Temporal Pincer - Eliminate KMP KMJ

In the temporal pincer solution, we left out one thing:

- Cleaning up the imperative array access code

We will revisit the temporal pincer, and remove random access.

Temporal Pincer Recap:

1) A list is good if any 'd' difference of consecutive elements is

   1 <= d <= 3

   Can we make a list 'good' by removing 1 element from it?

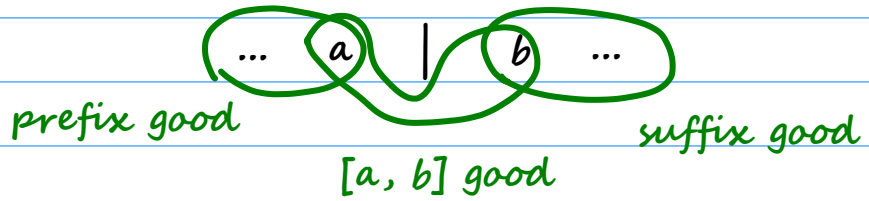
   Example: 1 2 2 3 4

2) Calculate prefixes and suffixes

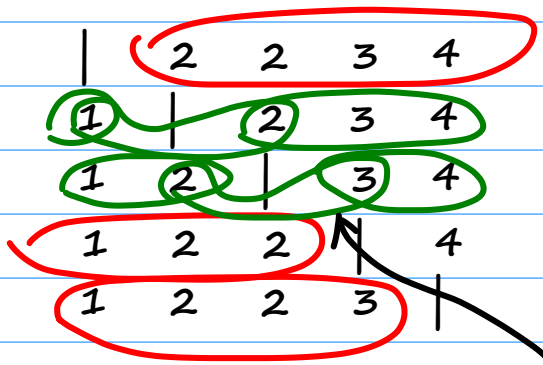
   Example:

|	2	2	3	4
1	|	2	3	4
1	2	|	3	4
1	2	2	|	4
1	2	2	3	|

3) Find at least one prefix suffix such that:



Example:



And, of course, compute:

- prefix list in O(N)
- suffix list in O(N)

The original temporal pincer did the merge by array accessing...

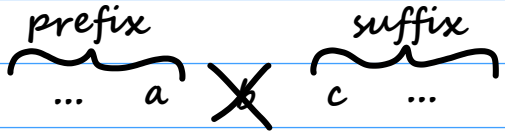
but we DO NOT want to do that in Haskell...

Temporal Pincer - Eliminate KMP KMJ

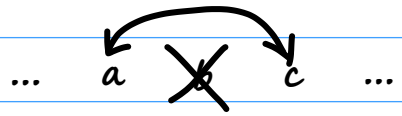
We want to get rid of C-style random array access.

So let's think: why are we doing it in the first place?!??!

After deleting 'b'



We want to 'merge' the prefix and suffix by checking 'a' 'c'



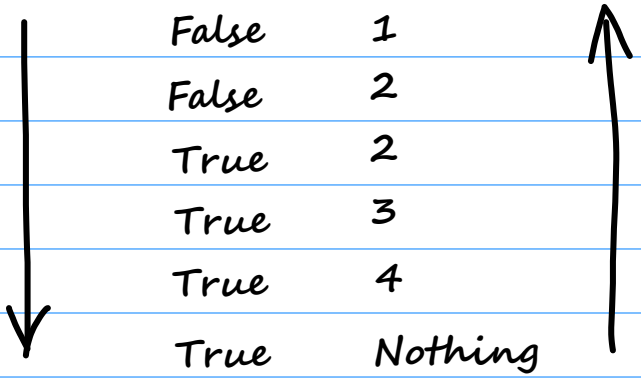
So, if we could produce:

- prefixes having 'a', and a boolean to tell if they are good
- suffixes having 'c', and a boolean to tell if they are good

Then we'd no longer need random access.

We will calculate prefixes/suffixes like this:

	Prefixes		Suffixes	
	is good?	end element	is good?	end element
#0	True	Nothing	False	1
#1	True	1	False	2
#2	True	2	True	2
#3	False	2	True	3
#4	False	3	True	4
#5	False	4	True	Nothing



So we have:

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

Temporal Pincer - Eliminate KM19 KMJ

When k = 0 (no removal, aka Part A) we just zip these together

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

Search for candidates where both prefix and suffix are (T, ...)

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

And check whether for the (_, a) (_, b) pair, 1 <= b - a <= 3 holds

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

1 <= 2 - 2 <= 3

In this case, this list cannot be made good by

removing k = 0 consecutive elements

(aka, no removal is allowed, i.e. Part A of the problem)

When k = 1 (one removal, aka Part B) we shift the suffixes left

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -) ← shift by 1

Search for candidates where both prefix and suffix are (T, ...)

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

And check whether for the (_, a) (_, b) pair, 1 <= b - a <= 3 holds

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

1 <= 2 - 1 <= 3 1 <= 3 - 2 <= 3

In this case, this list can be made good in 2 ways by

removing k = 1 consecutive elements

(aka, one removal is allowed, i.e. Part B of the problem)

Temporal Pincer - Eliminate KMI9 KMJ

The problem does NOT ask for it but let's see k = 2

When k = 2 (two removal) we shift the suffixes left

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -) ← shift by 2

Search for candidates where both prefix and suffix are (T, ...)

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

And check whether for the (_, a) (_, b) pair, 1 <= b - a <= 3 holds

prefixes (T, -) (T, 1) (T, 2) (F, 2) (F, 3) (F, 4)

suffixes (F, 1) (F, 2) (T, 2) (T, 3) (T, 4) (T, -)

A B C

A) 'a' is Nothing so prefix is empty

any good suffix merged with empty prefix is good

Aka: ~~1~~ ~~2~~ 2 3 4

B) 1 <= 3 - 1 <= 3

Aka: 1 ~~2~~ ~~3~~ 3 4

C) 1 <= 4 - 2 <= 3

Aka: 1 2 ~~3~~ ~~4~~ 4

And just out of curiosity... we had a 'bad' one, due to:

see the two consecutive 2s? prefix is bad :(

(F, 2)

(T, -)

1 2 2 ~~3~~ ~~4~~

see suffix is empty? suffix is good :)

Summary:

We changed from array indexing to list zipping.

For runtime behaviour, see:

https://youtu.be/gbO6X_aCxDE?t=230