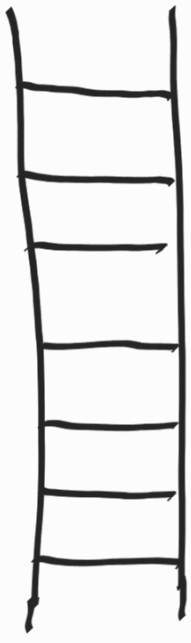
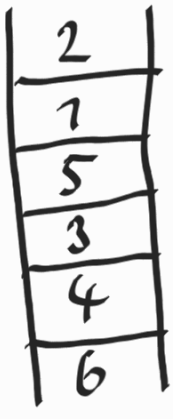
A

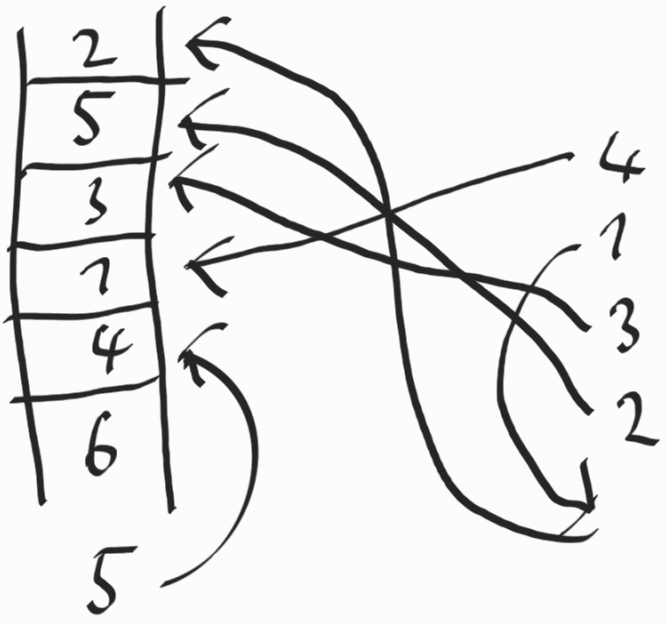
B

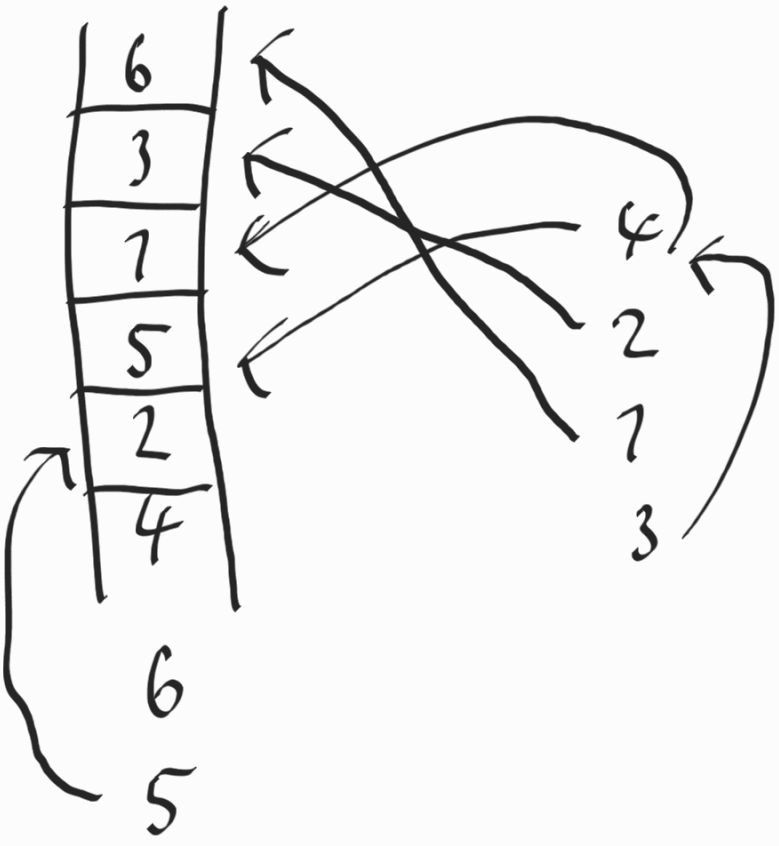


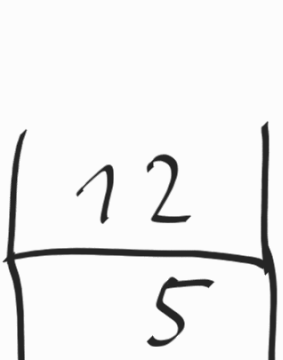
2
1
5
3
4
6




6
5

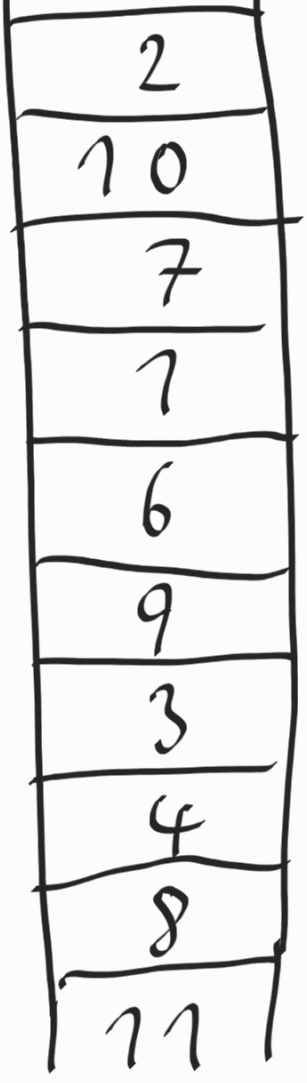

2
5
3
7
4
6
5
4
1
3
2


6
3
7
5
2
4
6
5
4
2
1
3


12
5


0
1
3
2


2
7



1
2
3
4
5
6
7
8
9
10
11
12

1
2
3
4
5
6
7
8
9
10

27
18
6
3
12
10
4
14
16
9

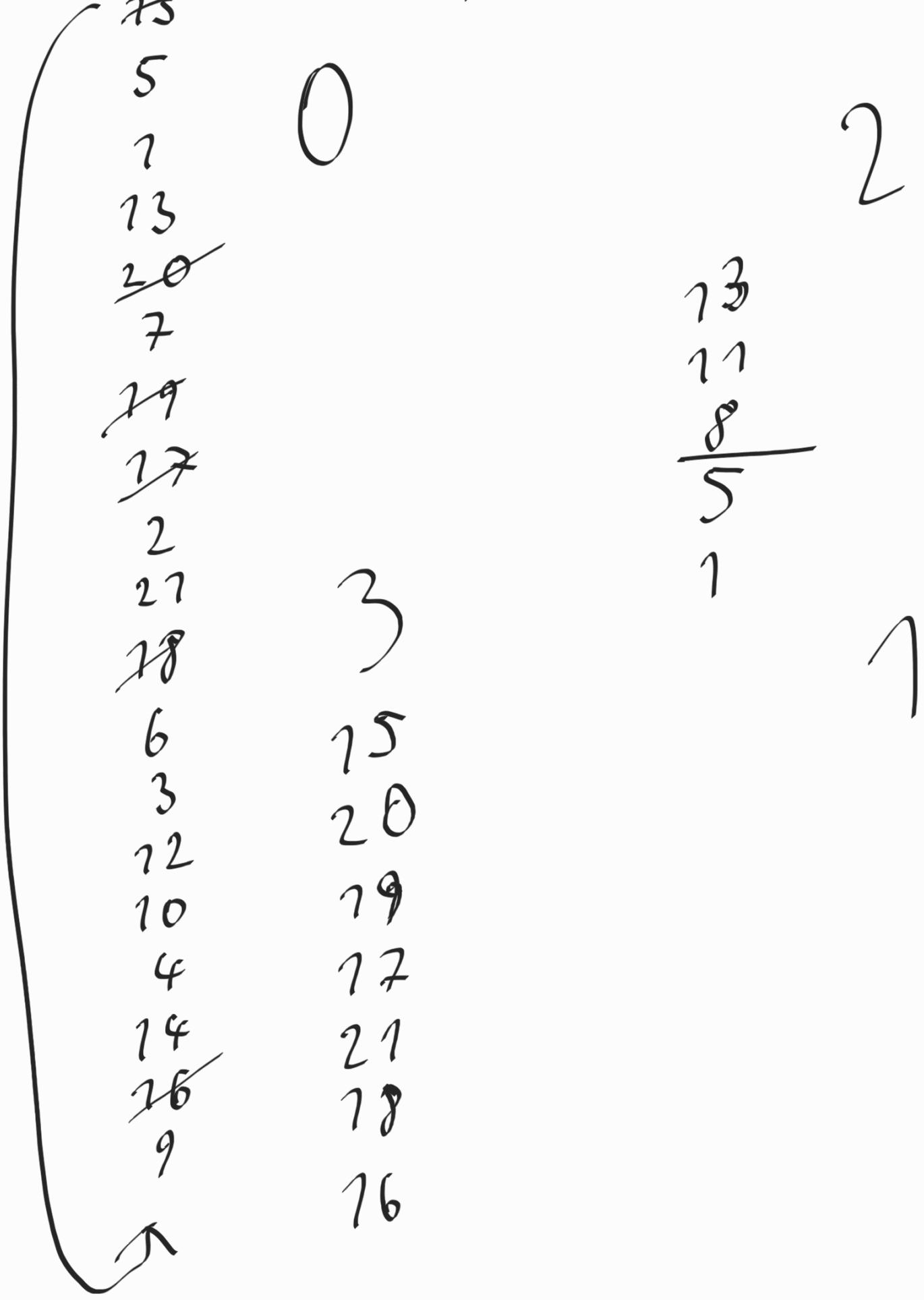
12
13
14
15
16
17
18
19
20
21
3
4
5

~~7~~
~~6~~
1
2

8
11

A

B


0
2
3
1
5
1
13
20
7
19
17
2
27
18
6
3
12
10
4
14
16
9
15
20
19
17
21
18
16
13
11
8
5
1

001
110
000
010
101
011
111
100

001
101
011
111

10

010
110
000
100

00

000

001
010
11 011
100
101
110
111
010
10

001
000

01100

1001
1110
0000
1010
0101
0011
1111
0100
0001

1000

1100

0101

0010

0110

0111

1011

10

(11)

