LLD - Problem Solving

1. Debt simplification
   - ~~optimal~~
   - ~~practical~~

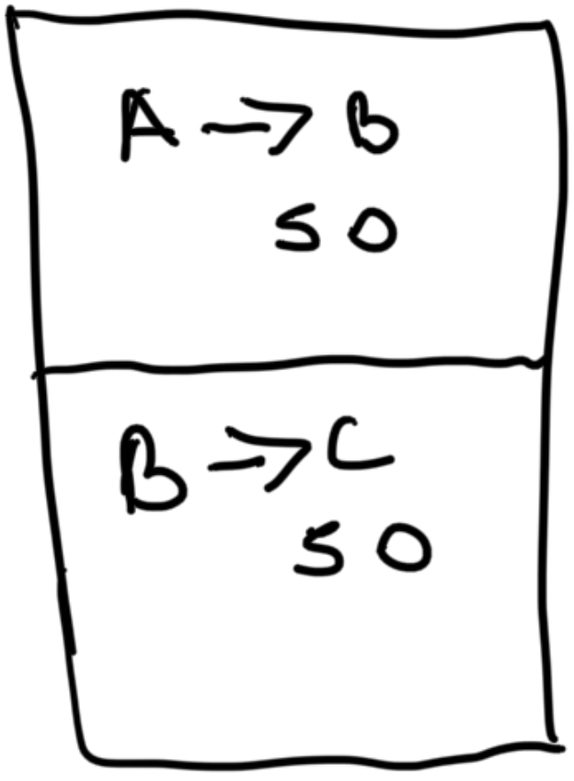
Q & A

Chess
- Design

- Command design pattern

Discussion

- Doubts
- Next steps



A -> B - SO

B -> C - SO

