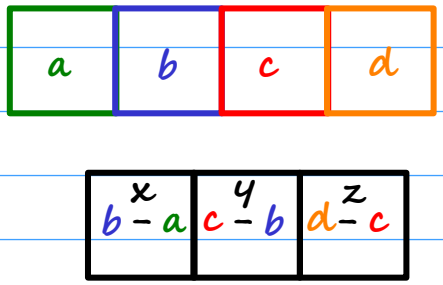
Before the whole O(N) shenanigans...

we are going to do some dumb things with differences...

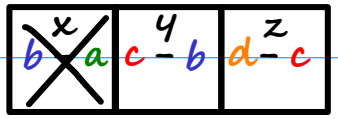
1) Transform into differences (b - a OR a - b does not matter...)



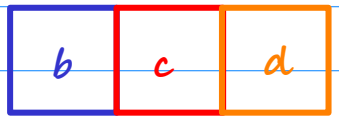
2) How to delete numbers if we have only differences?

2A) How to delete the first number?

Drop a

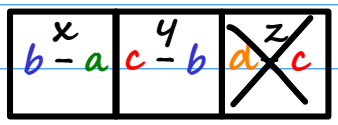


y : z : [] aka:

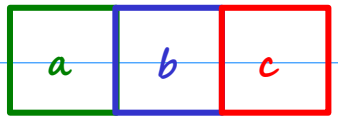


2B) How to delete the last number?

Drop d

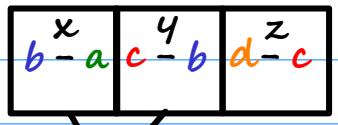


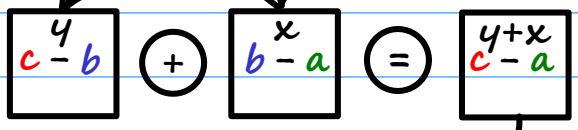
x : y : [] aka:

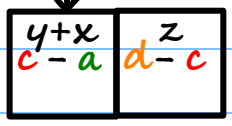


2C) How to delete the other numbers?

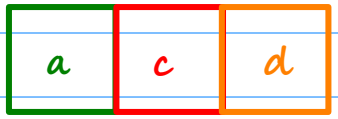
Drop b



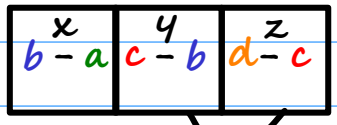


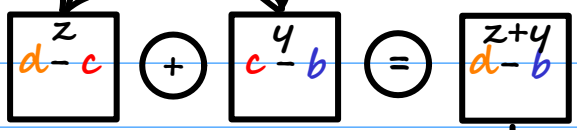


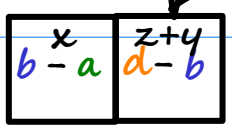
(y + x) : z : [] aka:



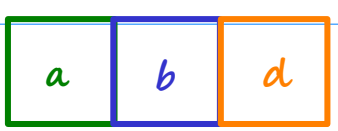
Drop c







x : (z + y) : [] aka:



So... O(N * N) is slow, we want the juicy O(N)...

Let's see the Maneater variant

1) How to represent state?

([Int], Int, [Int])

all OK differences

can be a Maybe

we need just 1 look-back

current difference

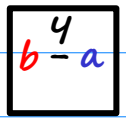
remaining differences

2) Cases

2A) We are at the last one

([], y, [])

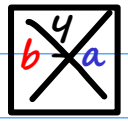
Let's say y is ok -> we can just keep (or drop last) -> True



Let's say y is bad

We can drop a -> True

We can drop b -> True

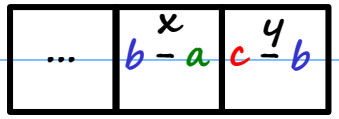


2B) We are at the last one

(x:_, y, [])

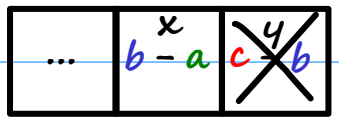
Let's say y is ok

we can just keep it -> True



Let's say y is bad

We can drop c -> True



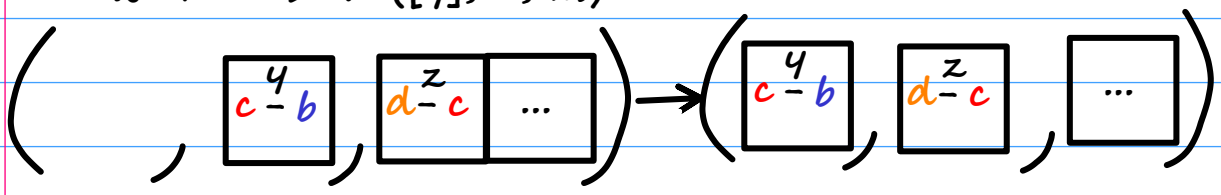
Maneater continued...

