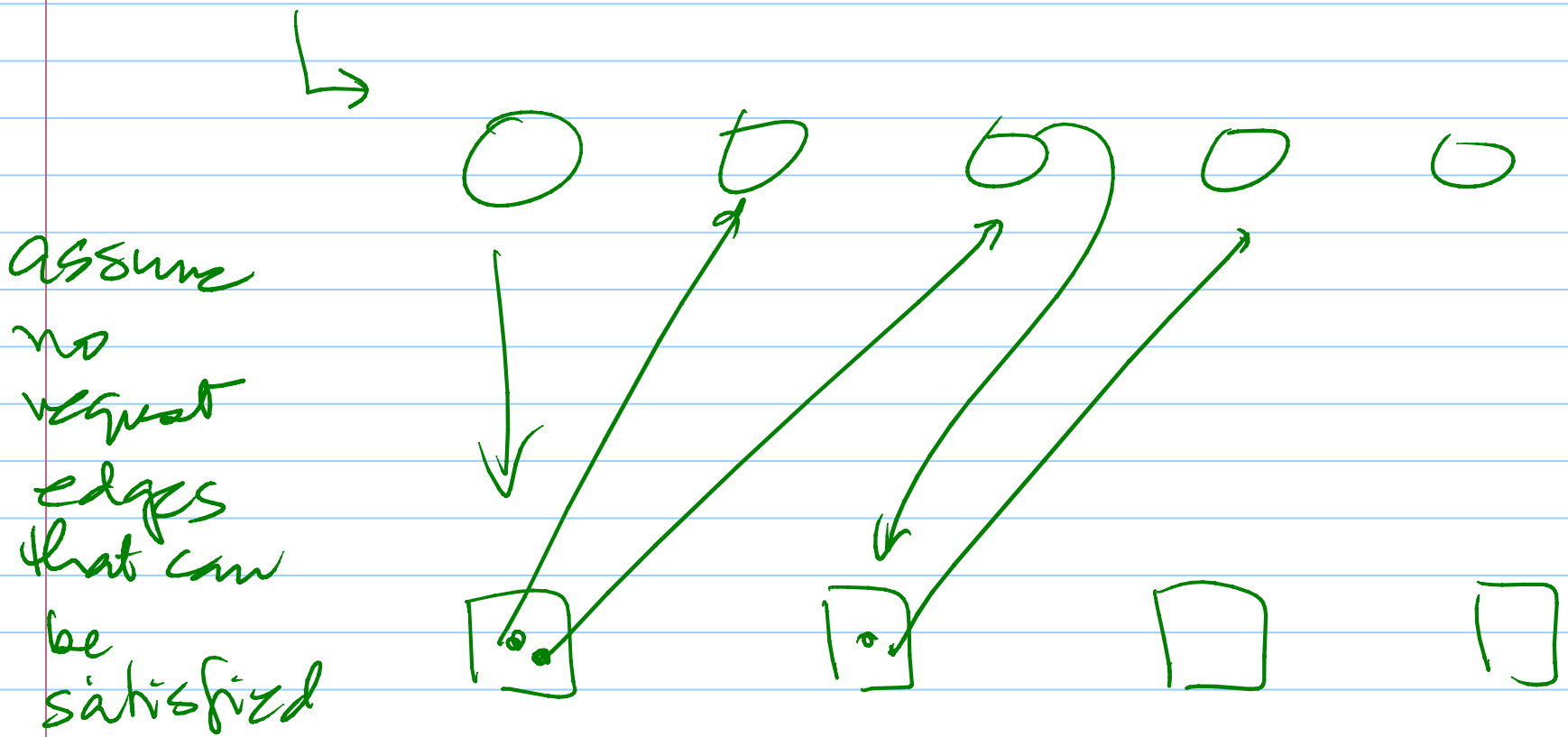
→ with least resources

→ all have pros & cons & starvation...


last edge
responsible for deadlock
victim

Deadlock detection in RAG

Copy



- make copy of RAG
- delete processes with no outgoing edges
- delete incoming edges of
  ↳ release resources
- allocate these released resources if there are request edges