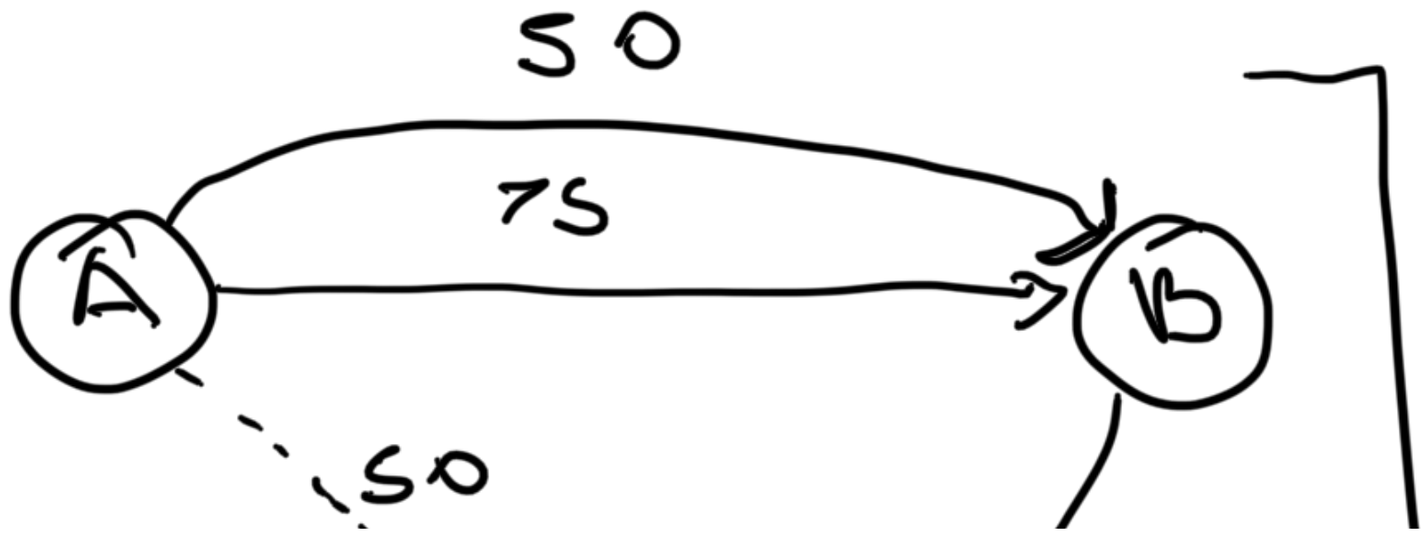
A -> C - SO

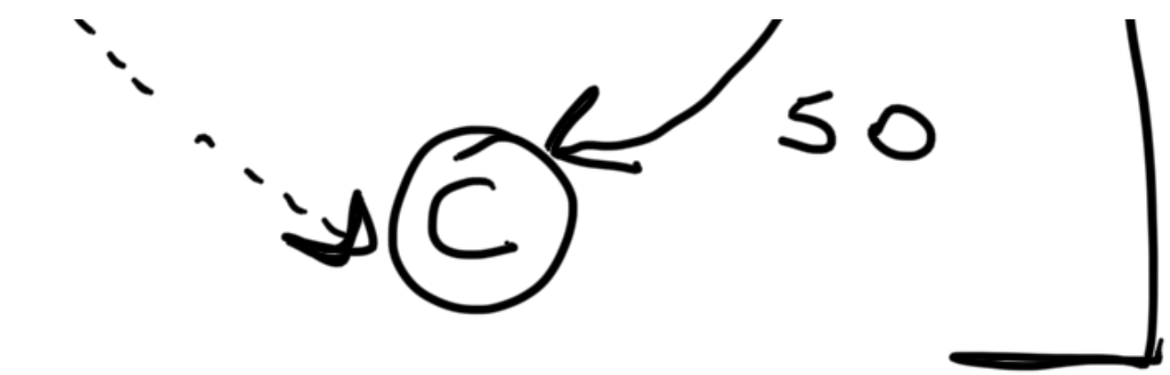
No. of transactions

An algorithm that can

→ settle all debts

→ minimises the no of transactions

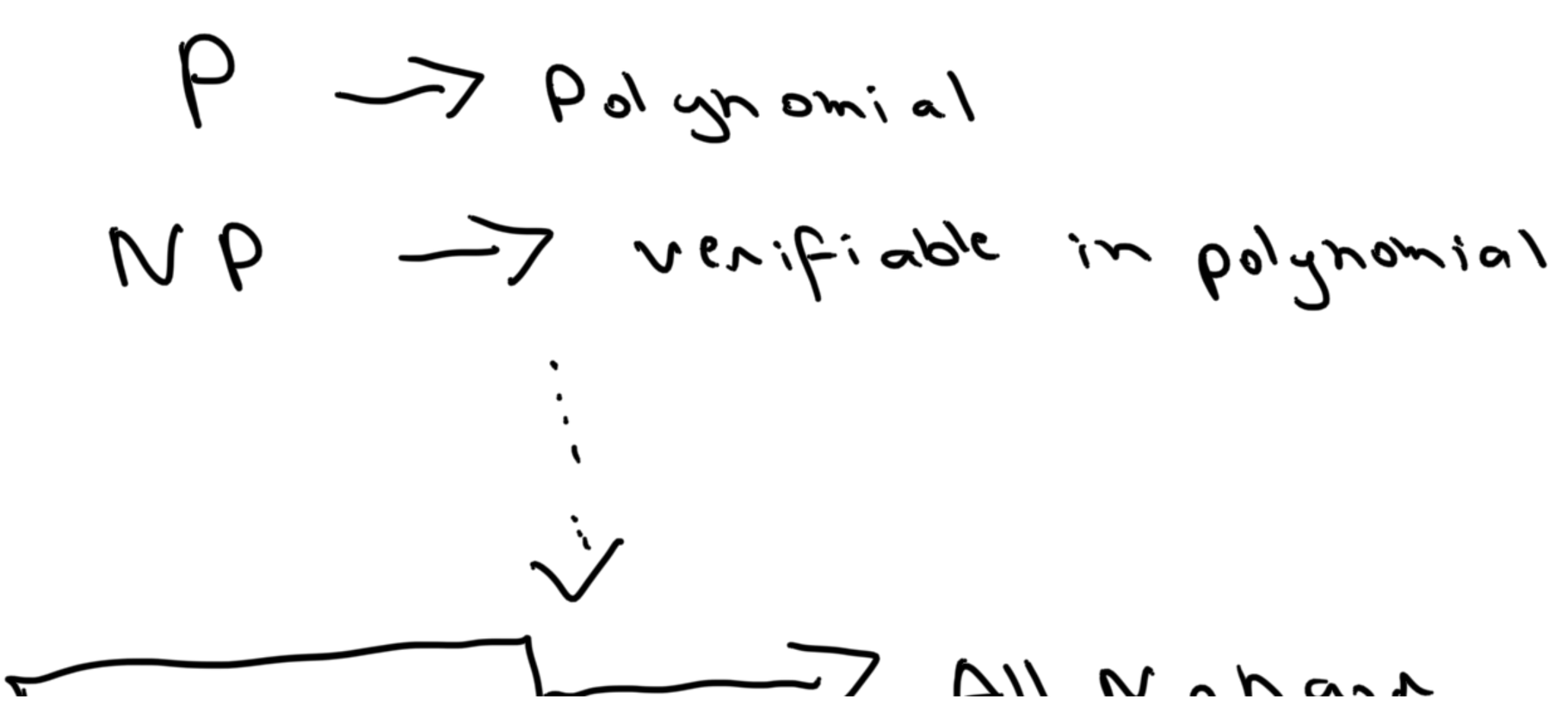




NP - complete

P → Polynomial

