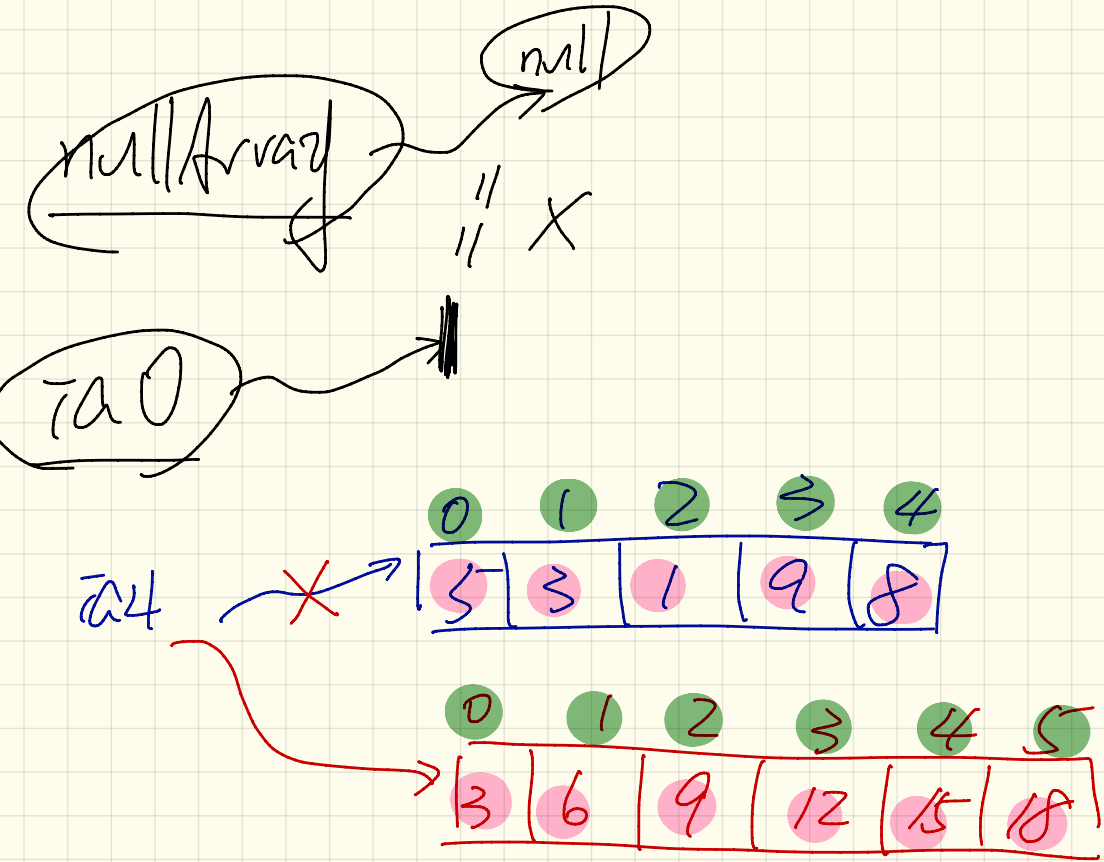
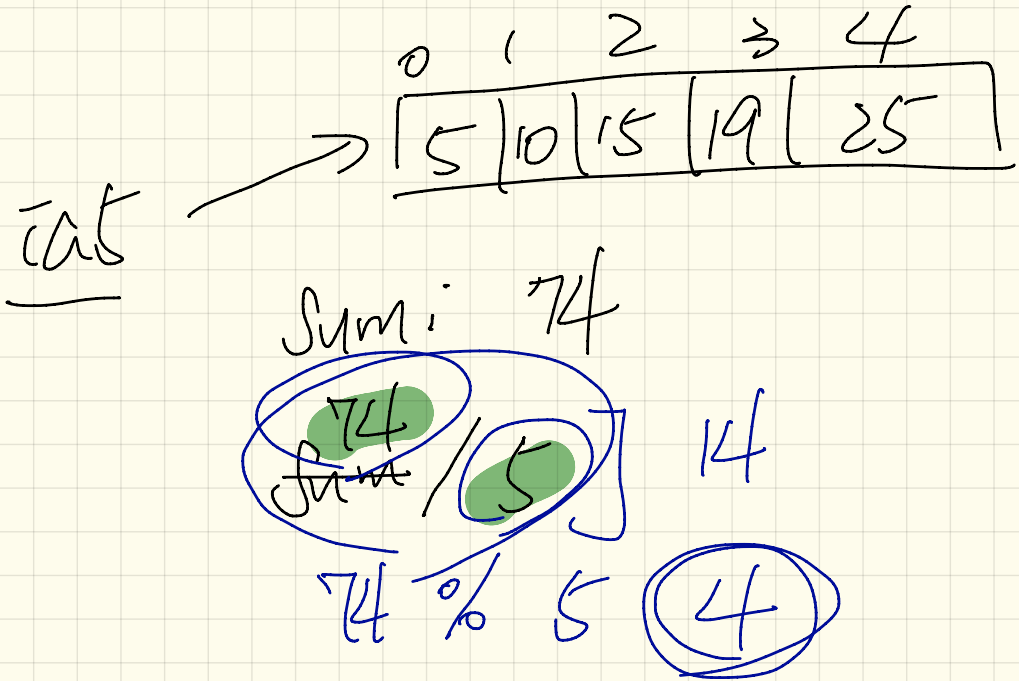
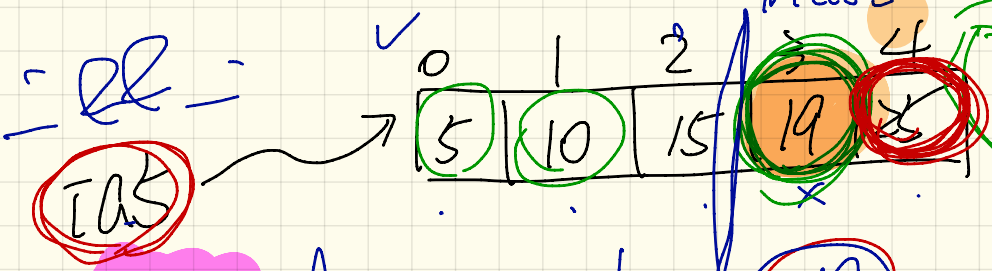
Solution to Example Test 2: Arrays and Loops