1000

1001

1011

1100

1110

1111

1010

0000

0101

0011

0100

0001

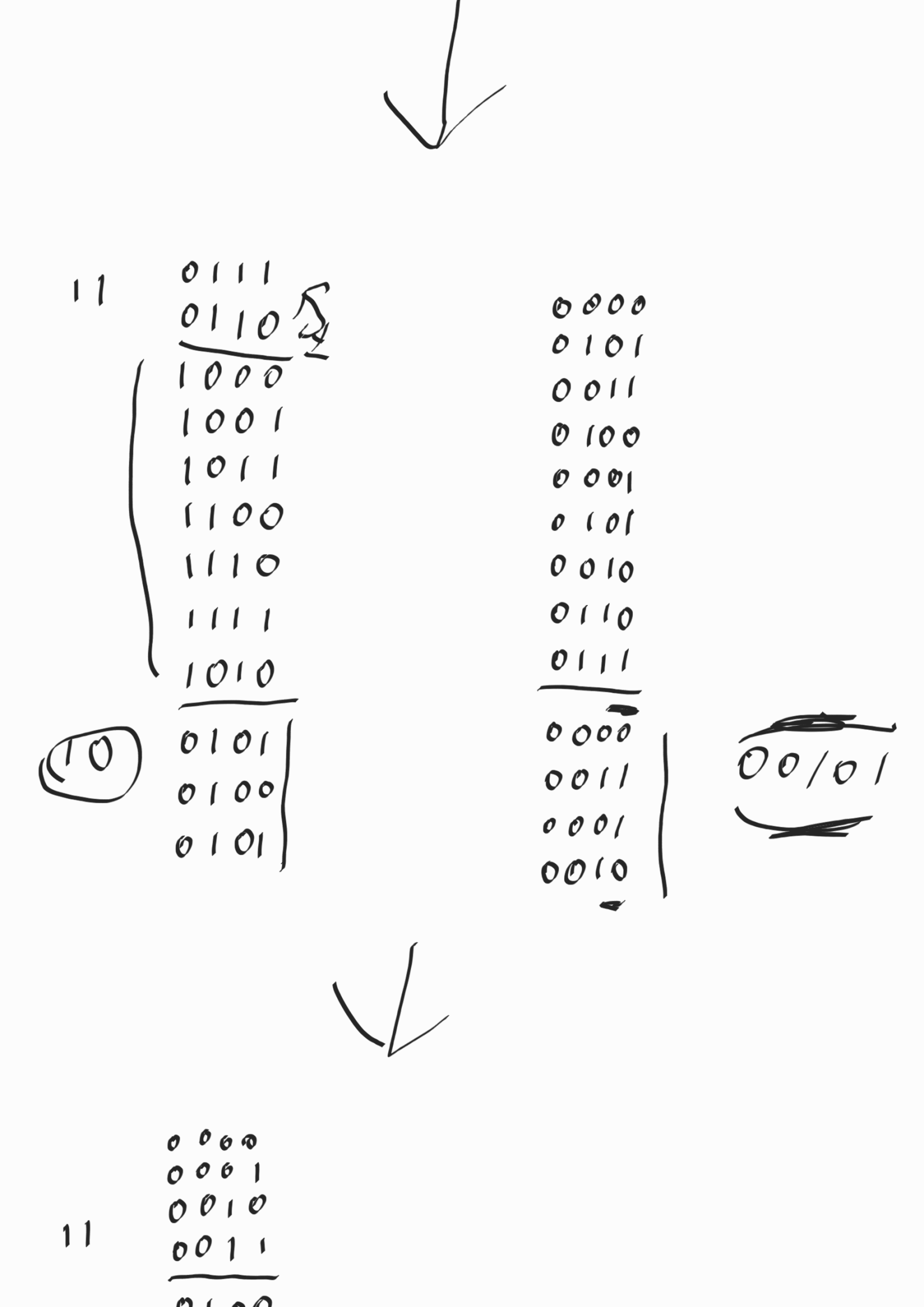
0101

0010

0110

0111

01/00



0100
> 0101
- 0101
0110
0111

1000
1001
1011
1100
1110
1111

~~0010~~

0010

10

~~0000~~
~~0011~~
~~0001~~
~~0010~~

0000
0001
-

01/01

30 values

1-10 | 11-20 | 21-30

21-23 | 24-26 | 27-30

27 29 29-30

base case of 3: push

although, I might need a check at the very beginning

if from not in order
swap from
push
if to not in order

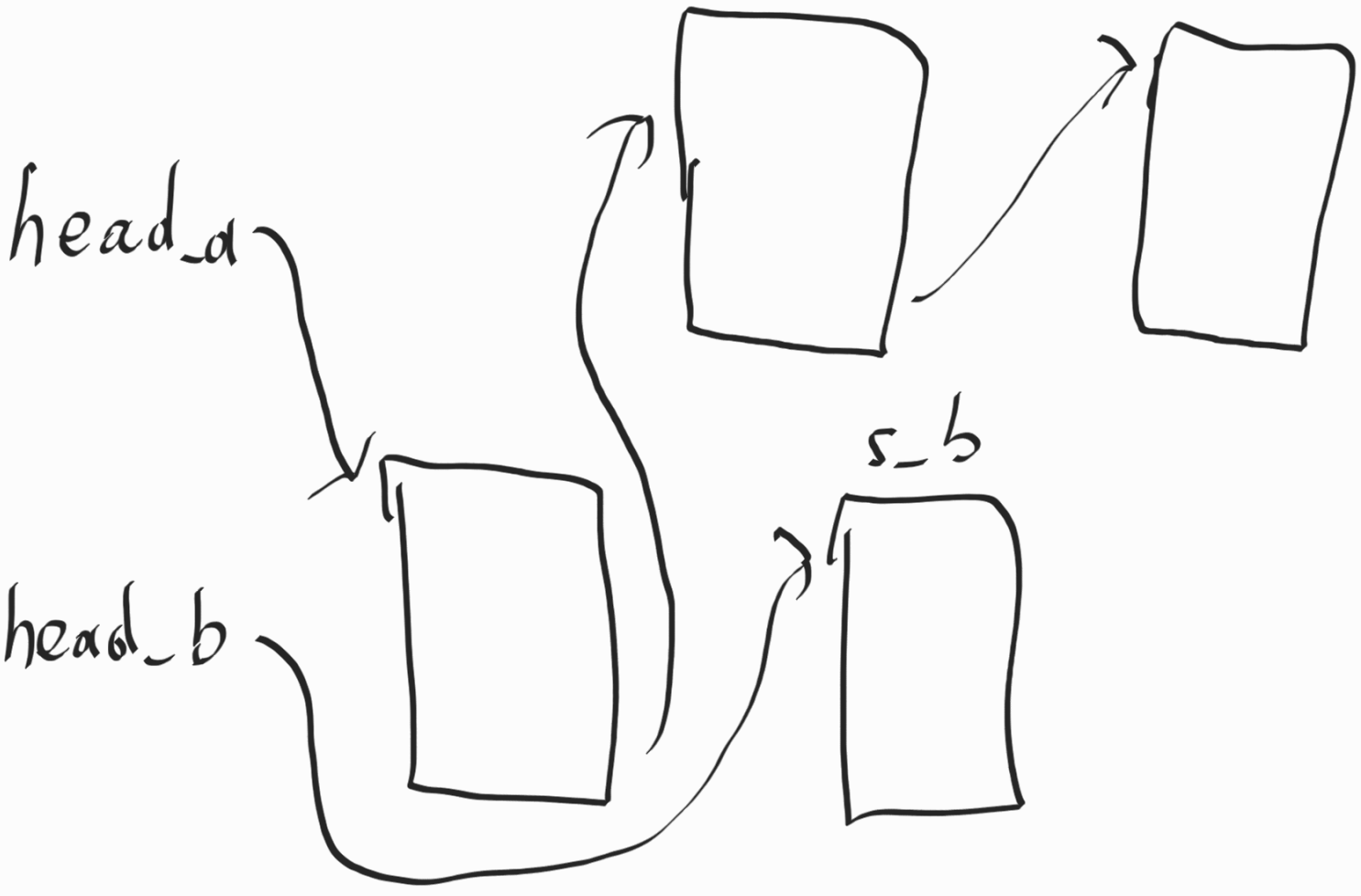
where I am

swap to

push

if to not in order

swap to



17-32

33-50

1-16

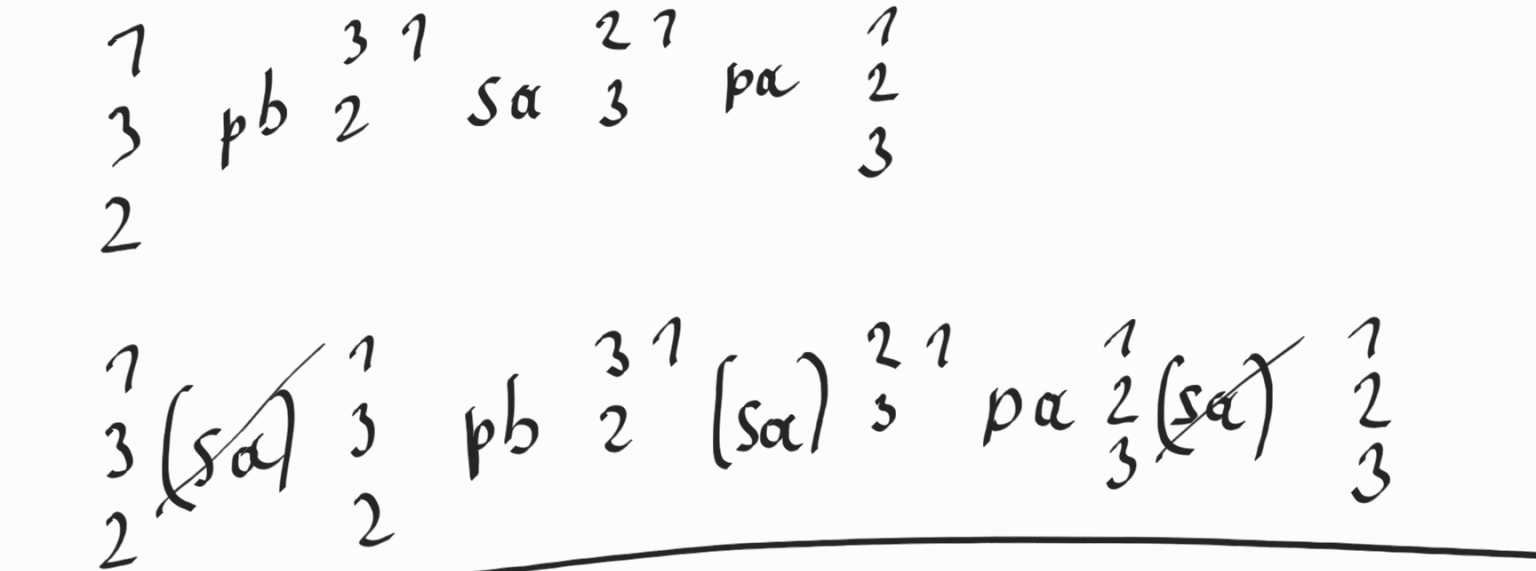
39 - 44

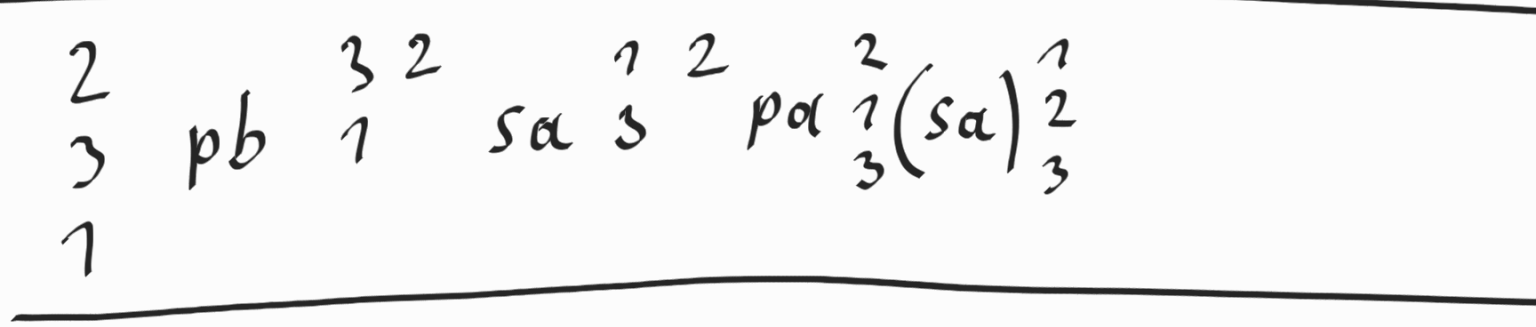
45 - 50 33 - 38

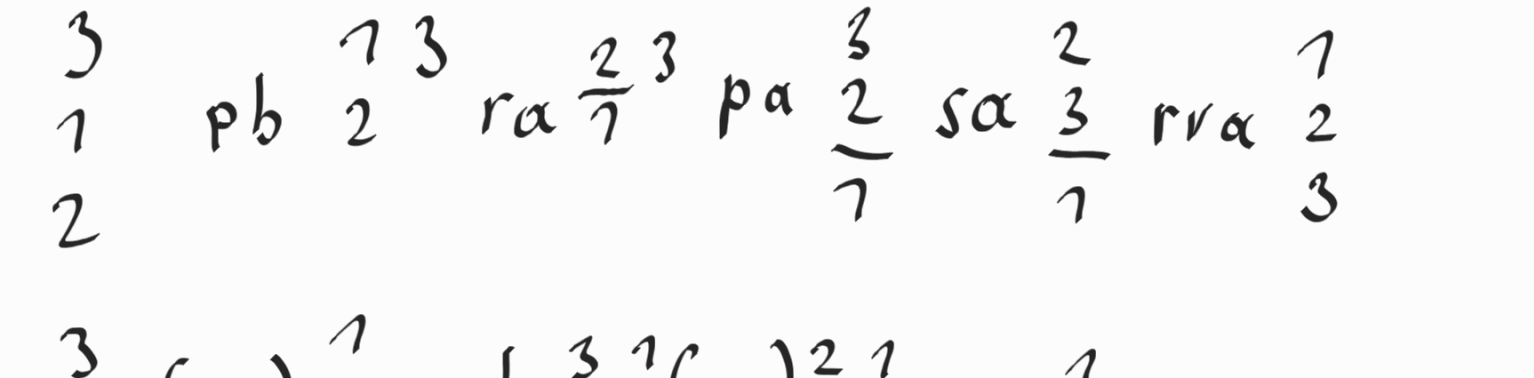
47-48

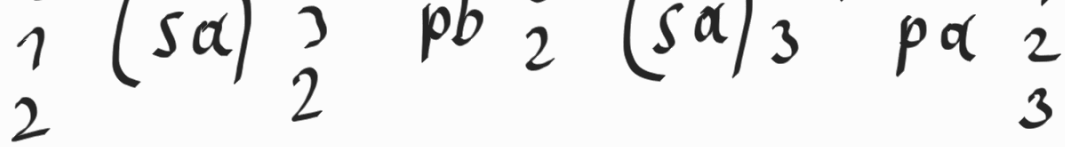
49-50 45-46

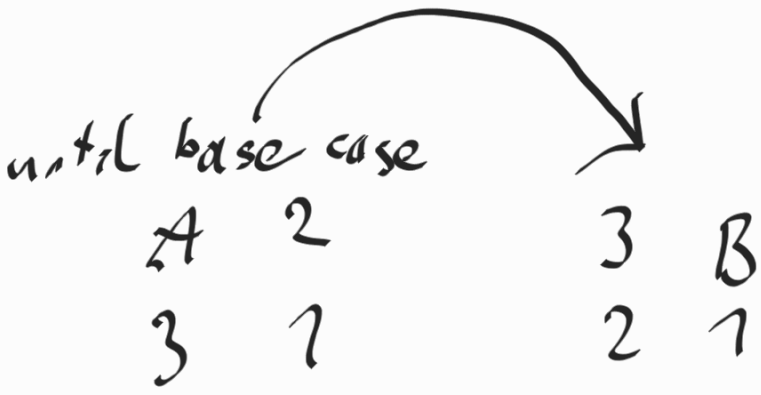
sort_three_a:

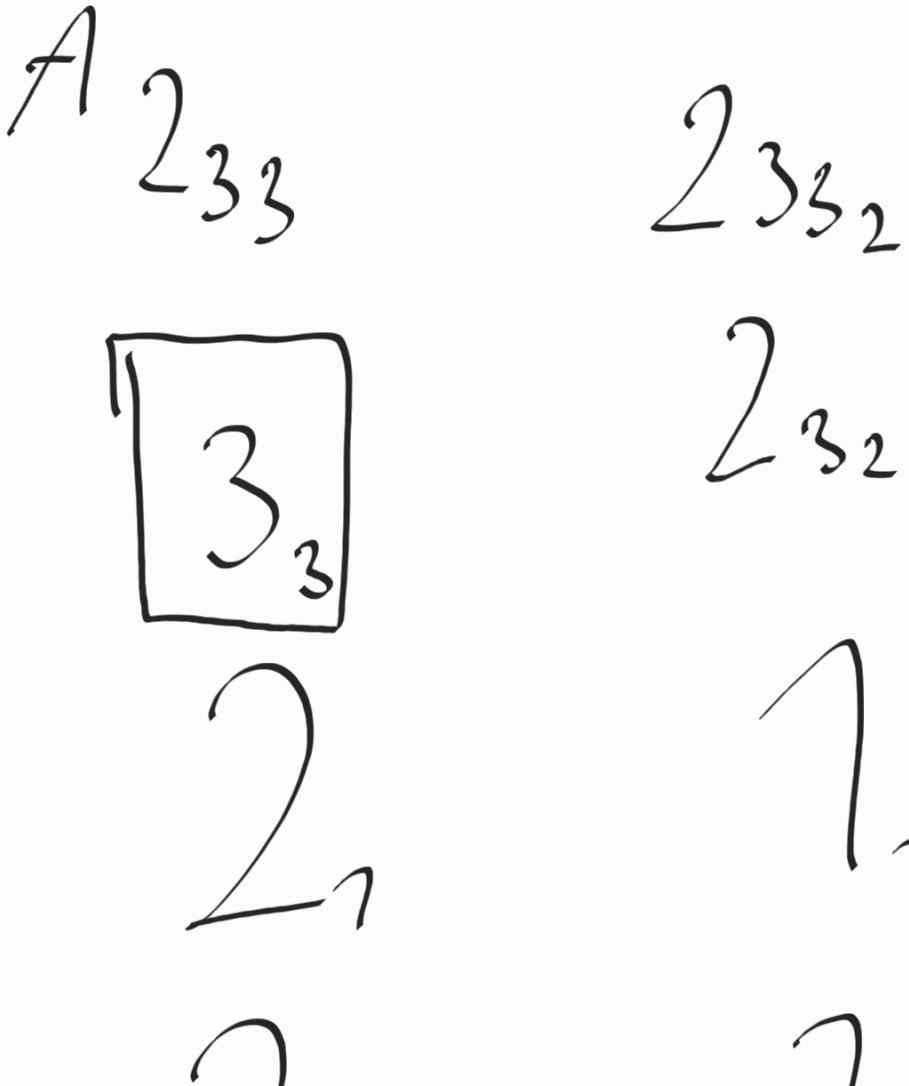









until base case
A 2
3 1
3 B
2 1

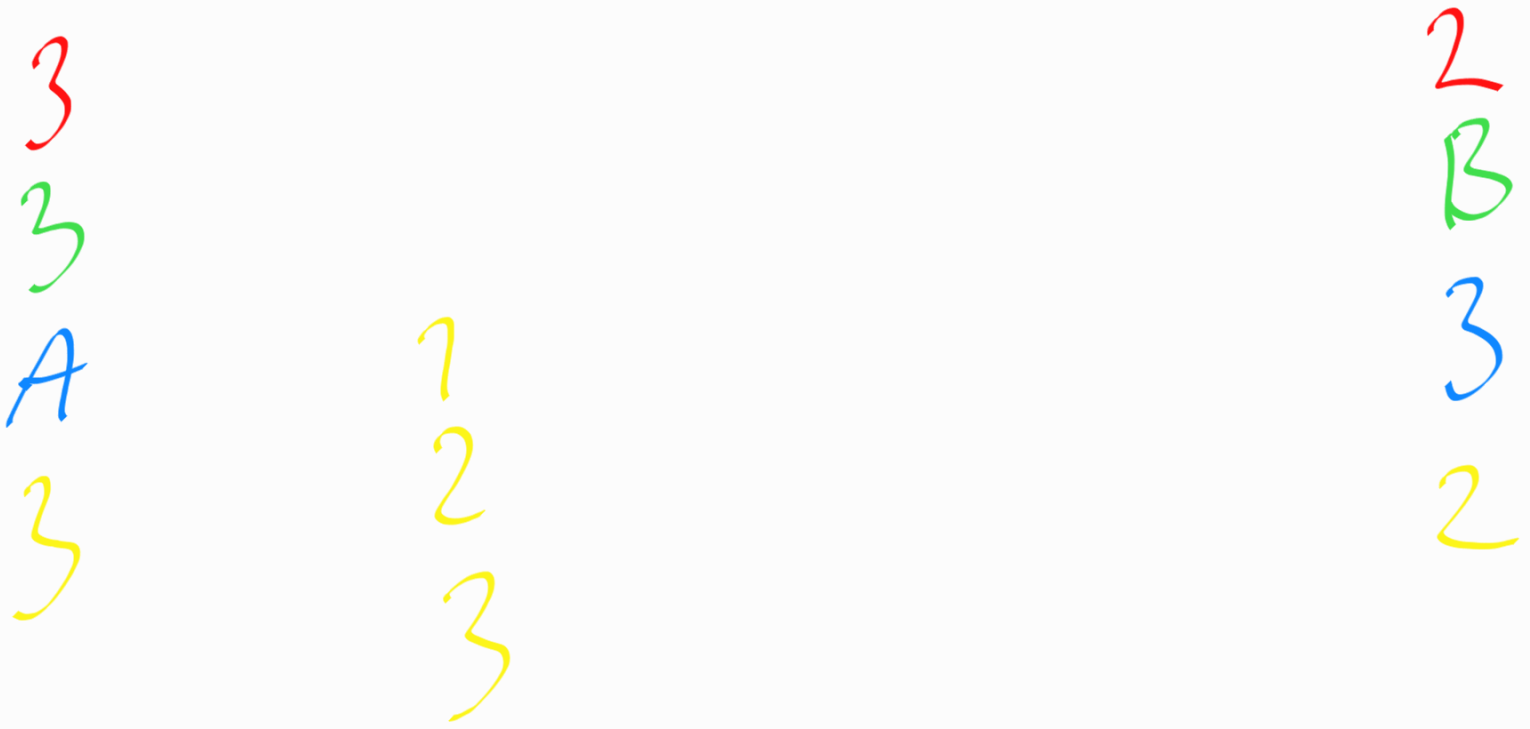


2_2 1_2

2_{3_3} 1_3

2_{3_1}

$2_{3_{1}}$


3
3
A
3
1
2
3
2
B
3
2


A

2

2

1

3 2

3 B

3 2

A 2

3 1

A 1

2 ↔ 1

1 B

1

1

1

B

quadrant

amount

A

B