2C) We are at the first one

([], y, z:zs)

Let's say y is ok

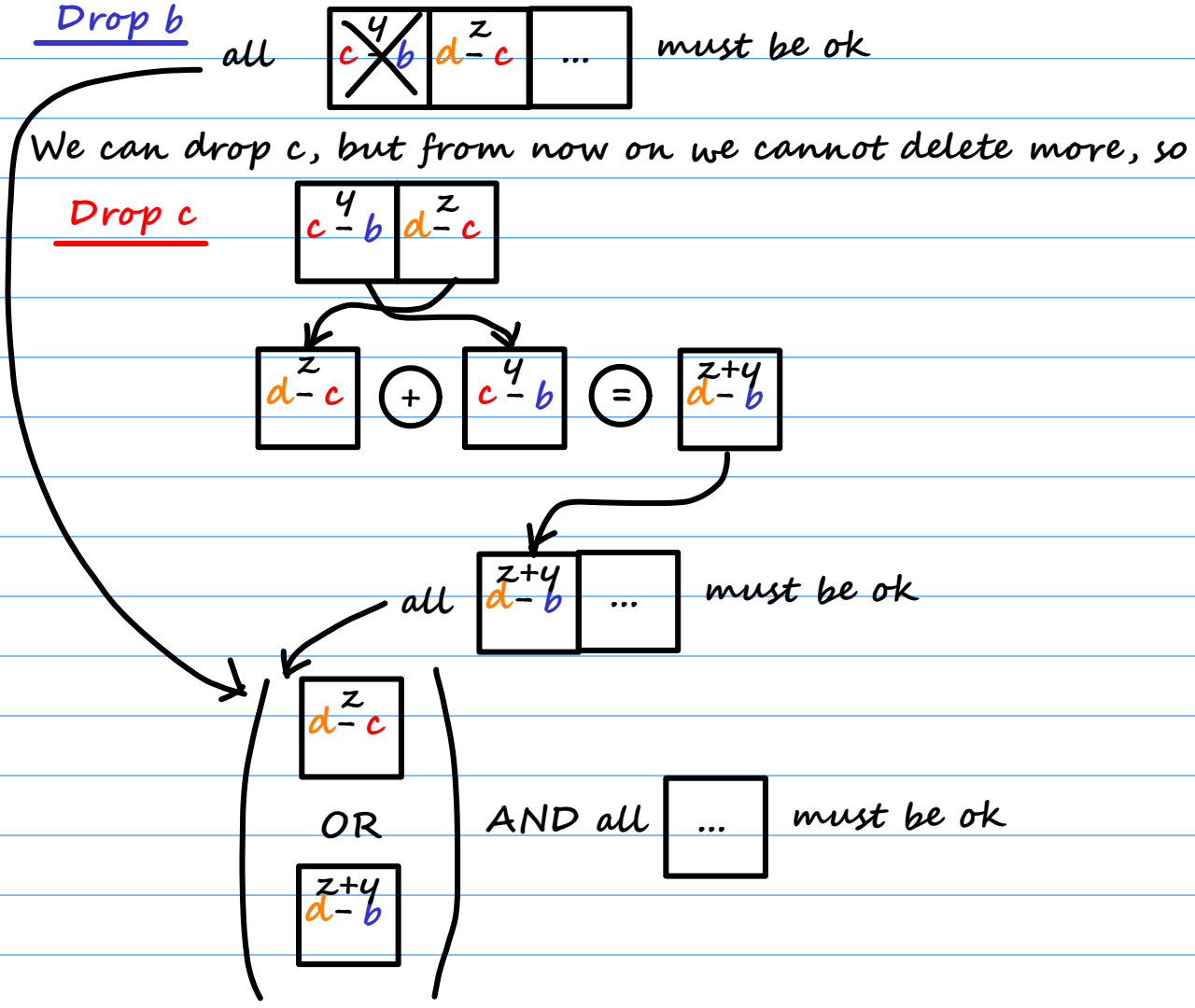
we recurse to ([y], z, zs)



Let's say y is bad

We can drop b, but from now on we cannot delete more, so

Drop b



Maneater continued...

2D) We are at an inside one

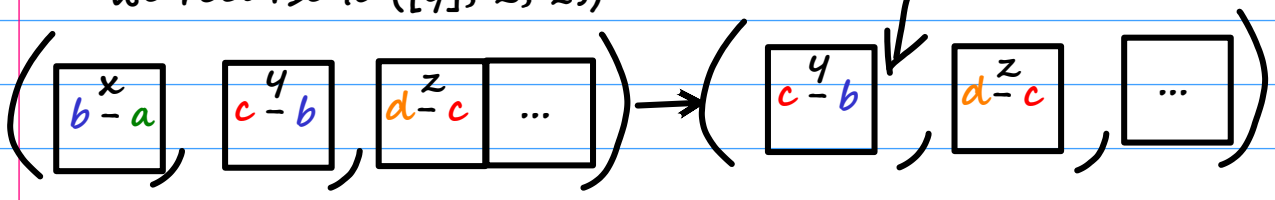
(x:_, y, z:zs)