NP → verifiable in polynomial



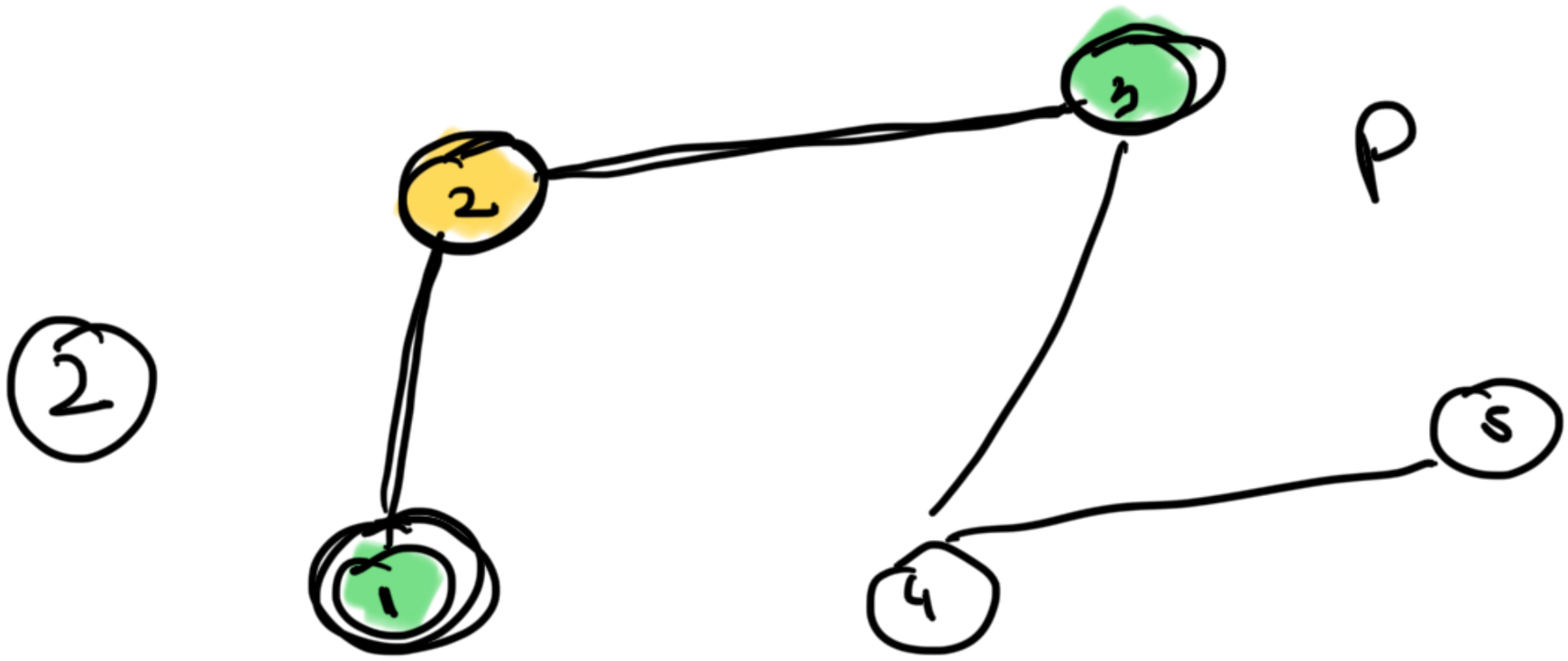
NP-Hard

NP-complete

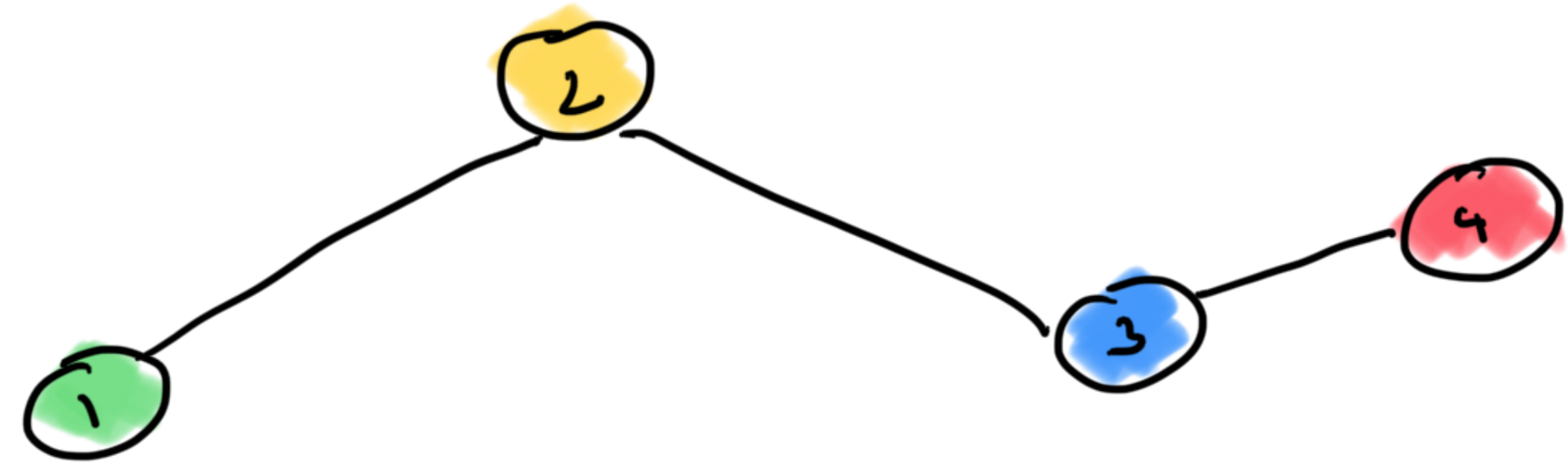
problems

---

node coloring



P



A → B

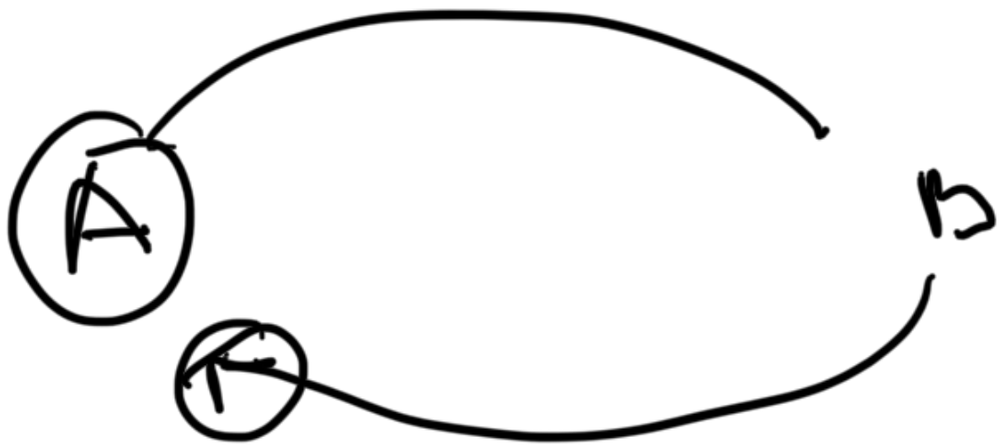
50

A → B

50

A pays B so