null
null/Array
X
TA0
0 1 2 3 4
5 3 1 9 8
TA4
0 1 2 3 4 5
3 6 9 12 15 18


0 1 2 3 4
5 10 15 19 25
cat
Sum: 74
74
Sum
/ 5
14
74 % 5
4

19 % 5 == 4

- int[] ea = {};
- int[] ea = new int[0];



witness of property violation : 19

allMultiplesOf5(as); False

True False T.

&& as[0] % 5 == 0

&& as[1] % 5 == 0

: as[3] % 5 == 0

&& as[4] % 5 == 0

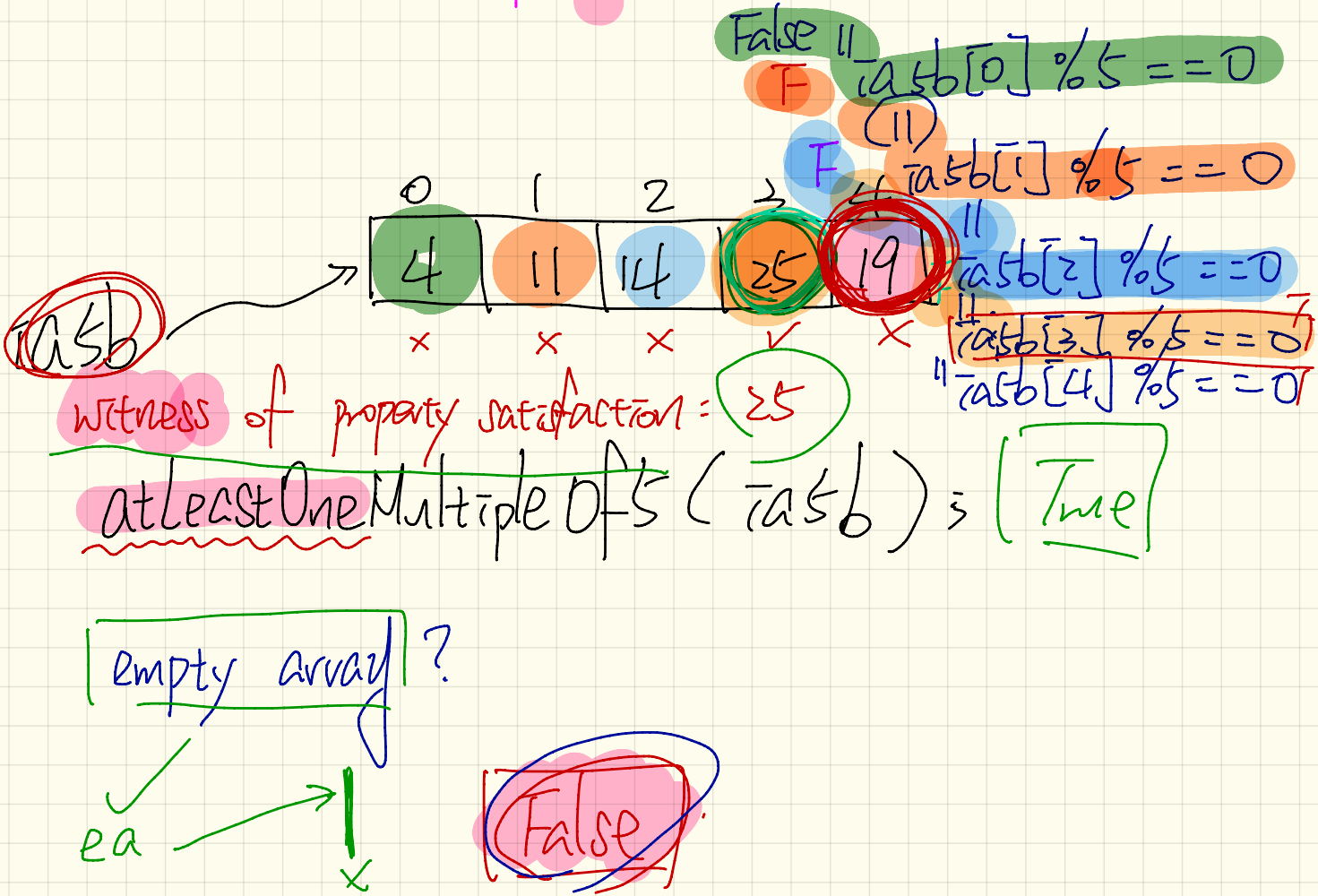
F

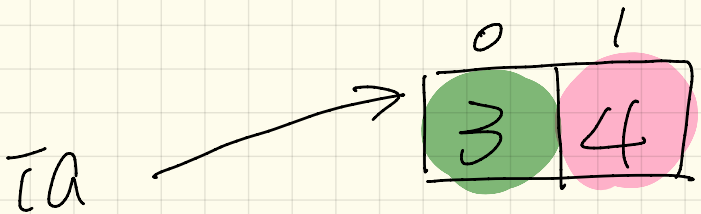
empty array?

ea

TRUE

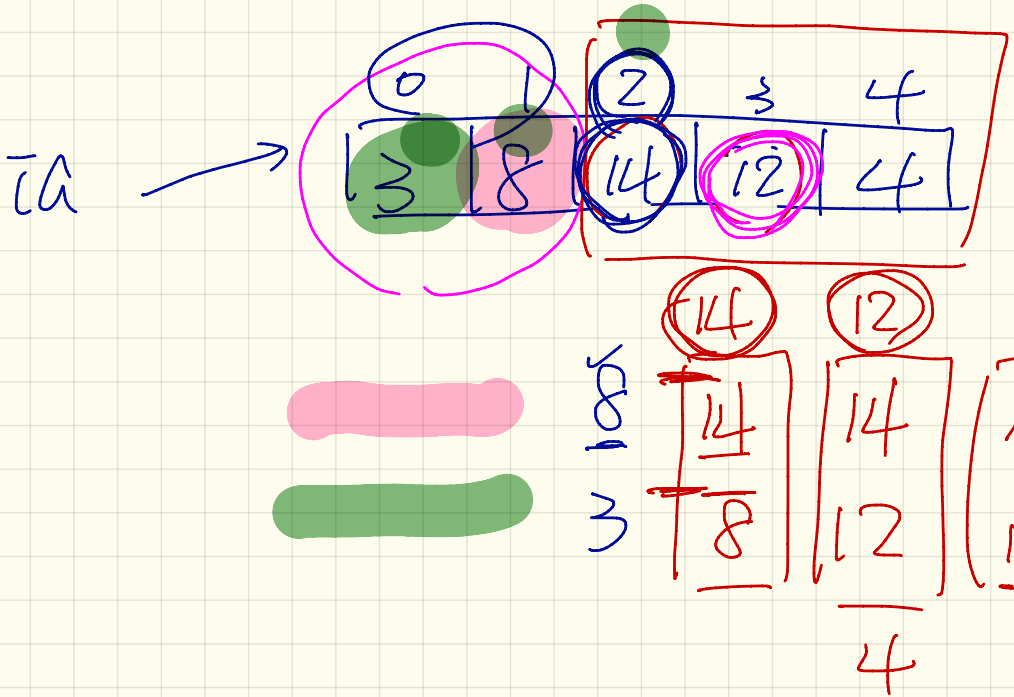
As long as we can find a violation witness in ea, then it's False; otherwise, True.