You can forget 'x'

it will never be used again

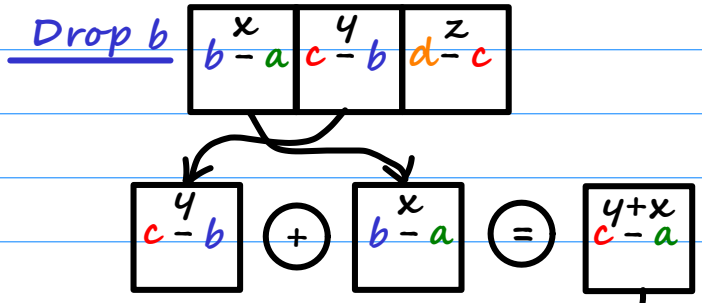
Let's say y is ok

we recurse to ([y], z, zs)

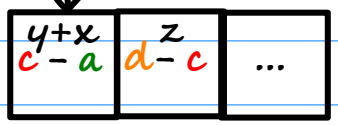


Let's say y is bad

We can drop b, but from now on we cannot delete more, so

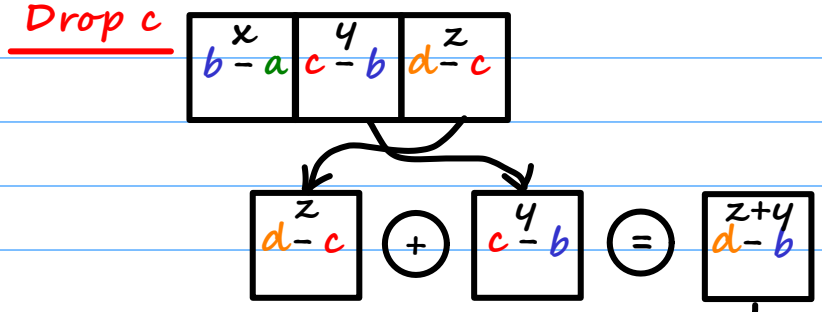


all

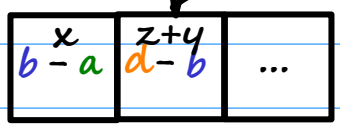


must be ok

We can drop c, but from now on we cannot delete more, so

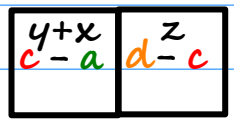


all



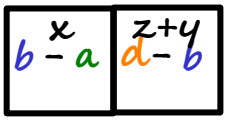
must be ok

all



OR

all



AND all

must be ok

Maneater Summary

3) Maneater works only with non-empty lists

So... nest Maneater inside a charade function

on empty list -> return True

else -> call Maneater

Summary:

We DO NOT care about anything else, we just want O(N) like

```
for (d : differences)
    if(!p(d)) return false
```

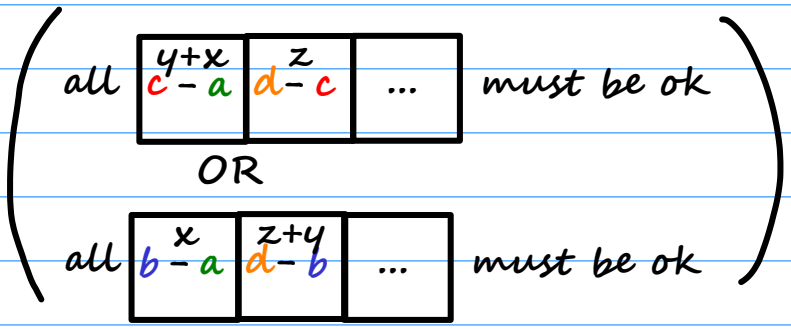
Unfortunately, we cannot do it immediately

So we play nice... bide our time...

BUT as soon as we 'use up' our delete choice

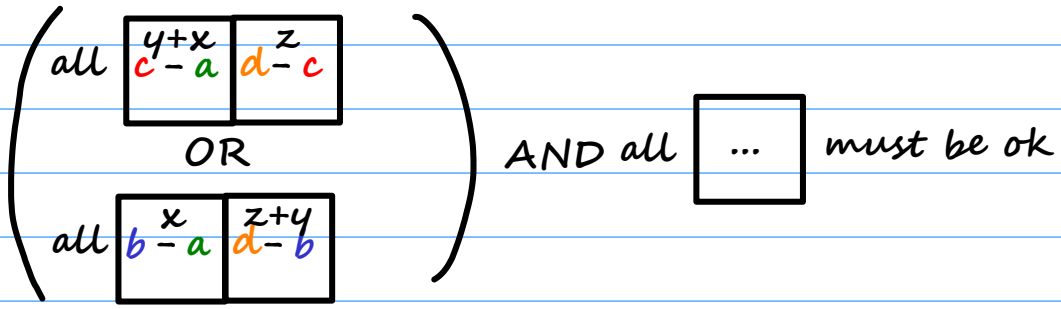
we show our true colors and TURN INTO A FOR-LOOP

But we must be smart... We don't want to do 2 loops, like:



So only if all preemptive checks are fine,

do we turn into a for-loop



Runtime characteristics

Below is a video detailing runtime characteristics:

https://youtu.be/KErUJUszUpO?t=122