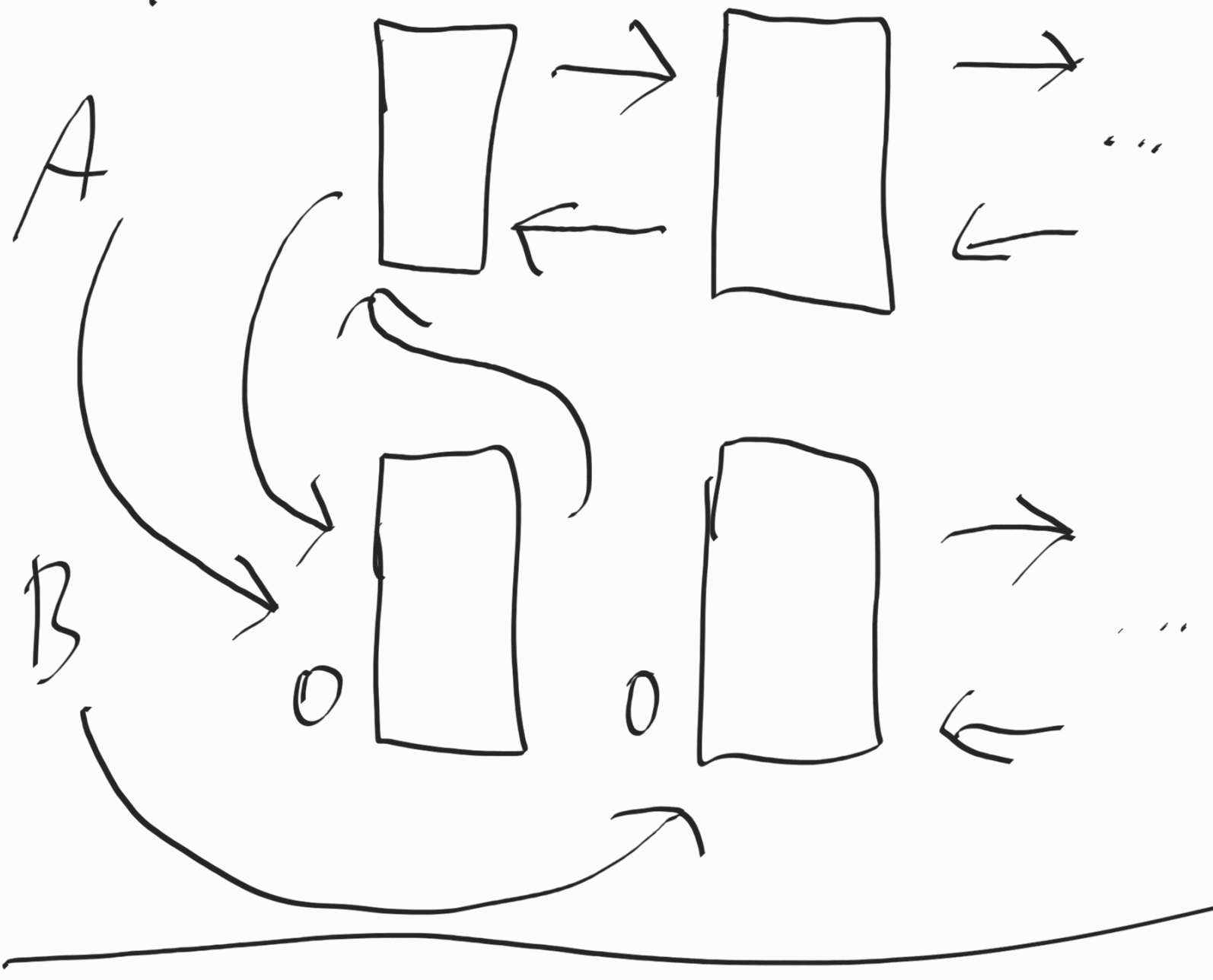
6 5 5 4 4

4 4 6 6 5

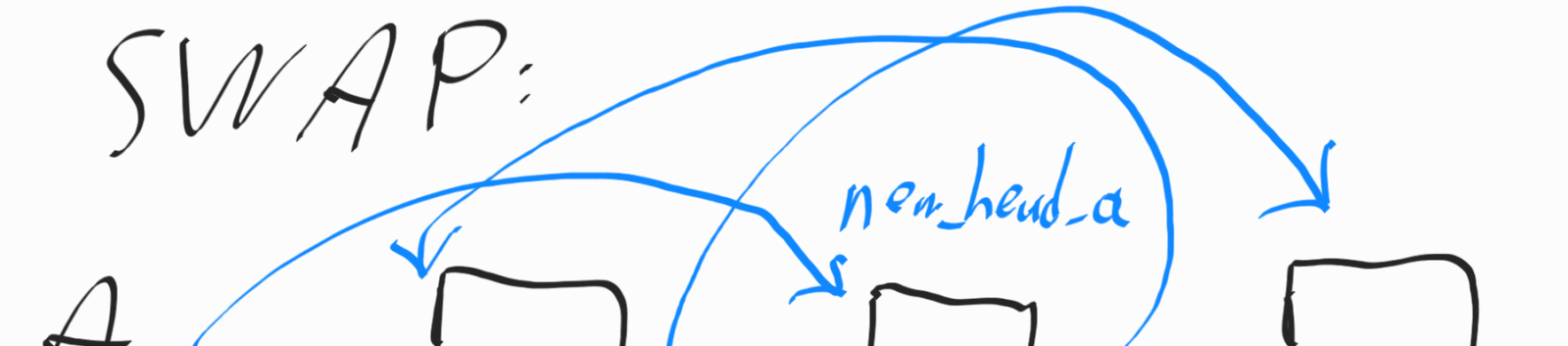
5 6 4 5 6

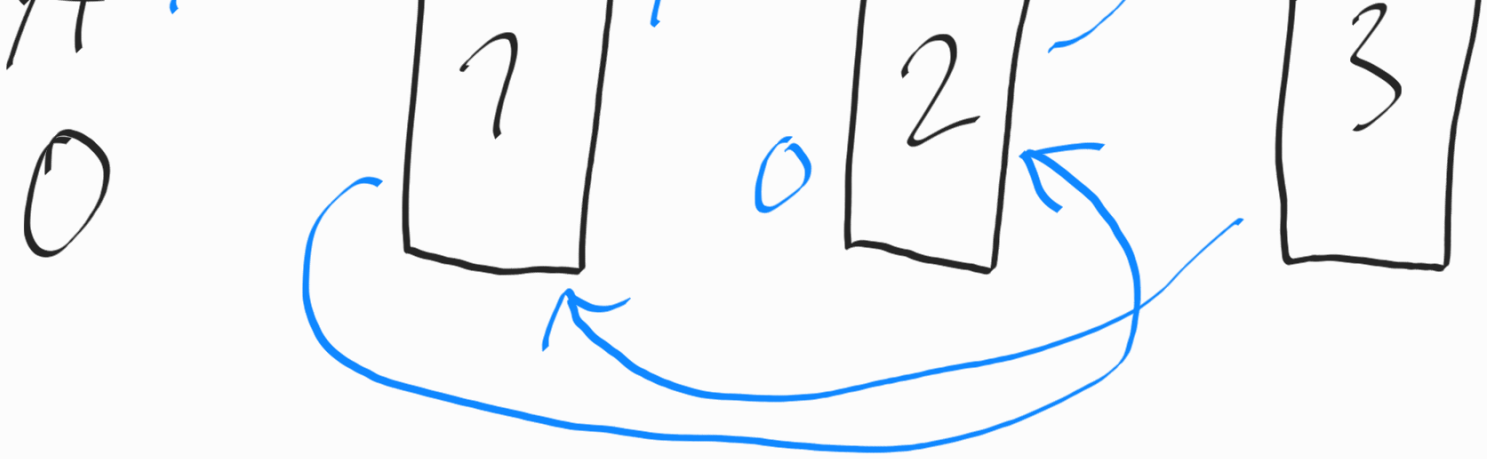
8
9

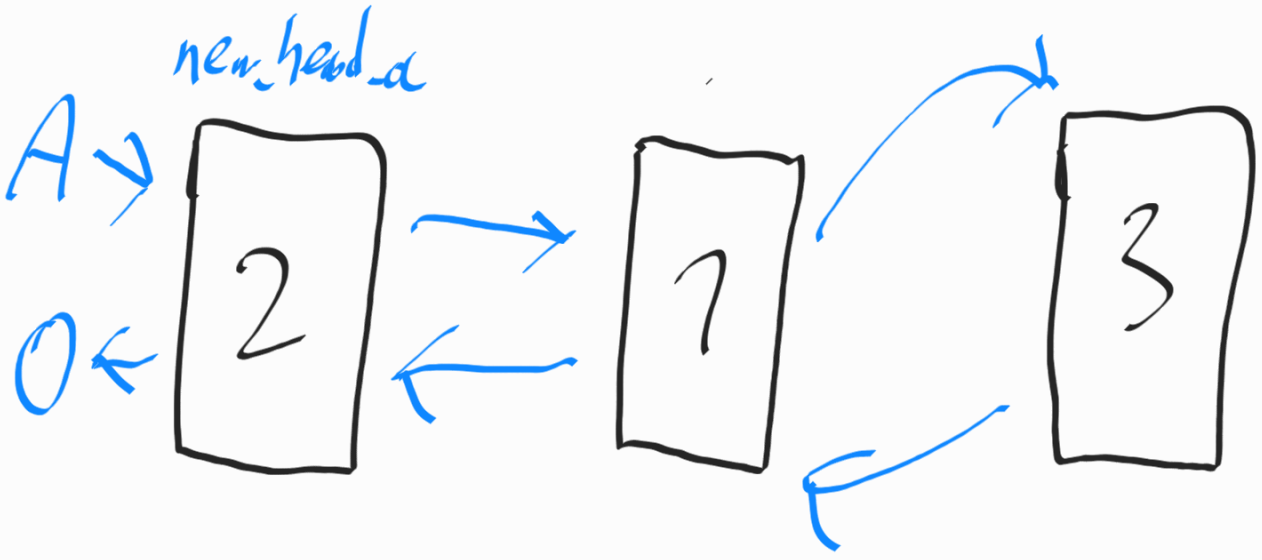
PUSH:



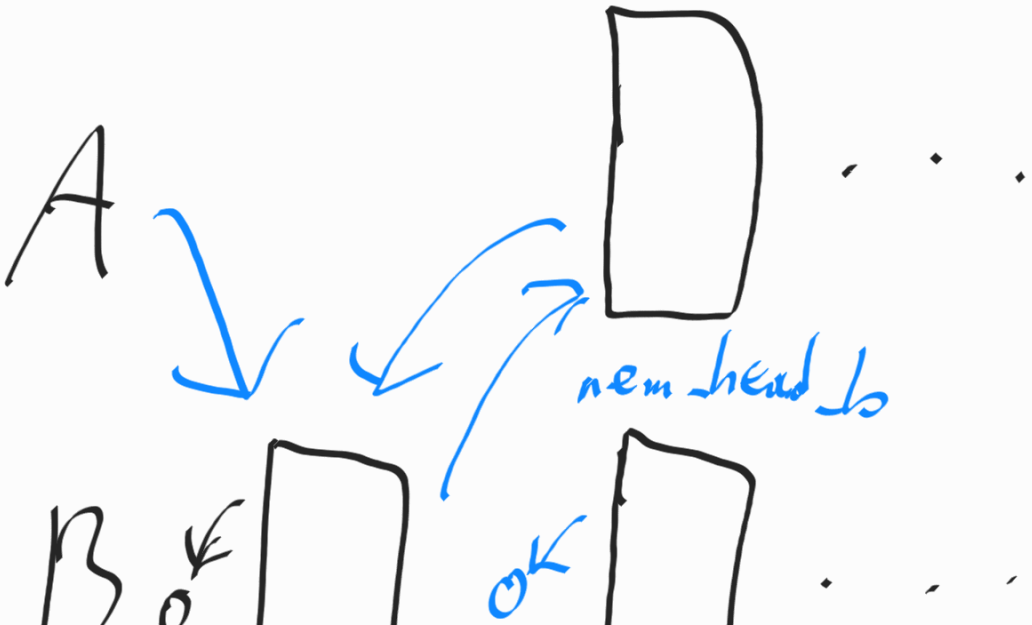
SWAP:

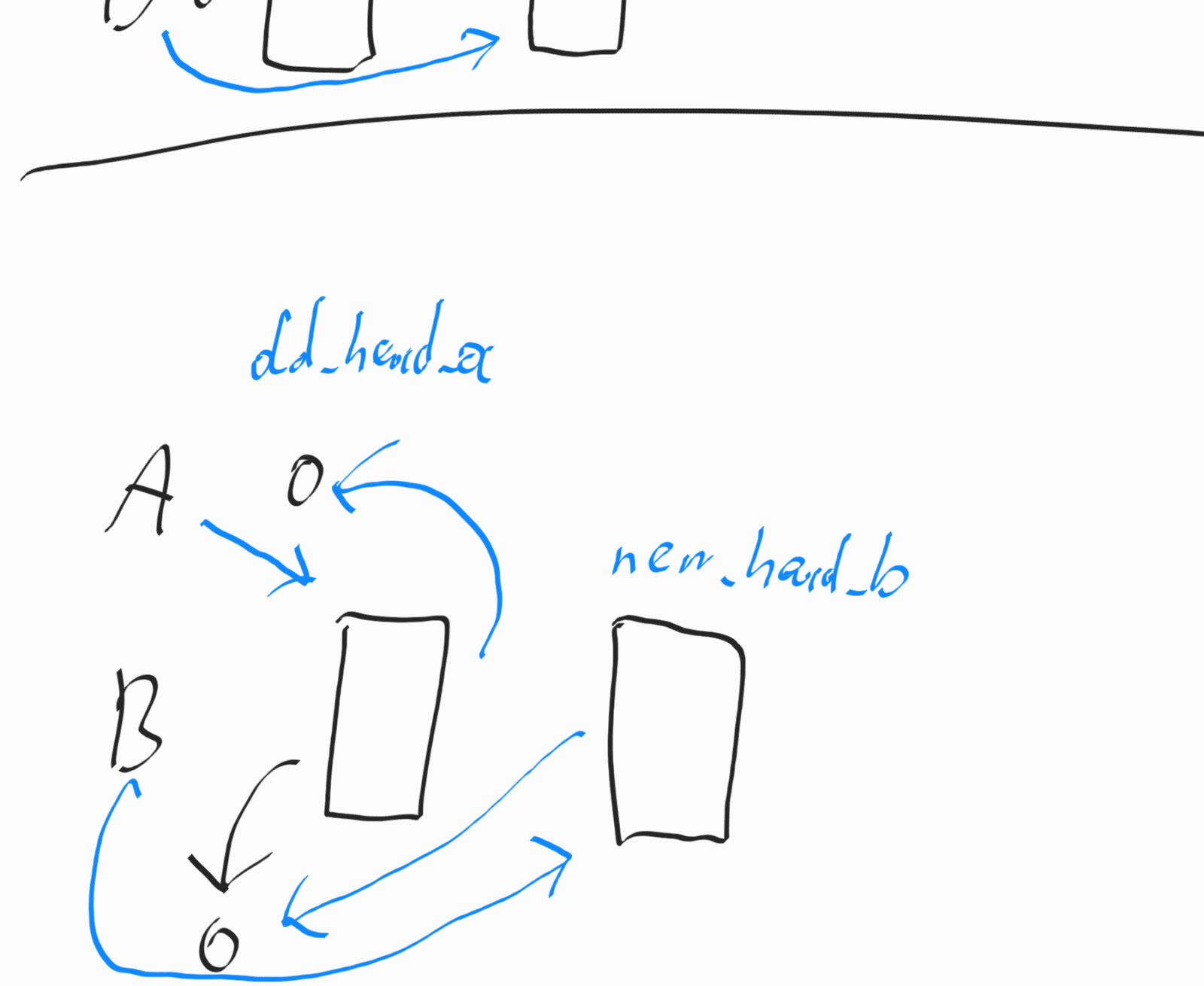






PUSH:

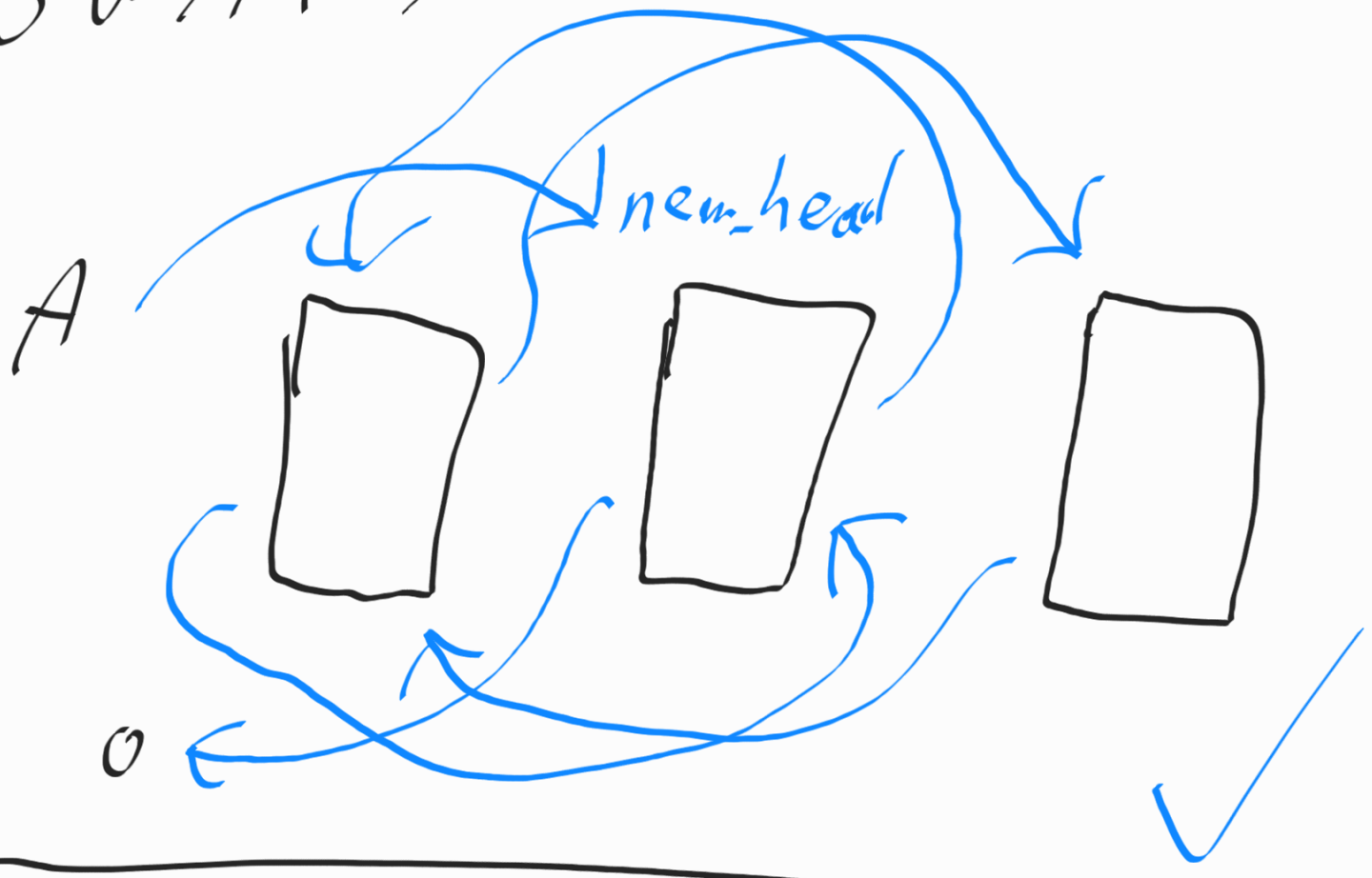


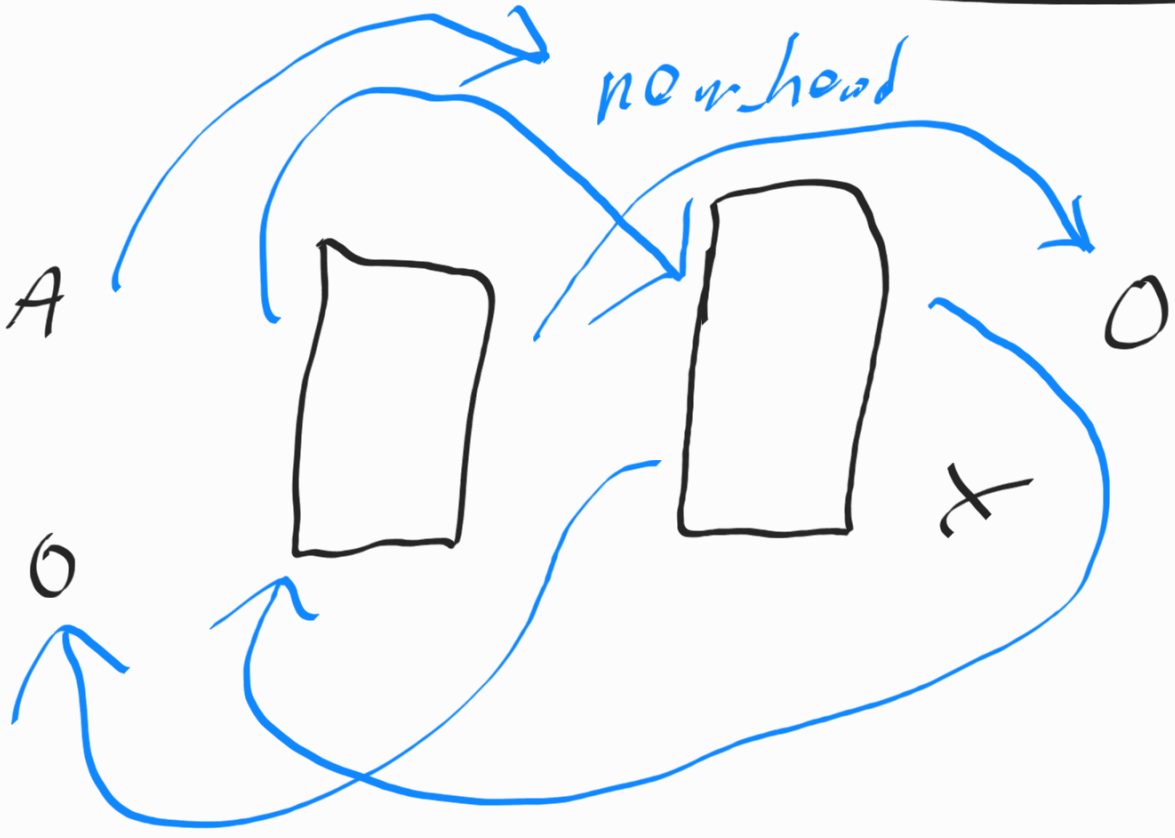

old_head_a
A
0
new_head_b
B
0

A
0
new_head_b
B
0

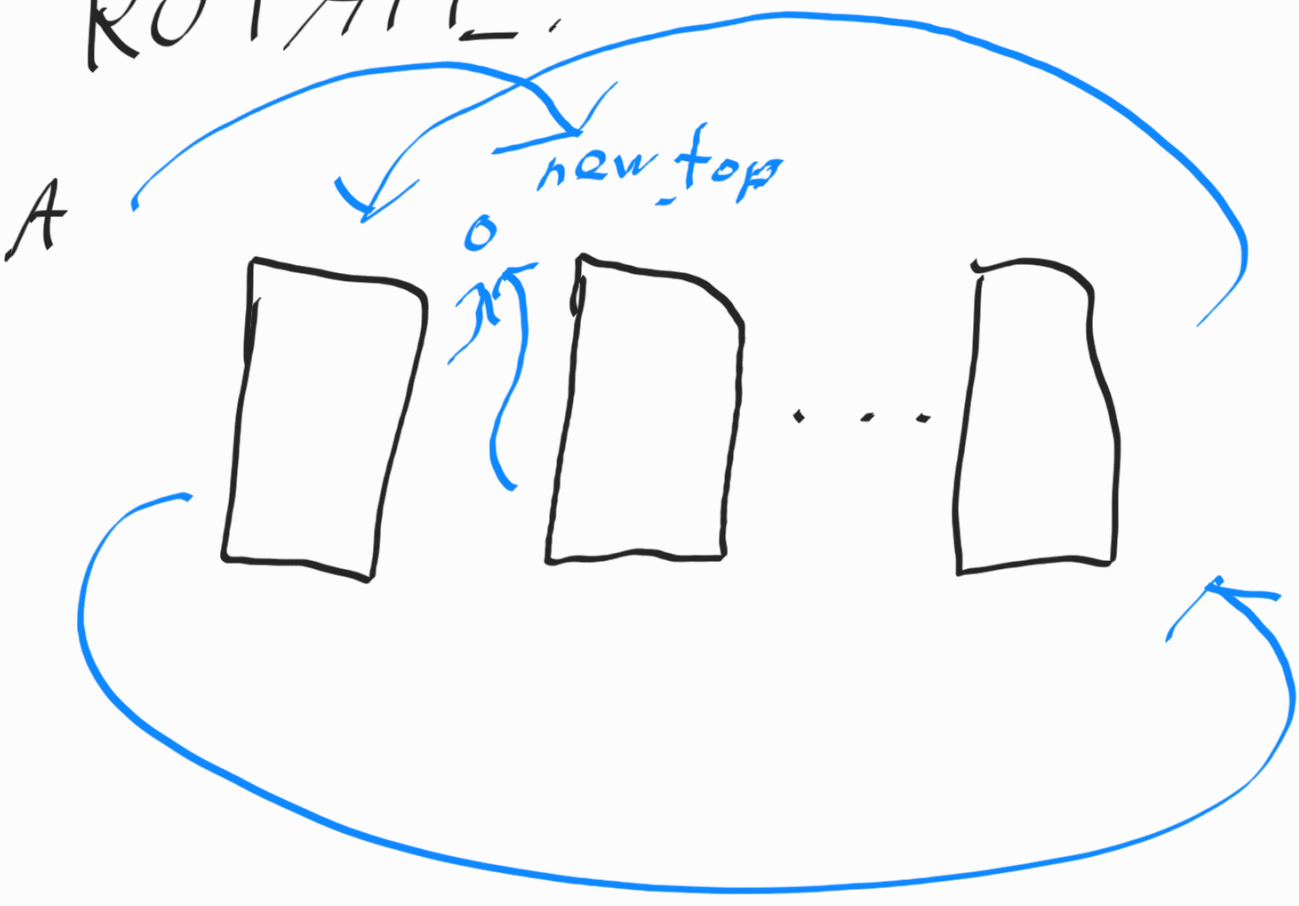
O

SWAP:

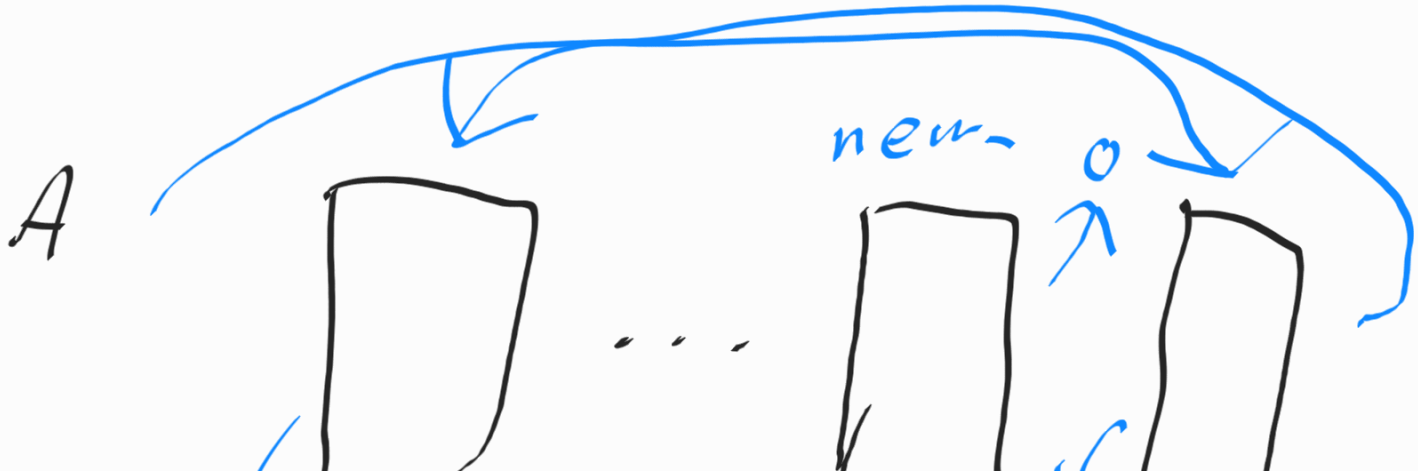


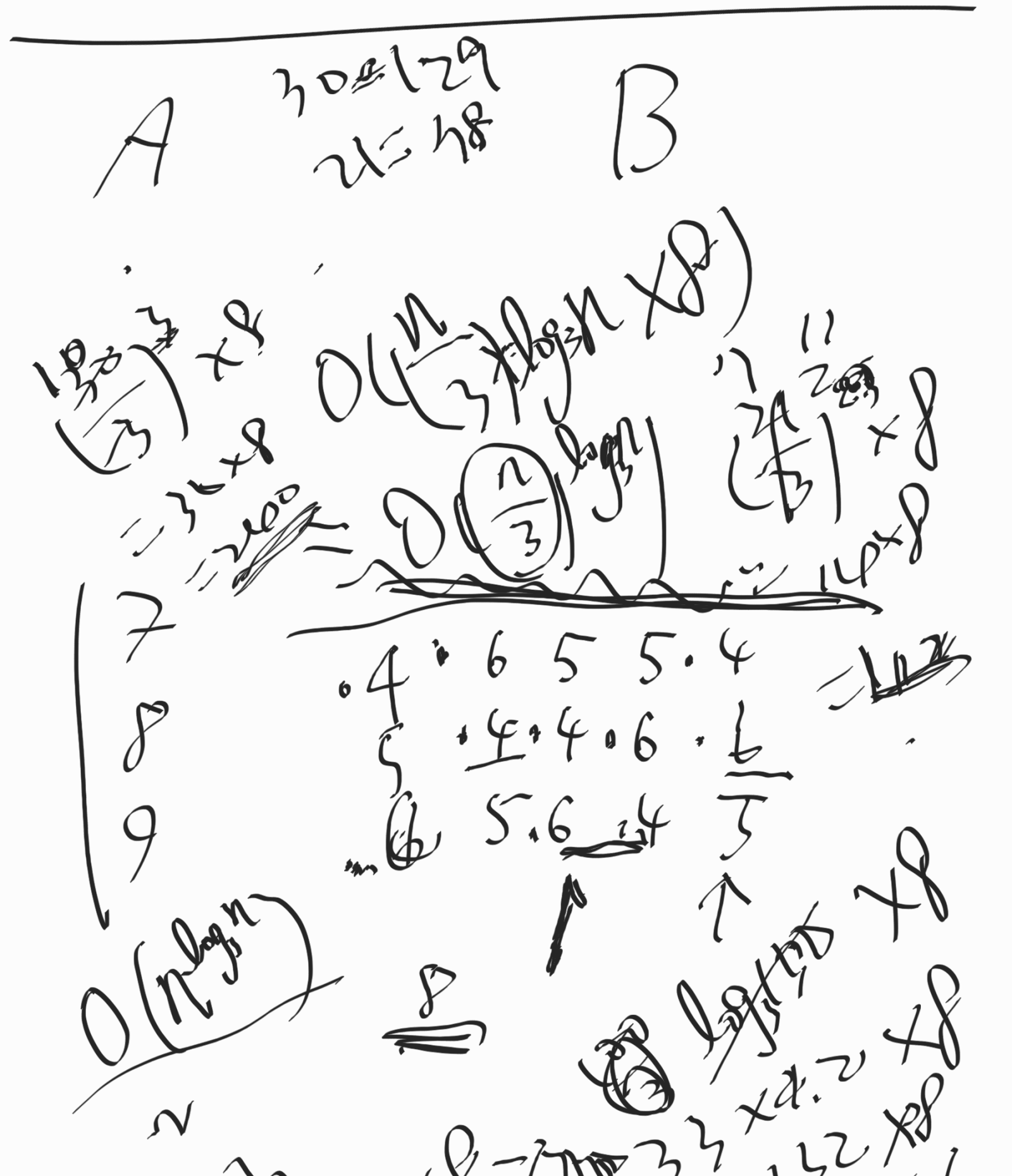


ROTATE:



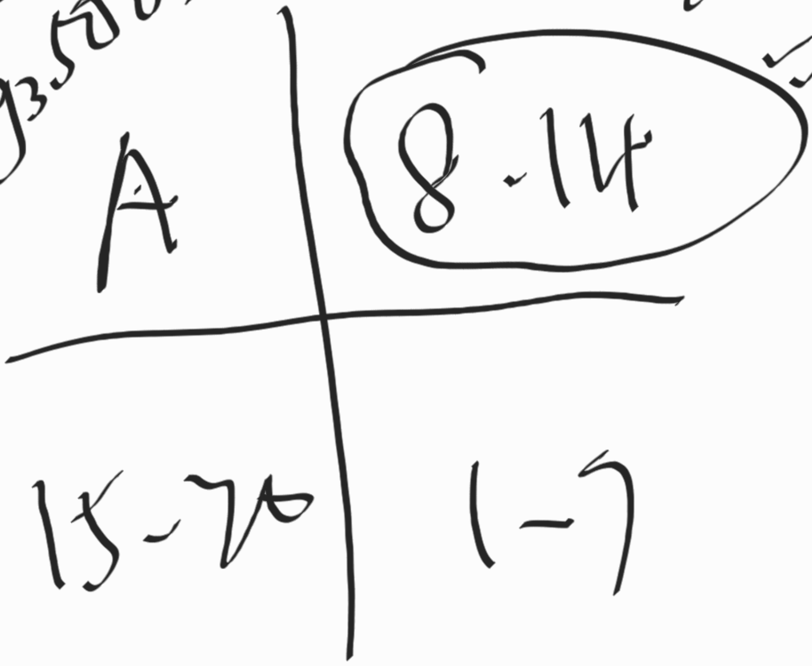
REV_ROTATE:



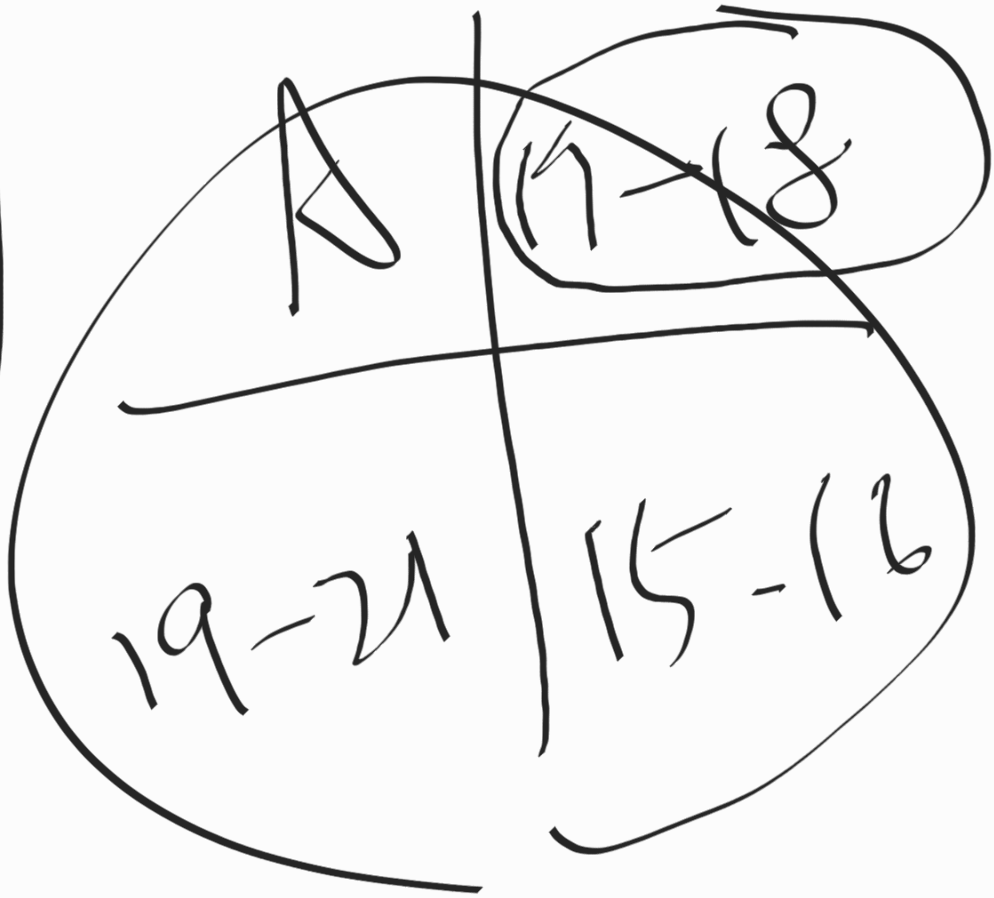


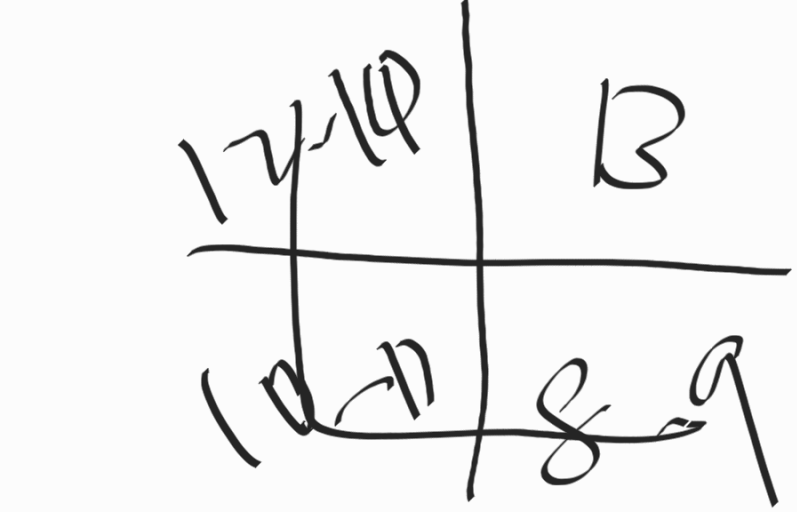
$\frac{500}{3} \times \log_3 500 \times 8$ = ... = 1056

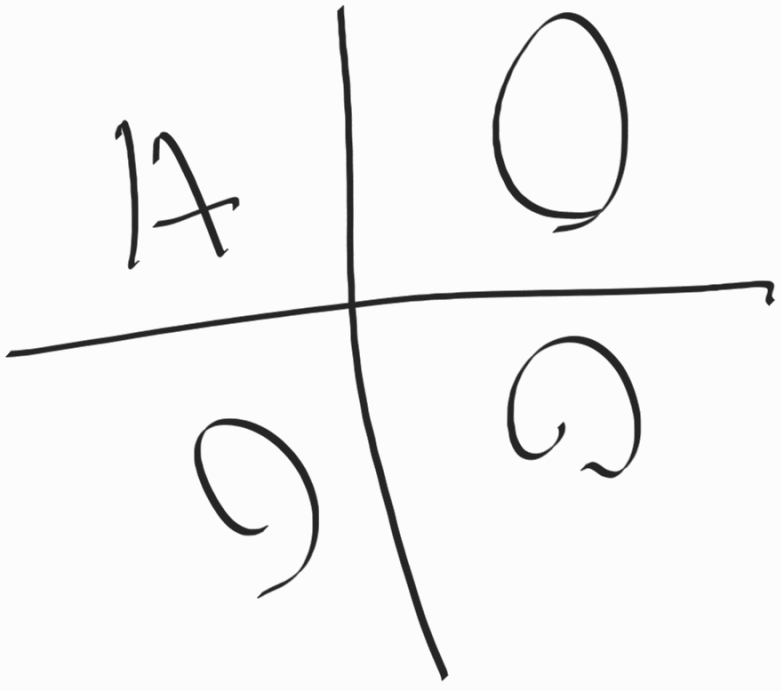
= 7

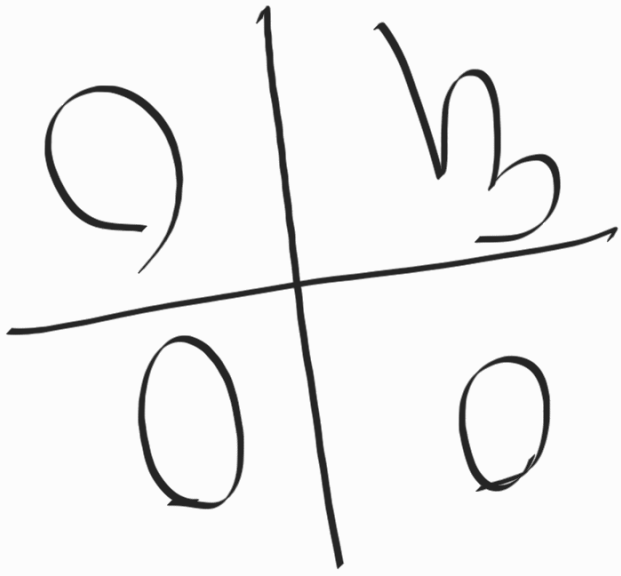


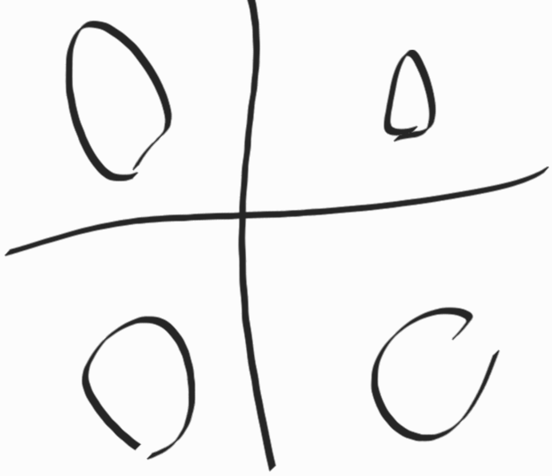
Null = ((void*)0)