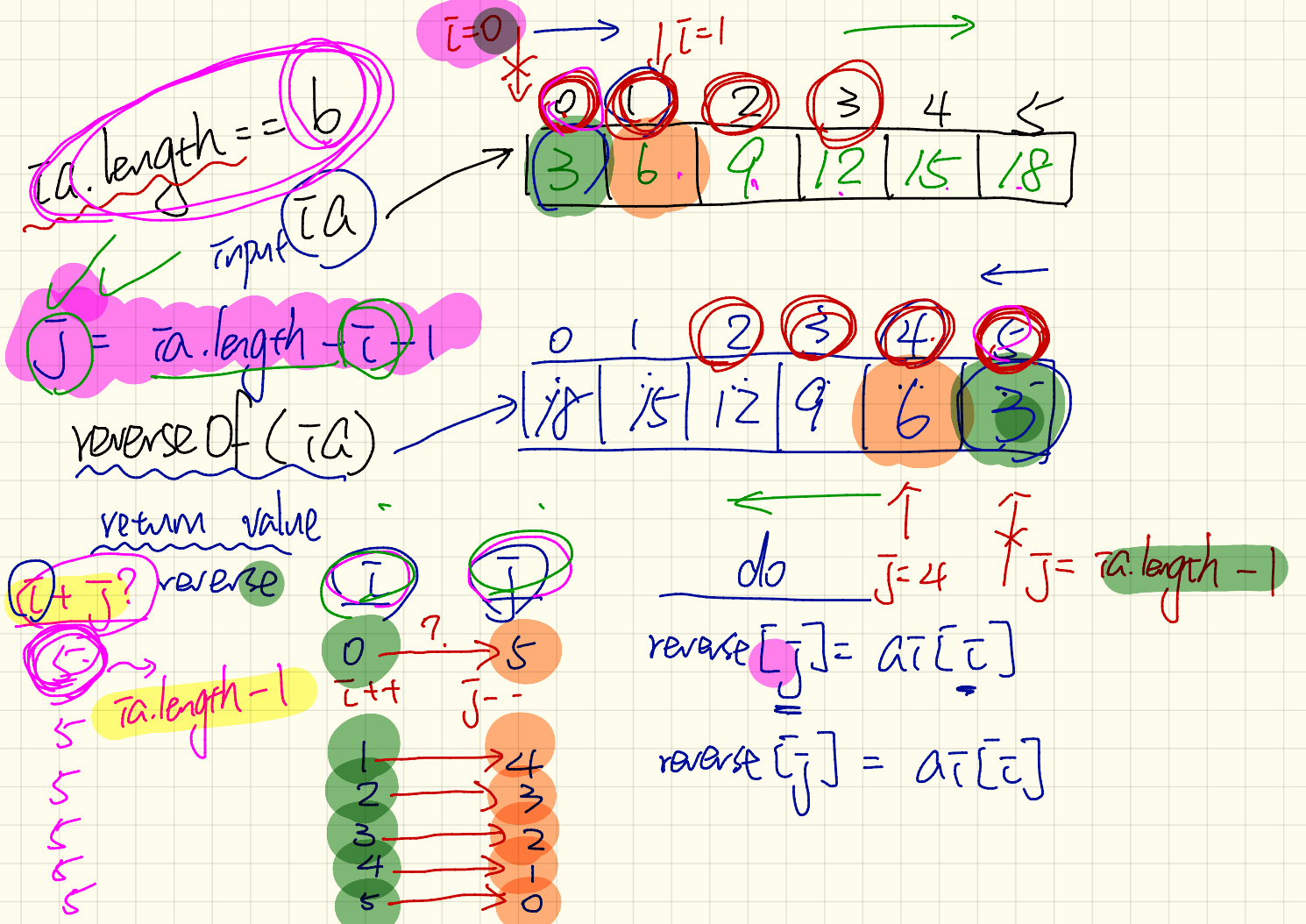

False ||
F
arr5b[0] %5 == 0
||
F
arr5b[1] %5 == 0
||
arr5b[2] %5 == 0
||
arr5b[3] %5 == 0
||
arr5b[4] %5 == 0
0
1
2
3
4
4
11
14
25
19
arr5b
x
x
x
✓
x
witness of property satisfaction : 25
atLeastOneMultipleOf5(arr5b) : True
empty array ?
ea
x
False

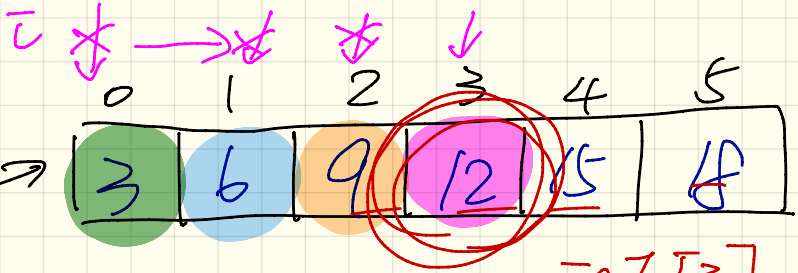


Exercise:

3rd maximum




i=0
i=1
0 1 2 3 4 5
3 6 9 12 15 18
ia.length == b
input ia
j = ia.length - i - 1
reverse of (ia)
0 1 2 3 4 5
j4 j5 12 9 6 3
return value
reverse
i + j ?
i
ia.length - 1
5
5
5
5
5
5
i
j
0 ? 5
i++
j--
1 4
2 3
3 2
4 1
5 0
do
j=4
j = ia.length - 1
reverse[j] = ai[i]
reverse[j] = ai[i]