A pays B so



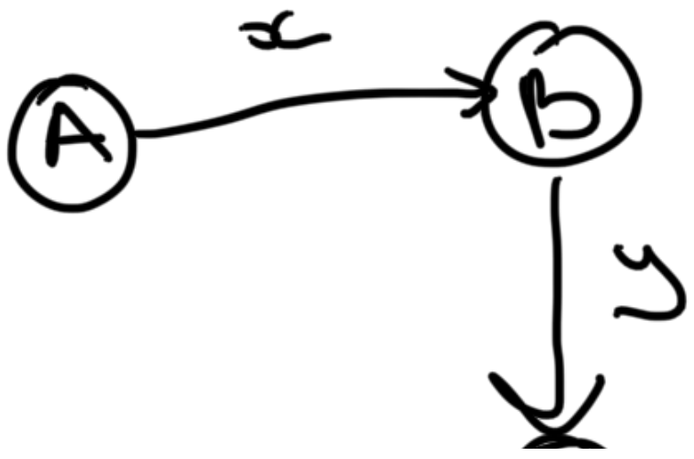
Balance

A - b - 50
A - C - 50

B - C - 50

C - A - 50

{ A: -100 , B: -50 , C: }

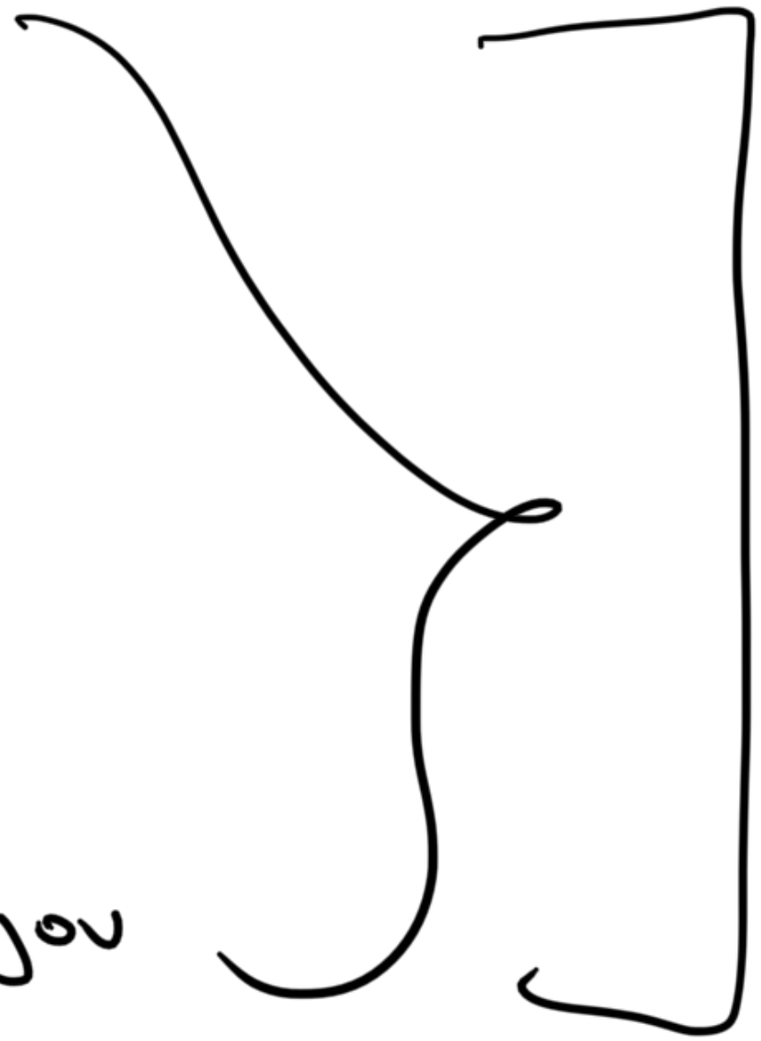


(C)

A → b

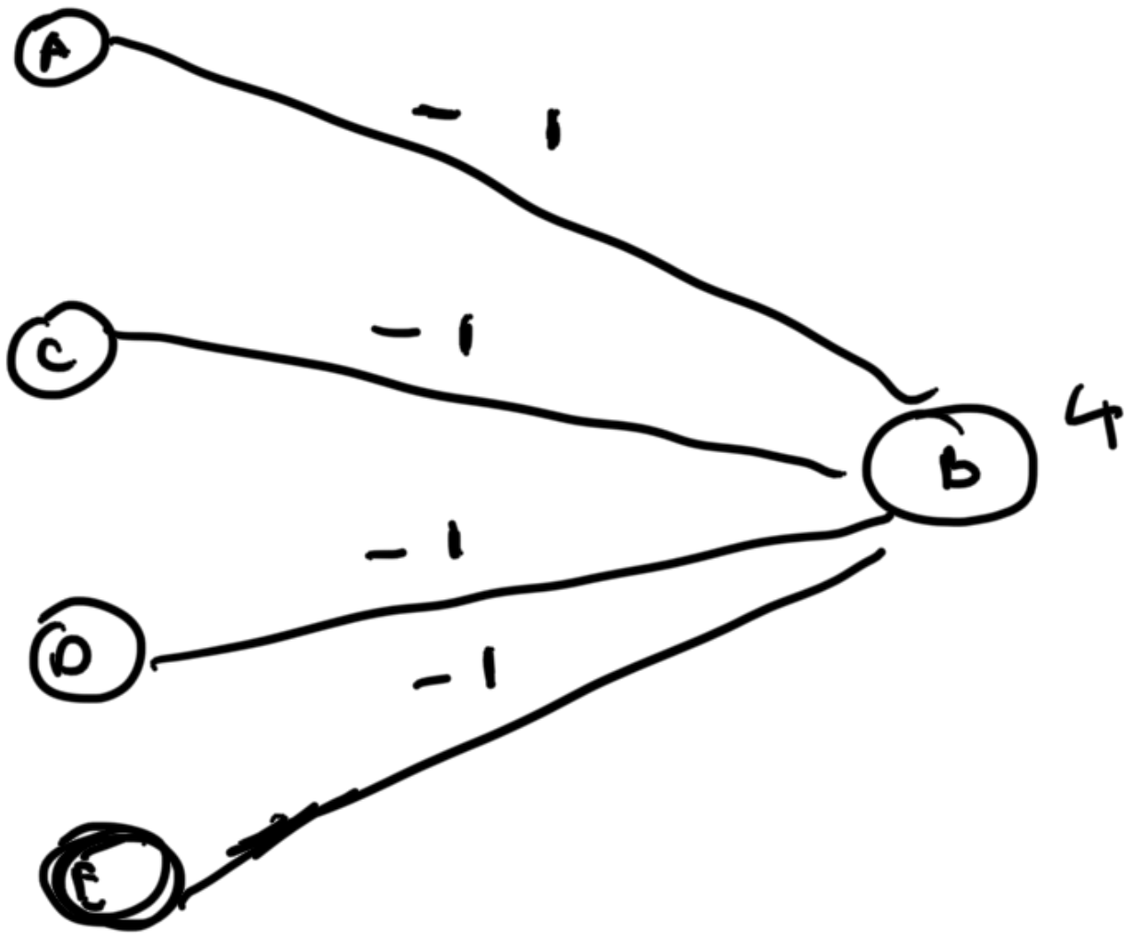
If value < 0:

Pay bank



If value > 0:

Bank pays you


A
C
D
E
B
4
- 1
- 1
- 1
- 1


A
E
C
C
4
B

D - ∈

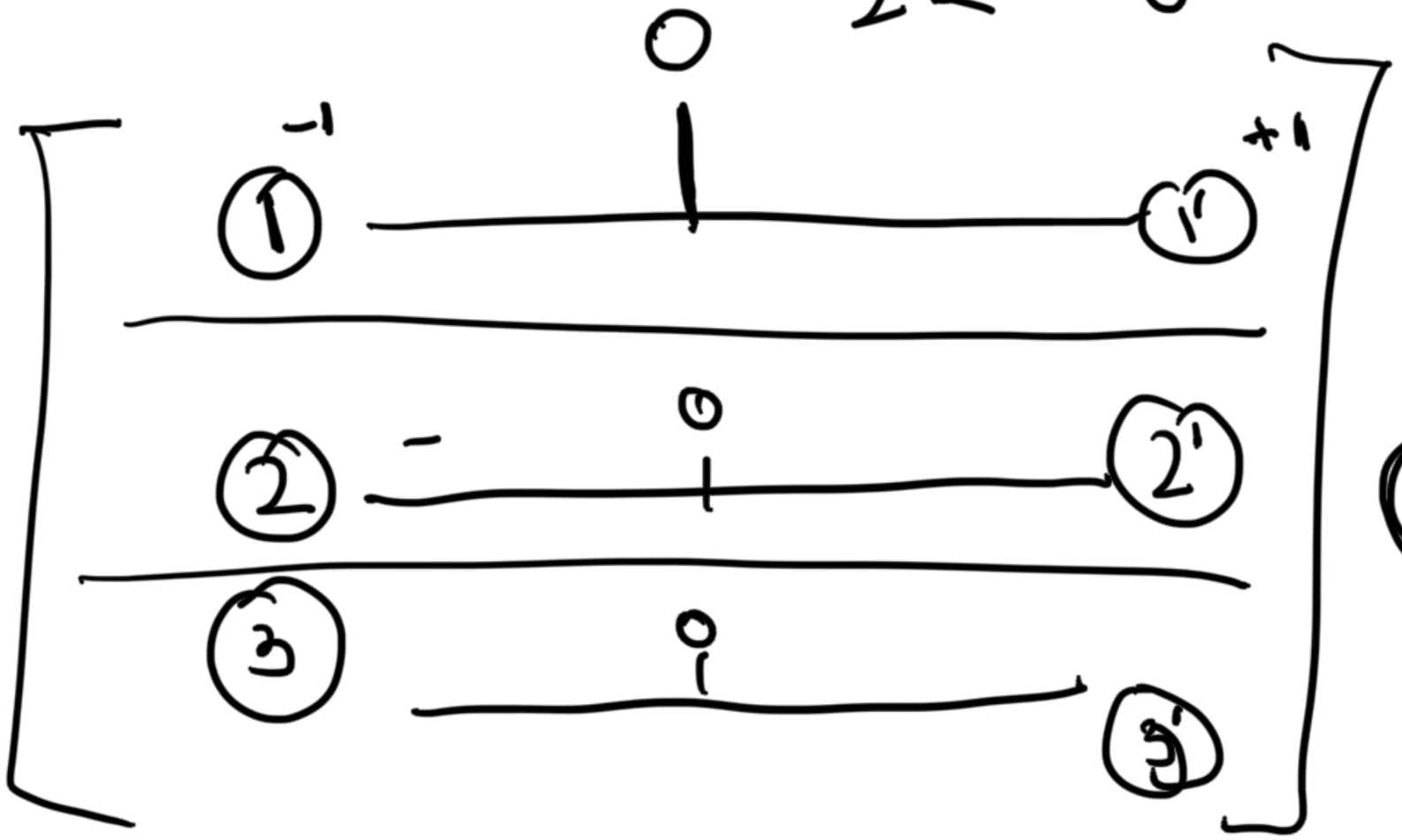
Optimal

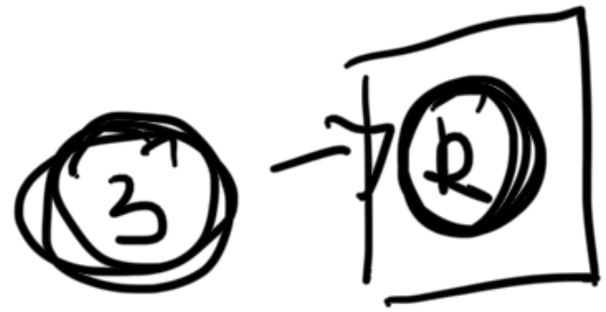
n - 1 →

2 R

R = -1 | R = +1

$2k \equiv 6$





$2k-1$

$= n-1$

---

n — nodes — people

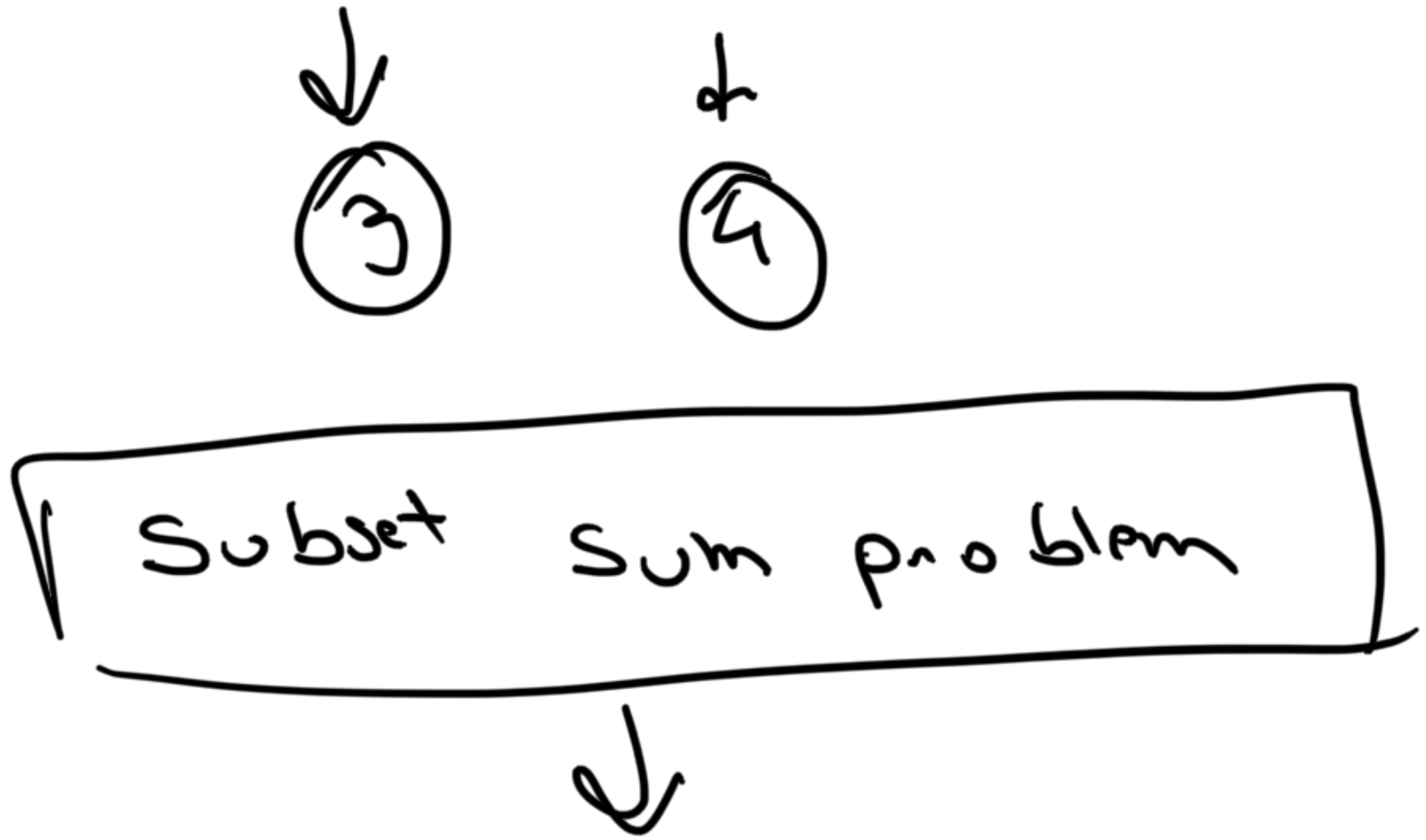
1 — 2 — 3

n-1 — transections

< n-1

subgraphs ⇒ 0