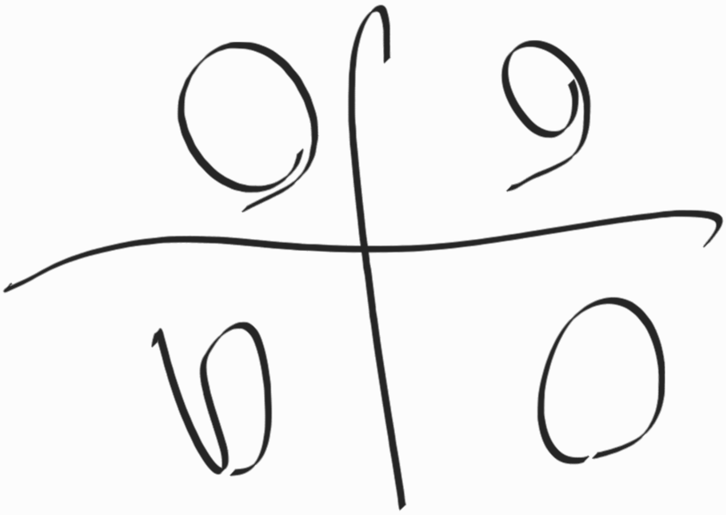












889

223 333

223

6 4 4 4

4	4			
5	5	5	6	6
6	6	6	5	5
4	5	5	6	6
6	6	4	4	5
5	4	6	5	4
			↑	↑

---

30 numbers:

1-30

A	11-20
21-30	1-10

21-30

A	24-26
27-30	21-23

27-30

A	28
29-30	27

11-20

17-20	B
14-16	11-13

17-20

A	19-20
18	17

1-10	7-10

7-10	4-6
1-3	B

A	9-10
8	7

---

want	have	LAST TARGET	MAKEGOALS
x	x	debug	debug
✓	✓	debug	–
(✓)	x	–	debug
x	x	–	–

24

A	8
8	8

need to remember blocksize for longer! otherwise the separation is not correct – or I need to know the pos_vorher all the time anew and adjusted locally

1 2 3 4 5

Ca i

1 2 3 4 5

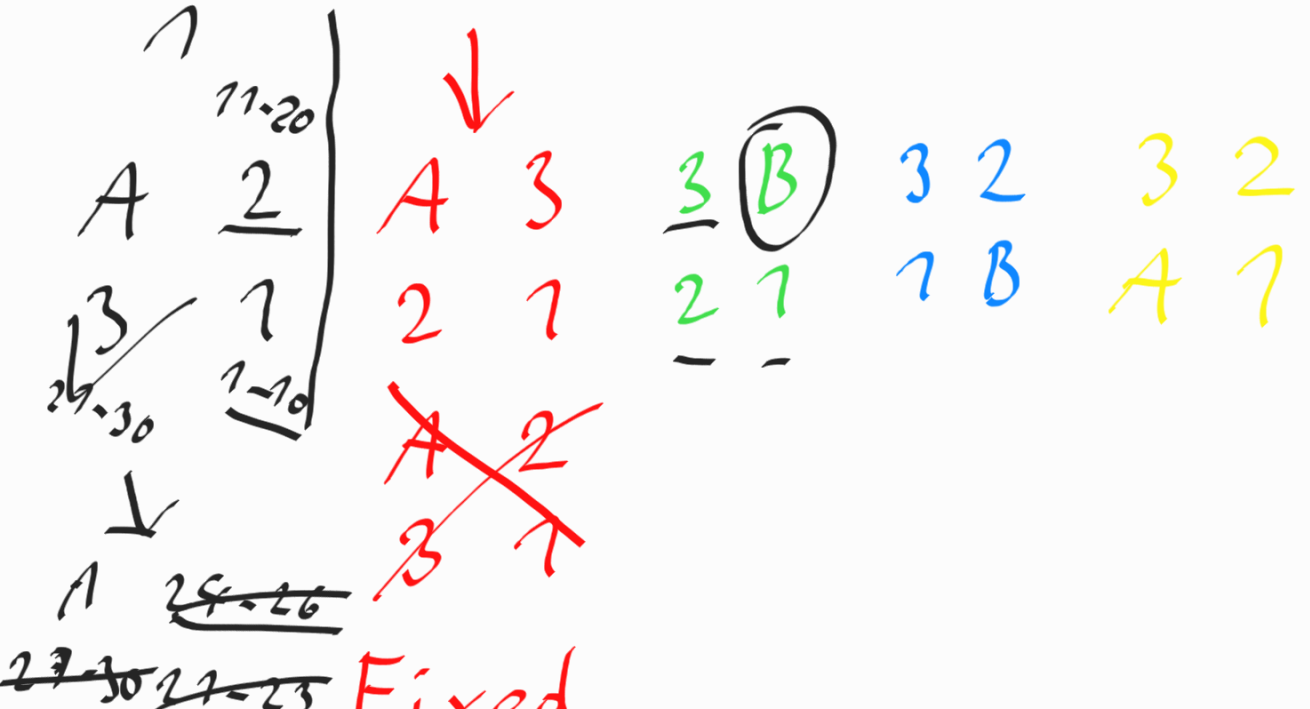
_ _ _ _ _

1 2 3 4 5

index 0 1 2 3 4

i++ --(len-1)

I did this bc it's easier to sort



Optimization for small amount

A	B	A	B	A	B	A	B	A	B	A	B	A	B
1												1	
5		5		2						2		2	
2		2		4		4		3		3		3	
4		4		3		3	2	4	2	4		4	
3		3	1	5	1	5	1	5	1	5	1	5	

1. pb → 2. ra → 3. pb → 4. sa → 5. pa → 6. pa

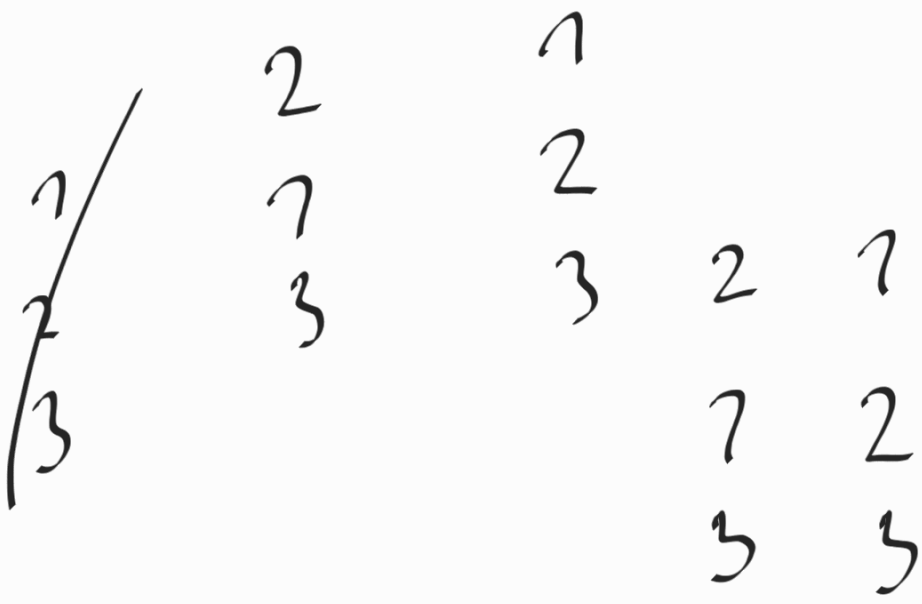
1		2	1	4
5		4	2	3
2		1	3	2
4		5	4	5
3		3	5	1
~~IIII~~	II	~~IIII~~	~~IIII~~	↑
				9!=7

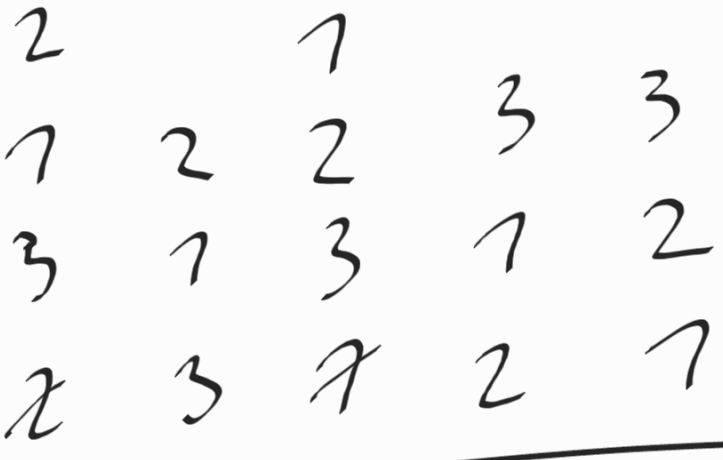
5 2 4

4 1 5

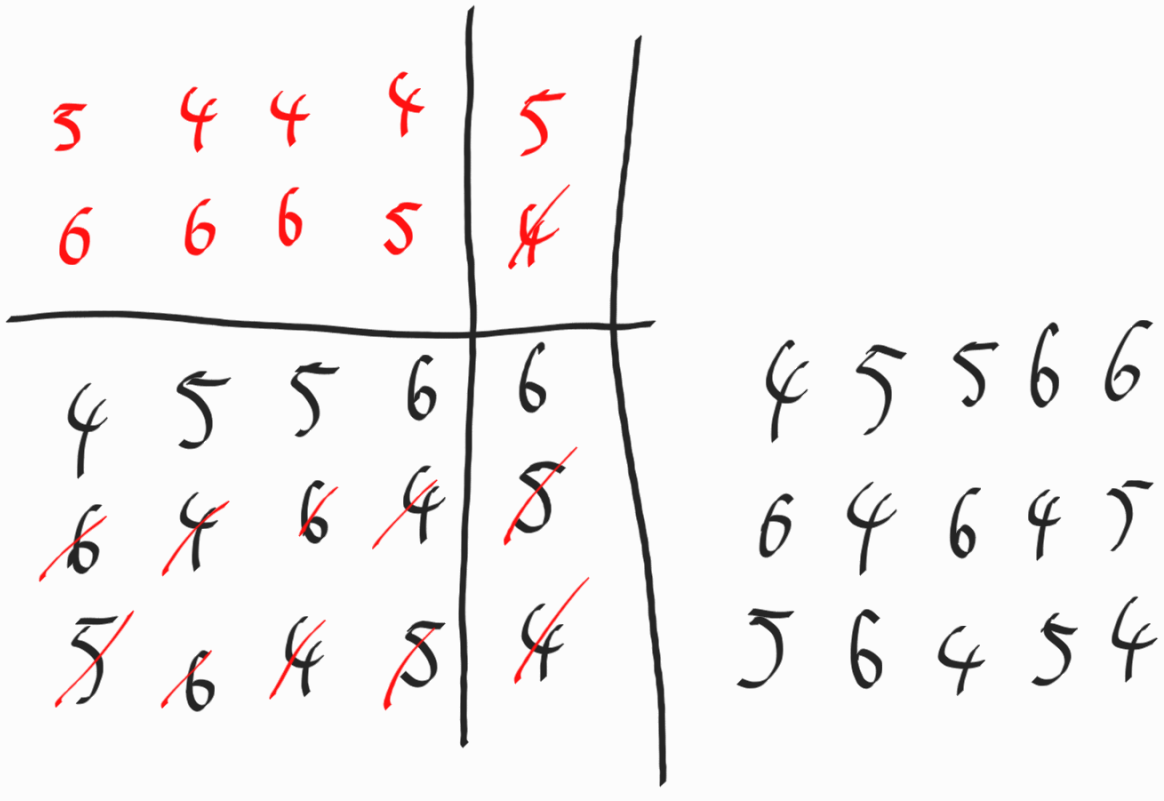
3 3

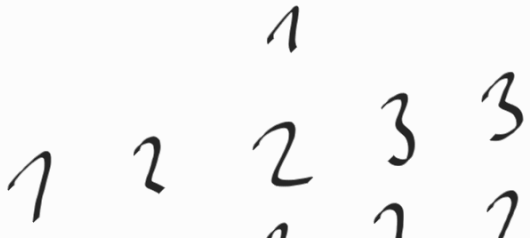
pa, ra, pb, ra, ra






5 5 4
5 4 4 5
5 5 4
5 4 4 5
4 5 5 6
6 4 6 4
5 6 4 5
4 5 5 6
6 4 6 4
5 6 4 5
2 3 3 3