Result

→ all processes deleted

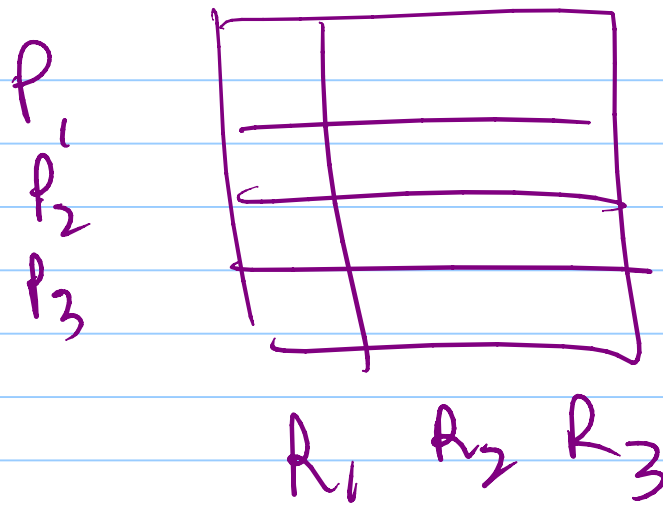
↳ NO deadlock

→ all / some processes still have outgoing edges

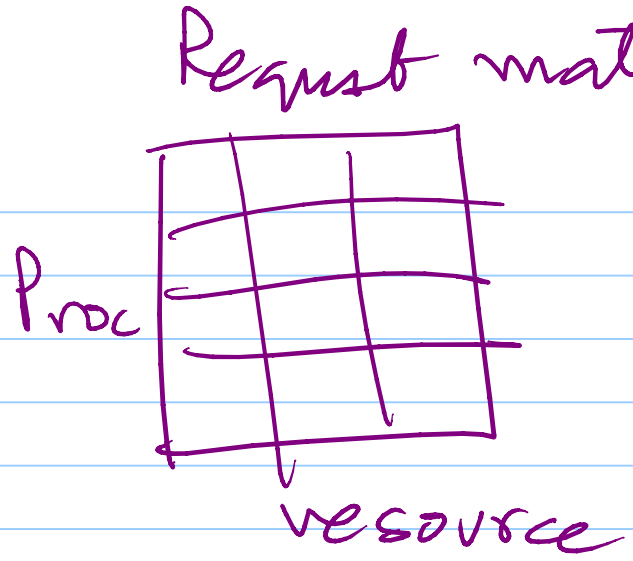
↳ Some of these are involved in the deadlock

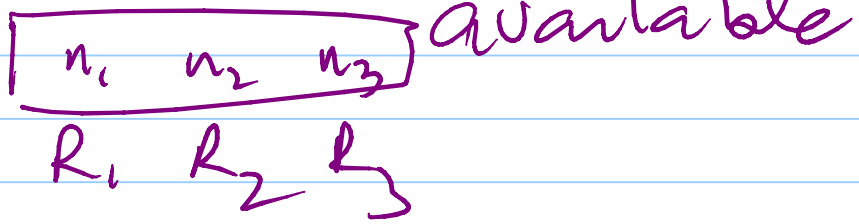
cycle detect ←

Allocation Matrix



Request matrix





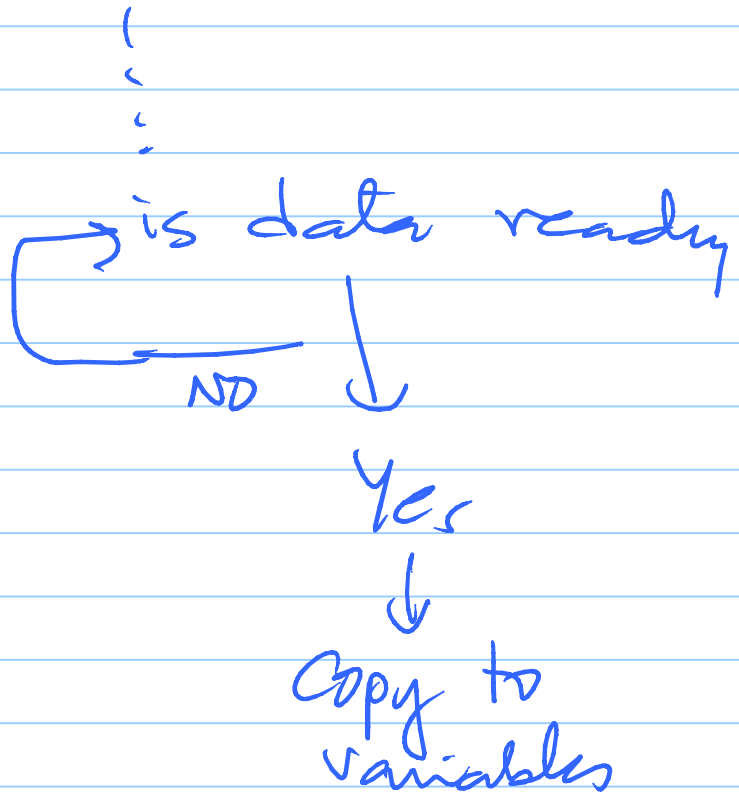
Input Output / Storage mngmnt

read (data)

write (data)

Polled I/O

read (data) ⟹



Magnetic disks