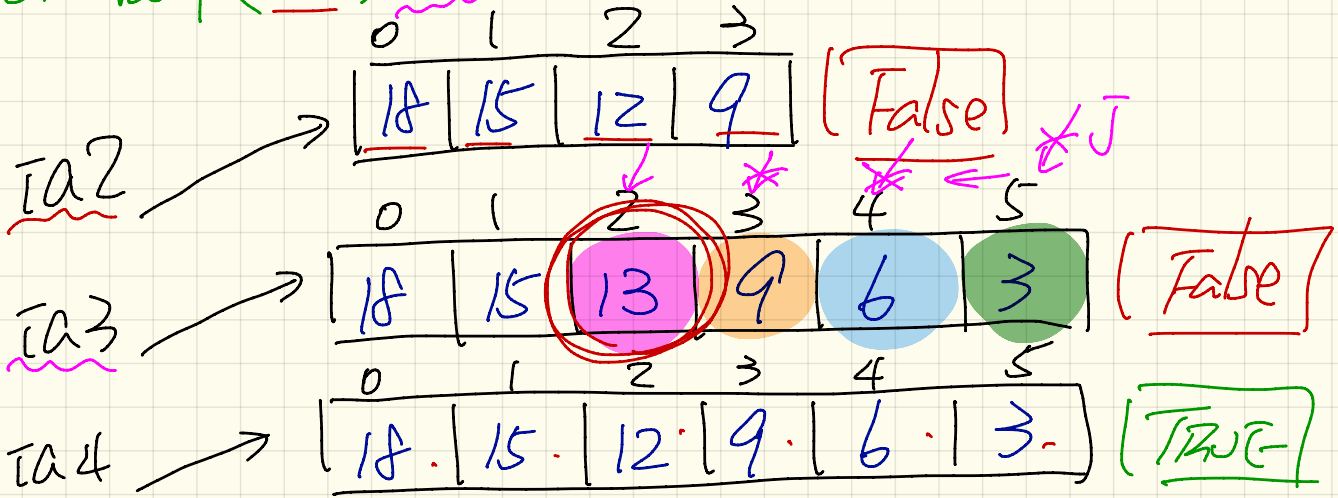


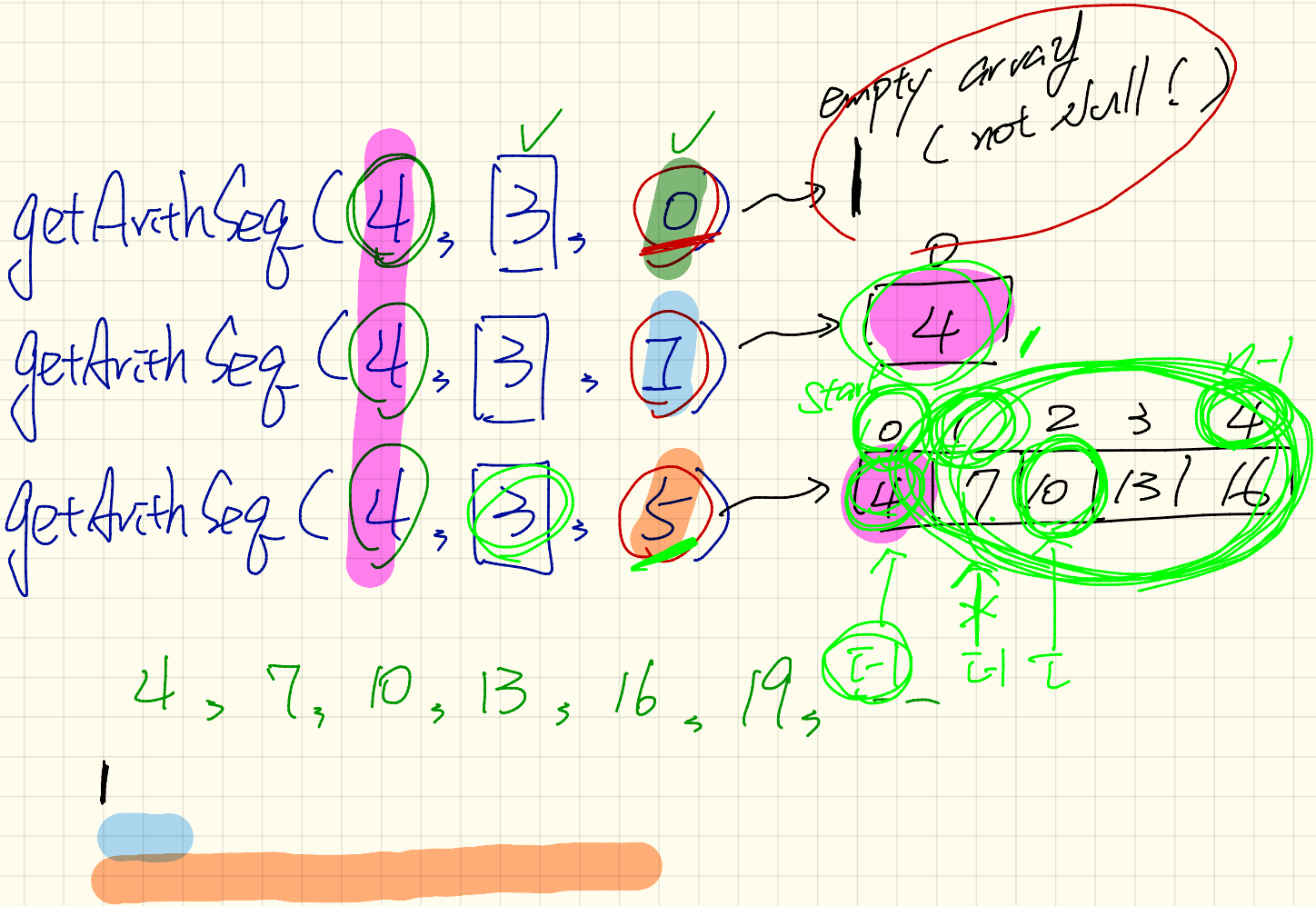
a1[3] and a3[2] are a violation witness.