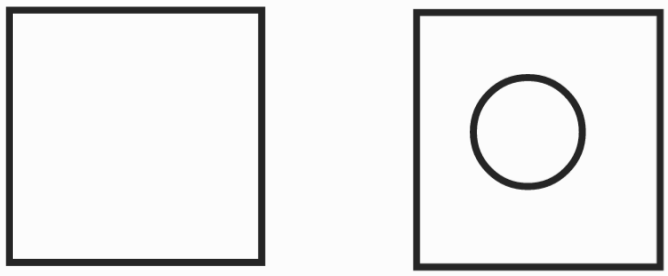

5 4 4 4
6 6 6 5
5
4
4 5 5 6
6 4 6 4
5 6 4 5
6
5
4
4 5 5 6 6
6 4 6 4 5
5 6 4 5 4

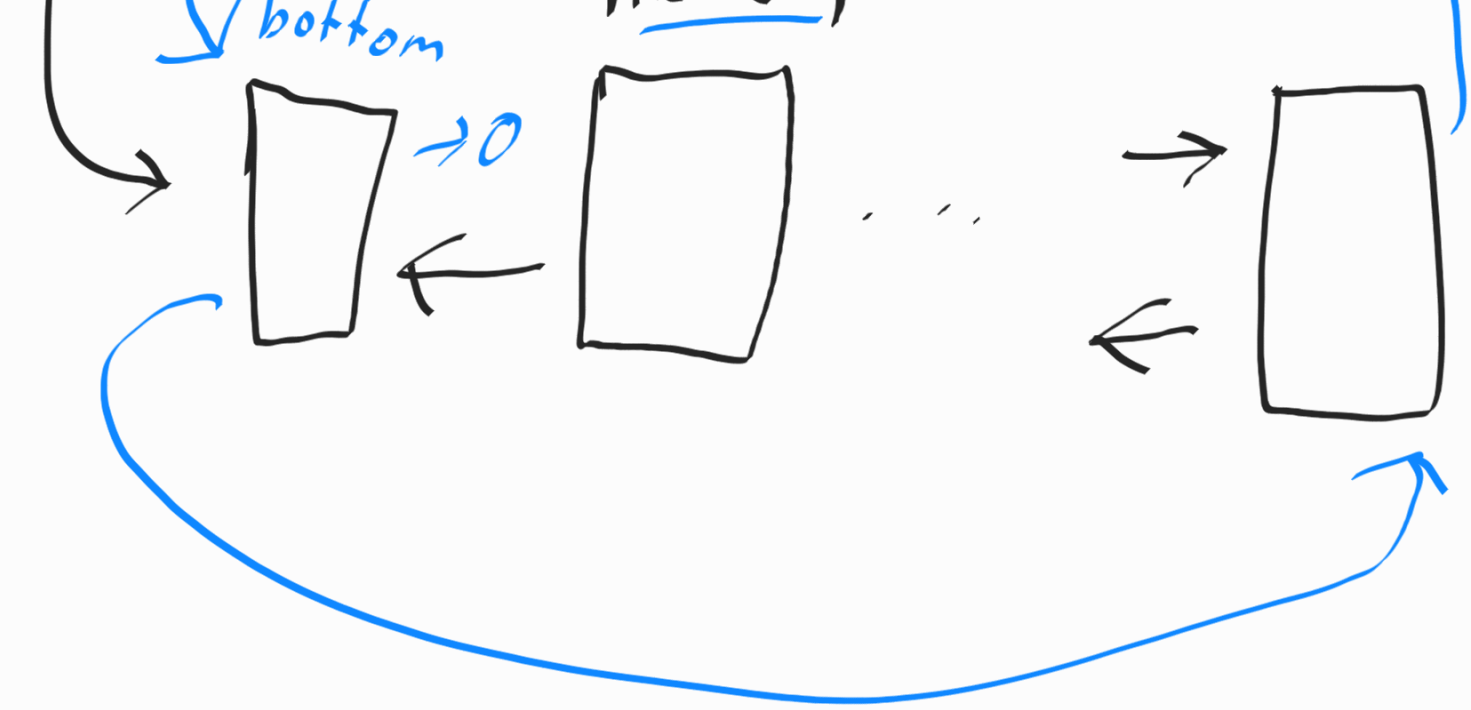

1
1 2 2 3 3

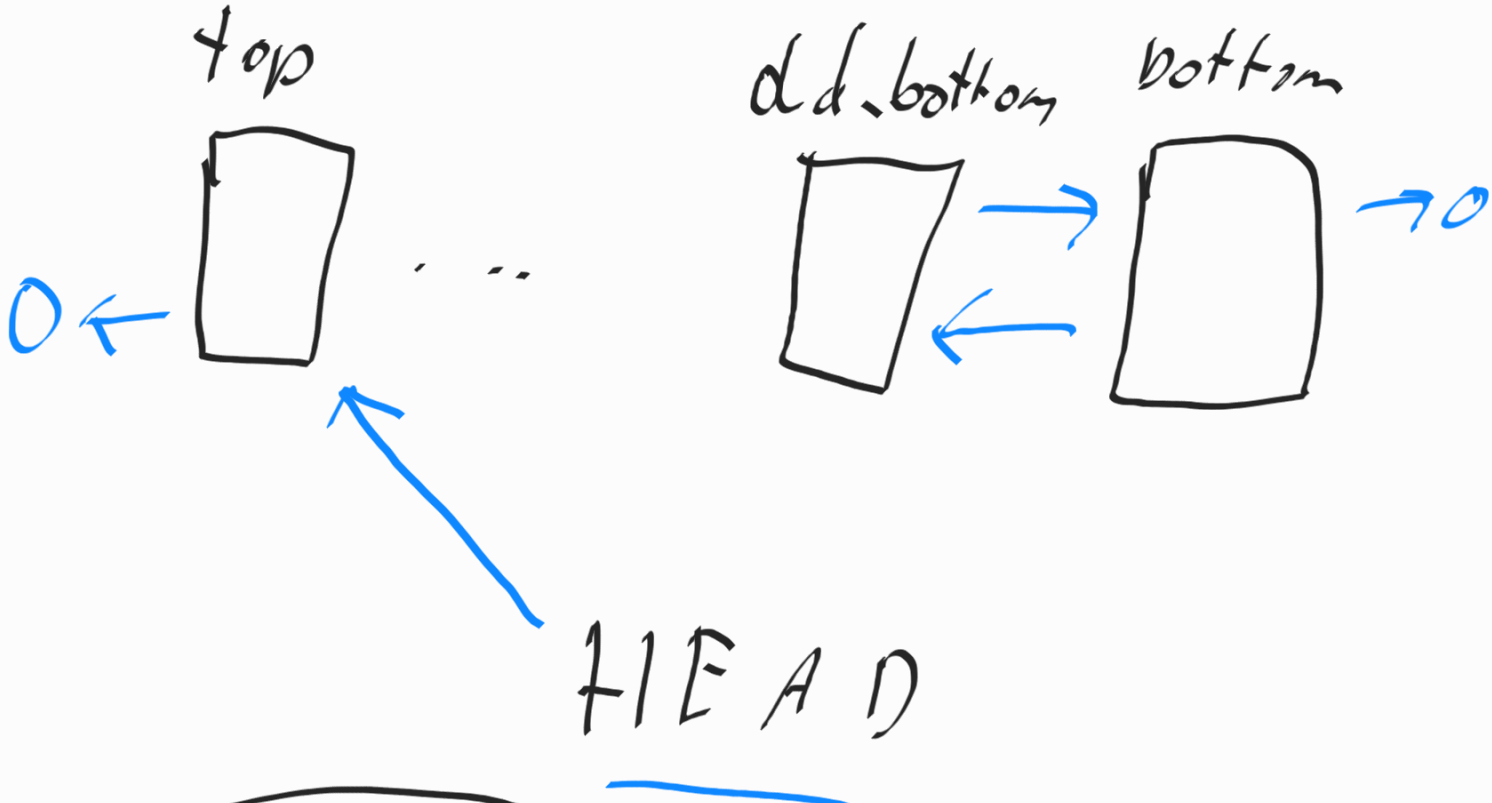
3 1 3 1 2

2 3 1 2 1

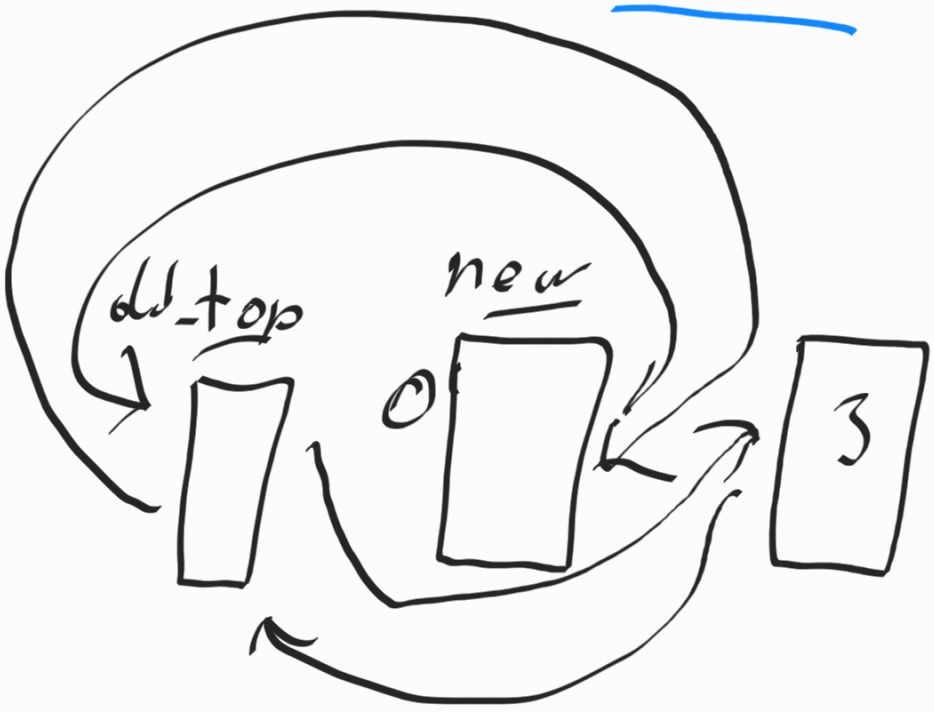
EVALUATIONS

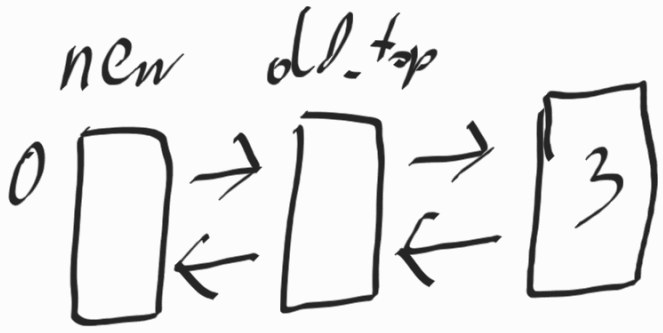






HEAD




new
old_tap
0
3