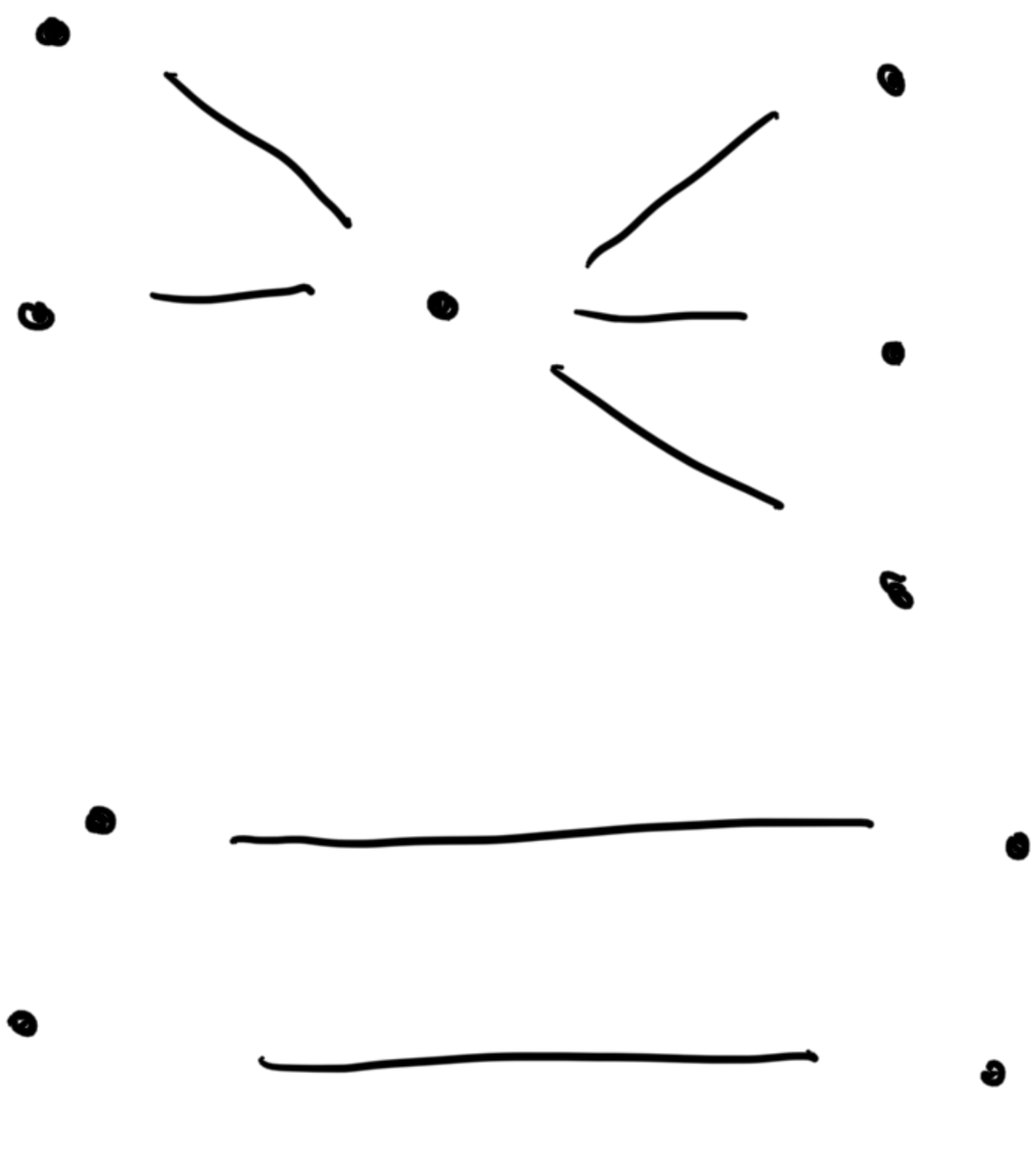
1 2 3 4 5



NP - complete

$O(n \cdot W)$ → # of graphs

2^{k-1}



A — B — 200

b — C — 300

B — D — 100

E — B — F00

2 (A) - -100

b - 200

c - 100

D - - 200

5

D - 200 ⇒ + 200

1. Computing net balance

2. Loop → m

   1. Pick the minimum → log n

      (0, 200)

   2. Pick the maximum → log n

      (b, 100)