isReverseOf(a1, a2)

isReverseOf(a1, a3)

isReverseOf((a1), (a4))




getArithSeq(4, 3, 0)
getArithSeq(4, 3, 1)
getArithSeq(4, 3, 5)
empty array
(not null!)
4
start
0
1
2
3
4
4
7
10
13
16
4, 7, 10, 13, 16, 19,

isArithSeq(a1)

a2

a3

a4

a1 → |

True

a2 → 0 | 4 |

True

a3 → 0 | 4 | 7 | 1

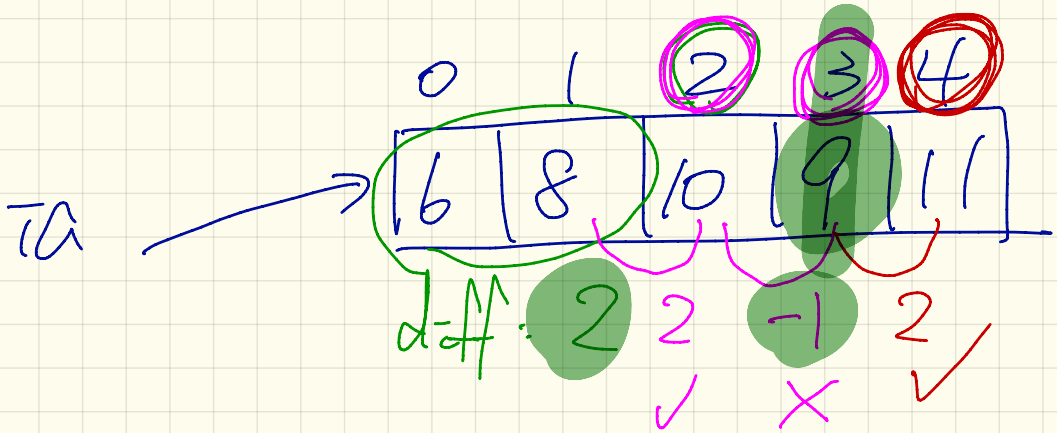
True

a4 → 0 | 4 | 7 | 1 | 10 | 2

True

a5 → 0 | 4 | 3 | 1 | 7 | 2 | 4 | 1 | 2

False

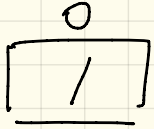

0
1
2
3