3. Add one transaction

b − 0	200

4. If overflow / underflow

   overflow → b − 100

Underflow → B + So

③ Repeat

Java

① priority queue

② Tree Set

↳ Heap

n log n

① Endpoint — Settle up

/balance? groupid ✓

POST /group/{groupId}/settle

Group Controller

↳ Group Service

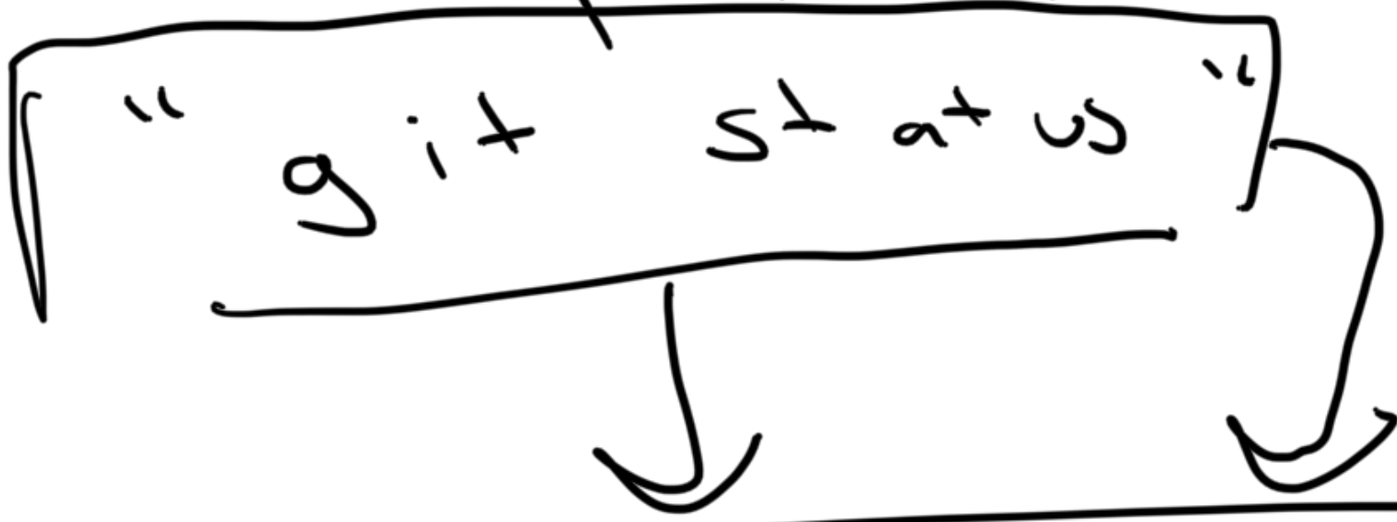
STRATEGY

Command

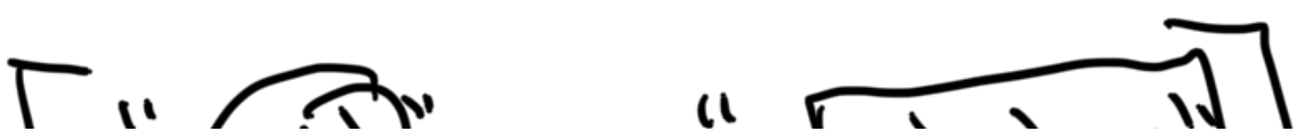
git status

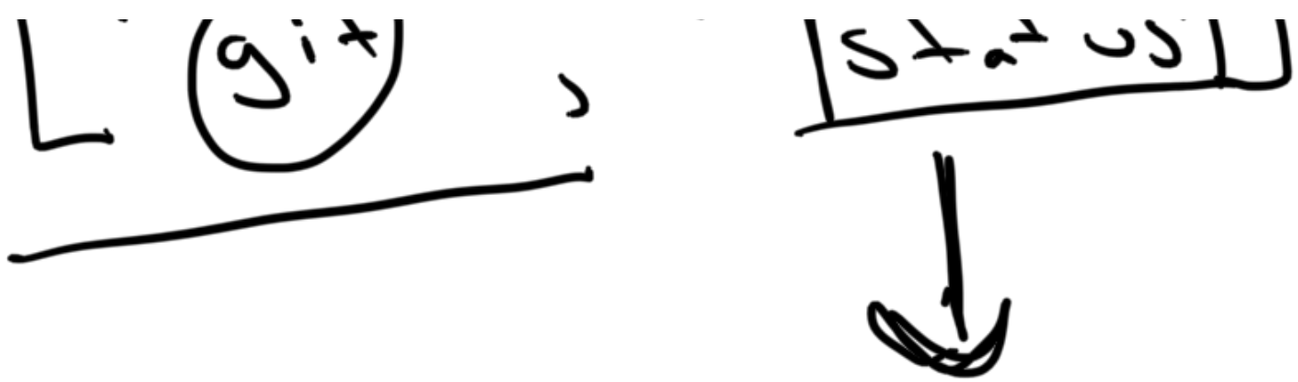
push origin master

① Read from command line



① Tokenisation

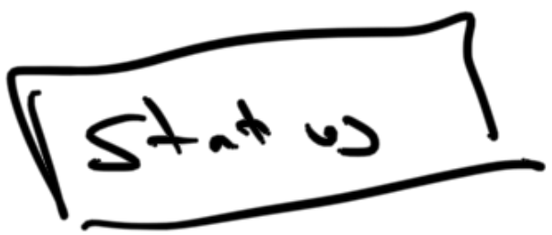




Command name.

" 

origin remote"



-> tokens [1]

-> tokens [2 - 5]

main() {

if(command == "status") {

call Status API

} else if(== push)

call Push API

}

Status)

Push (control)

Command interface

interface Command {

boolean matches (String name)

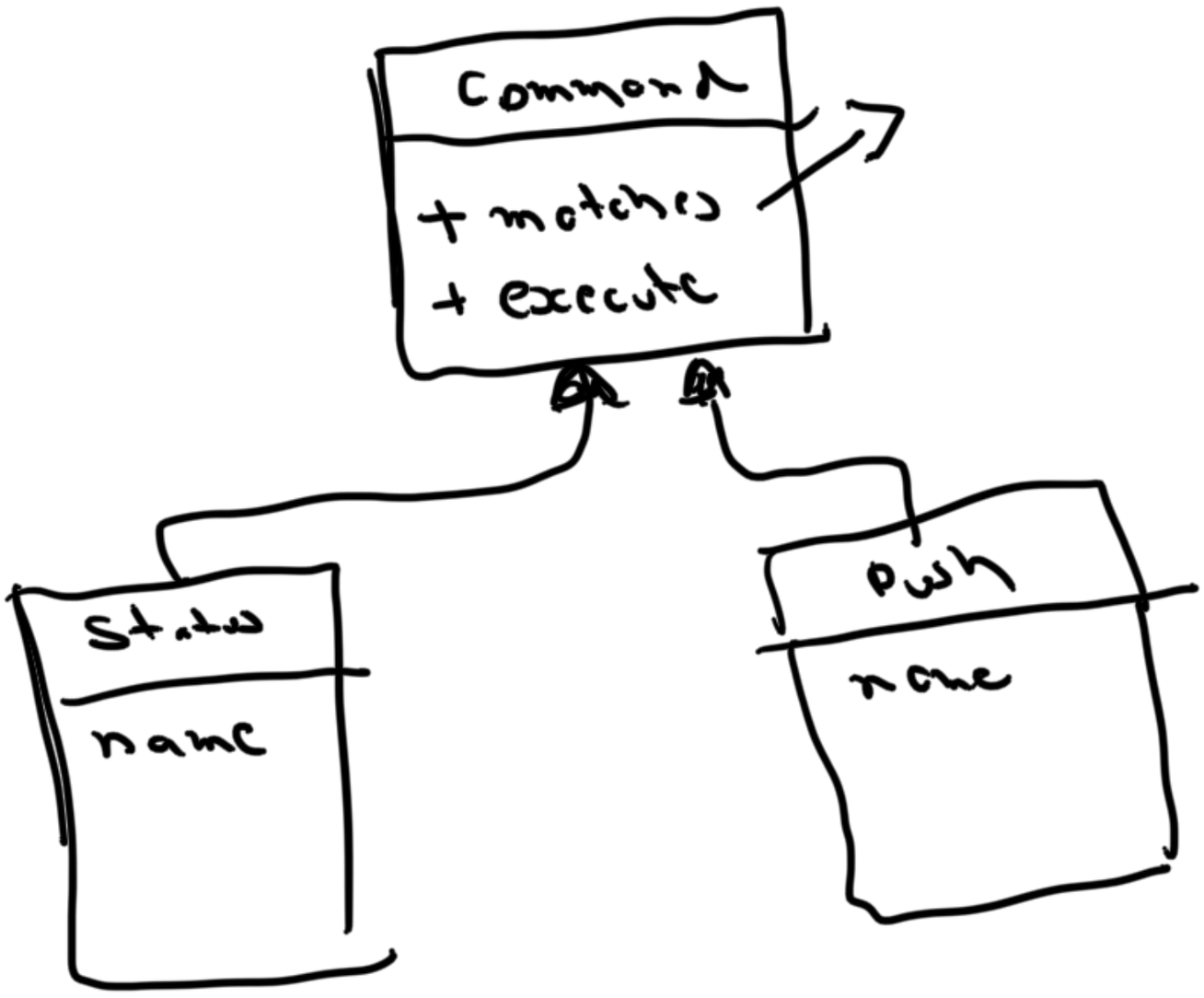
void execute (String[] tokens)

}

List<Commands>

→ input : stdw

→ if any command
can handle "stats"



Window

↓

git

Os

↓

git

System

→ All these commands are usable

→ At runtime add / remove a command

→ register

→ deregistering

CommandRegistry

→ HashMap<Name, Command>

→ register

→ dc.register

→ execute() {

→ matches

→ executes

}