4
16
8
10
9
11
diff
2
2
-1
2
✓
X
✓

Fibonacci Sequence

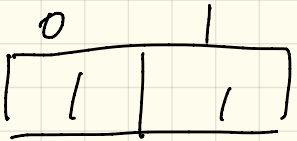
1, 1, 2, 3, 5, 8, 13, 21, ??
34

getFibSeq(0) → 1

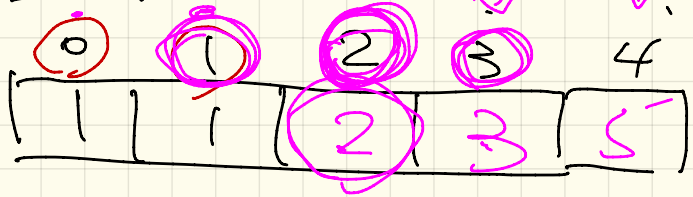
getFibSeq(1) →



getFibSeq(2) →



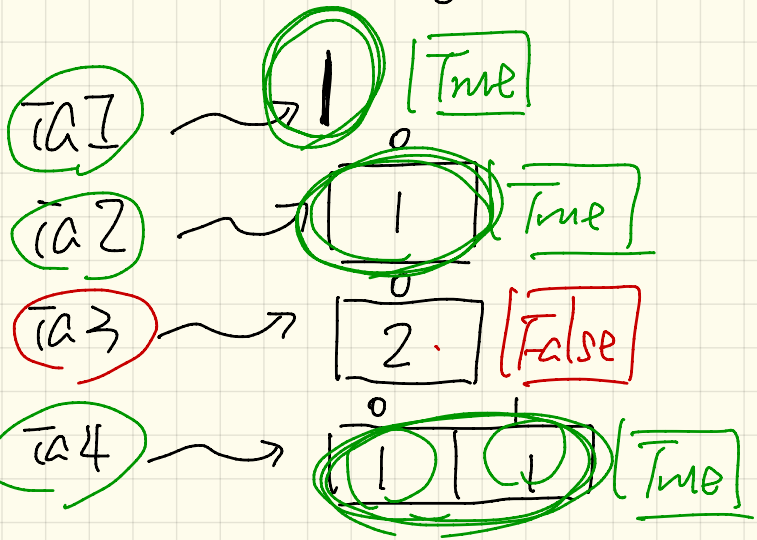
getFibSeq(5) →

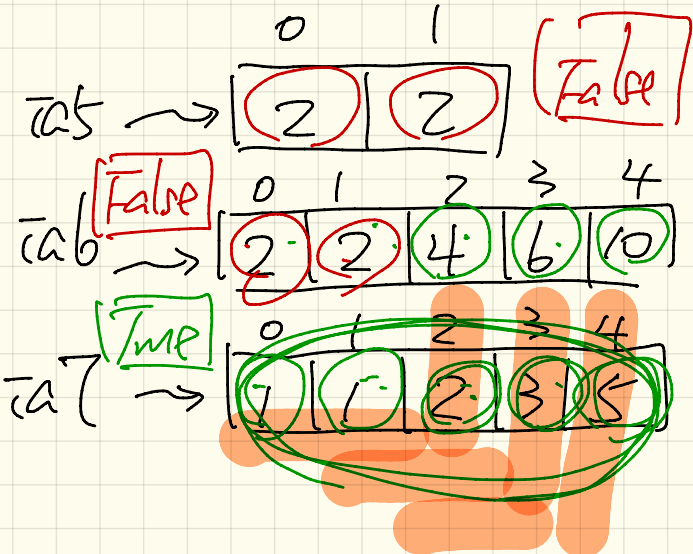


Fibonacci Sequence

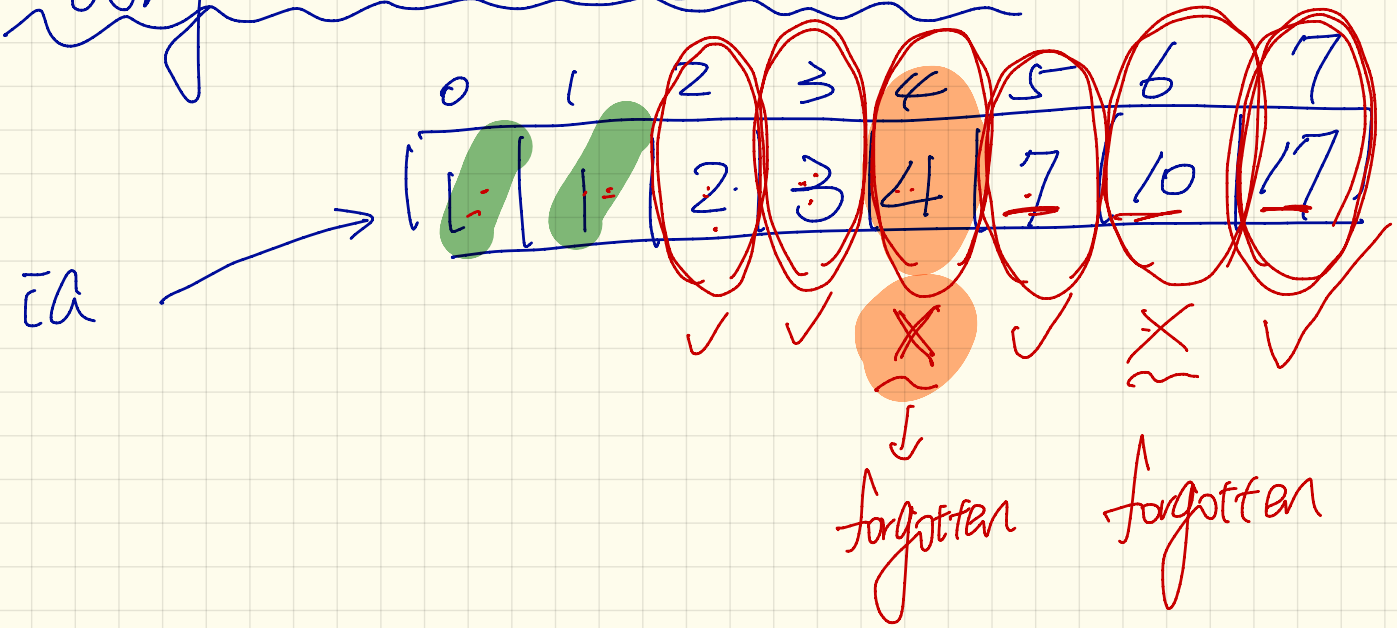
1, 1, 2, 3, 5, 8, 13, ...

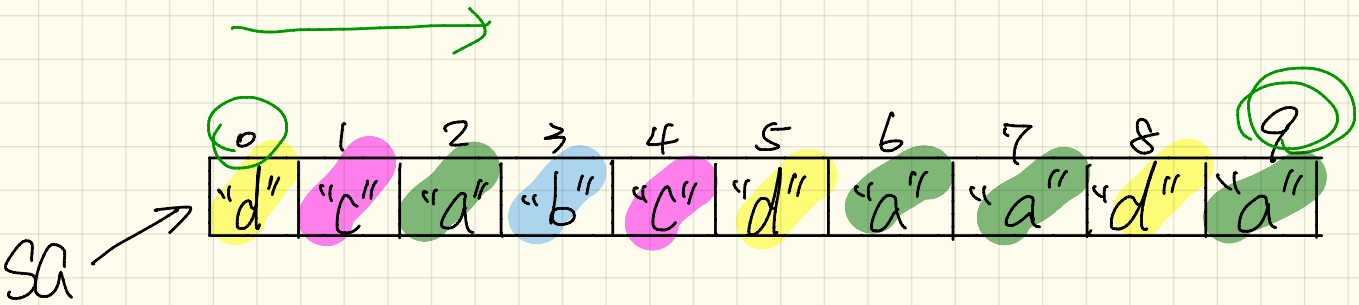
isFibSeq(__)





Why Version 2 is incorrect





numberOfOccurrences(sa , "a") 4

"b" 1

"c" 2

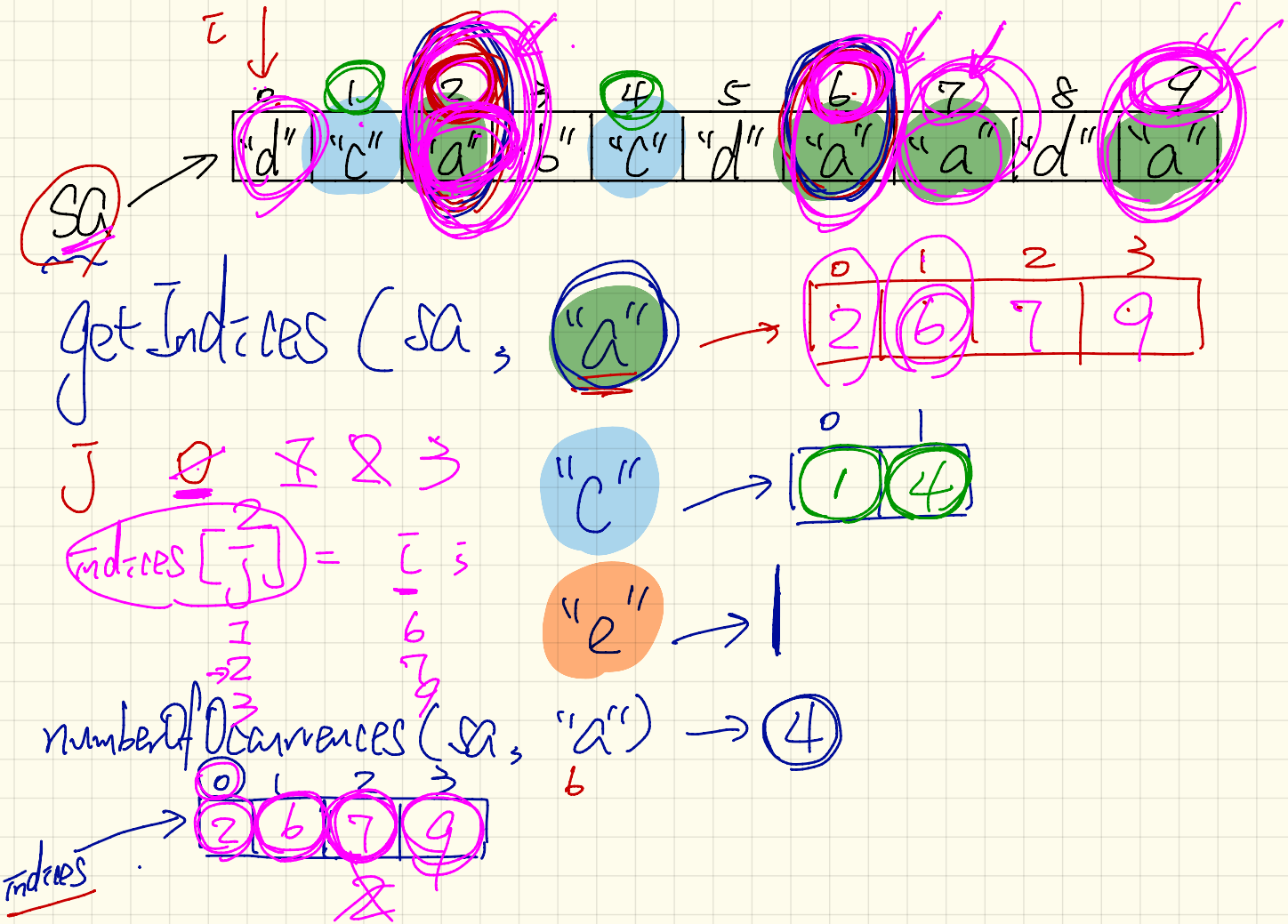
"d" 3

"e" 0

ea → 1

(ea , "a") 0

"b" 0


i
0 1 2 3 4 5 6 7 8 9
"d" "c" "a" "b" "c" "d" "a" "a" "d" "a"
sa
getIndices(sa, "a") → 0 1 2 3 / 2 6 7 9
j 0 1 2 3
indices[j] = i's
1 → 2
6 → 7
9 → 3
"c" → 0 1 / 1 4
"e" → |
numberOfOccurrences(sa, "a") → 4
indices
0 1 2 3 / 2 6 7 9
2