

# How I won the Dyalog programming contest

Arianna  
Locatelli,  
sept. 2015

# About me

- I come from Uboldo, a little town about 30 minutes from Milan.
- I am 17 years old and so I currently am in my fourth year at Liceo Classico S.M. Legnani, and I mainly study humanistic subjects such as Latin and Ancient Greek.

# How I heard about APL

Despite I am not taught informatics at school, I was introduced to the world of programming by my math teacher Roberto Minervini.

During a year-long course I learnt more and more about APL, (legrande's book) until he proposed me to take part into the competition.

- Of course, I was aware that a high school student would stand little chance against undergraduates...

...but, in my case,  
something made the  
difference...



It is important to mention how the first step to the solution of each problem, apart from their most technical aspects, is of course their abstract understanding, and discussions and confrontations between me and my classmates were essential to find ideas that were later translated into APL language.

Without those moments, I certainly would have never managed to achieve the victory in this competition.

This is why I believe that working together can really help reaching extraordinary results which could never be achieved by yourself.

However even the help of internet has been very useful ;)



ROSALIND | Identify x Next Big Future: The x

nextbigfuture.com/2010/06/51-to-100-states-of-america-flag.html

App Google Webmail - TeleTu http://gasusa.a... TeleTu - Posta i...

**Generator**  
future, united states

Ad Support: Nano Technology Netbook Technology News Computer Software

A researcher looked at past flag designs that have been flown and determined the flag patterns that would be used if more states were added from 51 to 100. Slate created the flag pattern generator for six combinations based on the work of Skip Garibaldi of Emory University

**The N States of America** Slate

How would the stars on the flag be rearranged if more states joined the Union? Historically, there have been many configurations, six of which are included in this widget. Enter a number of stars in the box to see which fit your scenario. [Read more about the configurations in Slate.](#)

Stars: 51 long short **alternate** equal wyoming oregon

Embed This





# The Competition

The contest consisted of two phases:

Phase I had not very articulate problems that could each be solved with a single line of APL code .

Phase II, on the other hand, consisted of three sets of problems (Bioinformatics, Applications problems and Recreational and Game problems) of varying complexity.

Here, I will take into consideration the problems of both phases that I liked most and that I found most interesting.

However, in some cases, I will explain functions a little bit different from those I had sent for the contest, to simplify the exposure.

# PHASE I

## UNLUCKY 13

We have to change every 13 in 12.99 into an array.

So the function is:

$$\text{sol8} \leftarrow \{\omega - 0.01 \times 13 = \omega\}$$

I subtract from  
the starting  
vector the  
obtained vector

I multiply the  
boolean vector  
for 0.01

I obtain a Boolean  
vector in which 1  
corresponds to 13



# HE'S SO MEAN, HE HAS NO STANDARD DEVIATION (phase I)

The standard deviation of a population is calculated by taking square root of the average of the squared differences of the values from their average value. The mathematical formula is

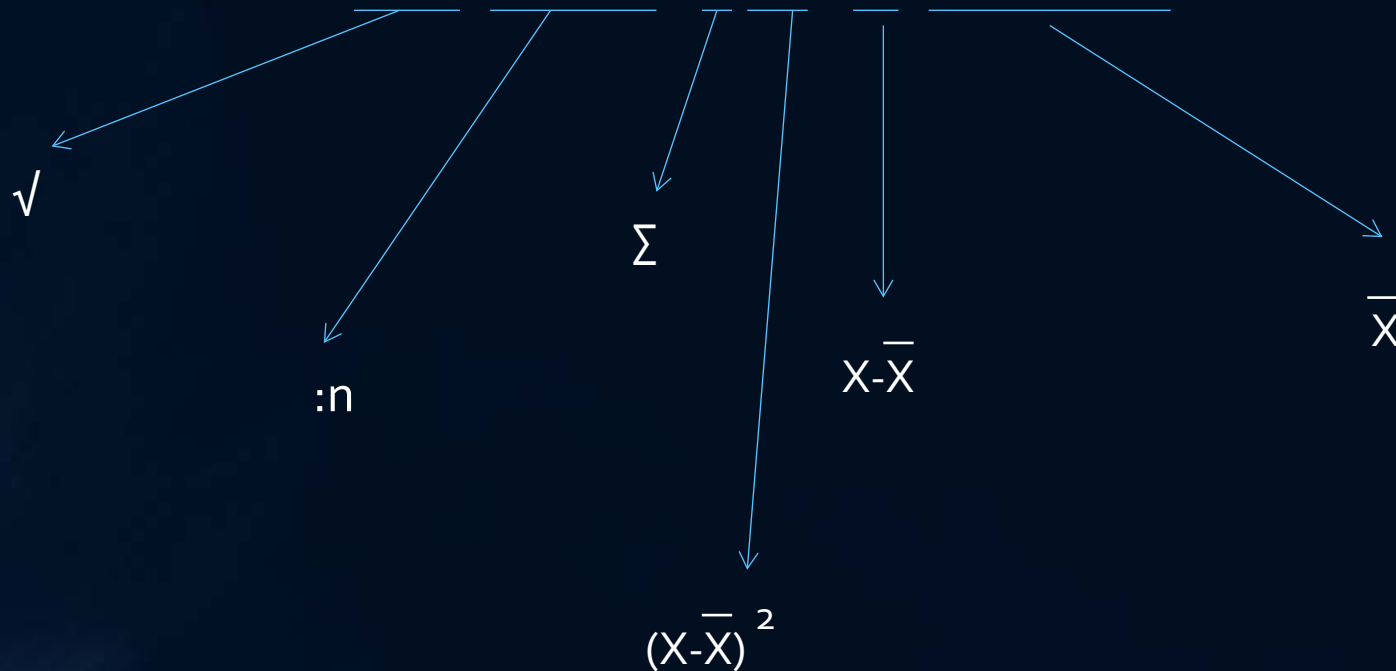
$$\sqrt{\sum(x-\bar{x})^2/n}$$

Write a dfn that returns the population standard deviation of its numeric array left argument.

# THE FUNCTION

$$\sqrt{\sum(x-\bar{x})^2 / n}$$

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$



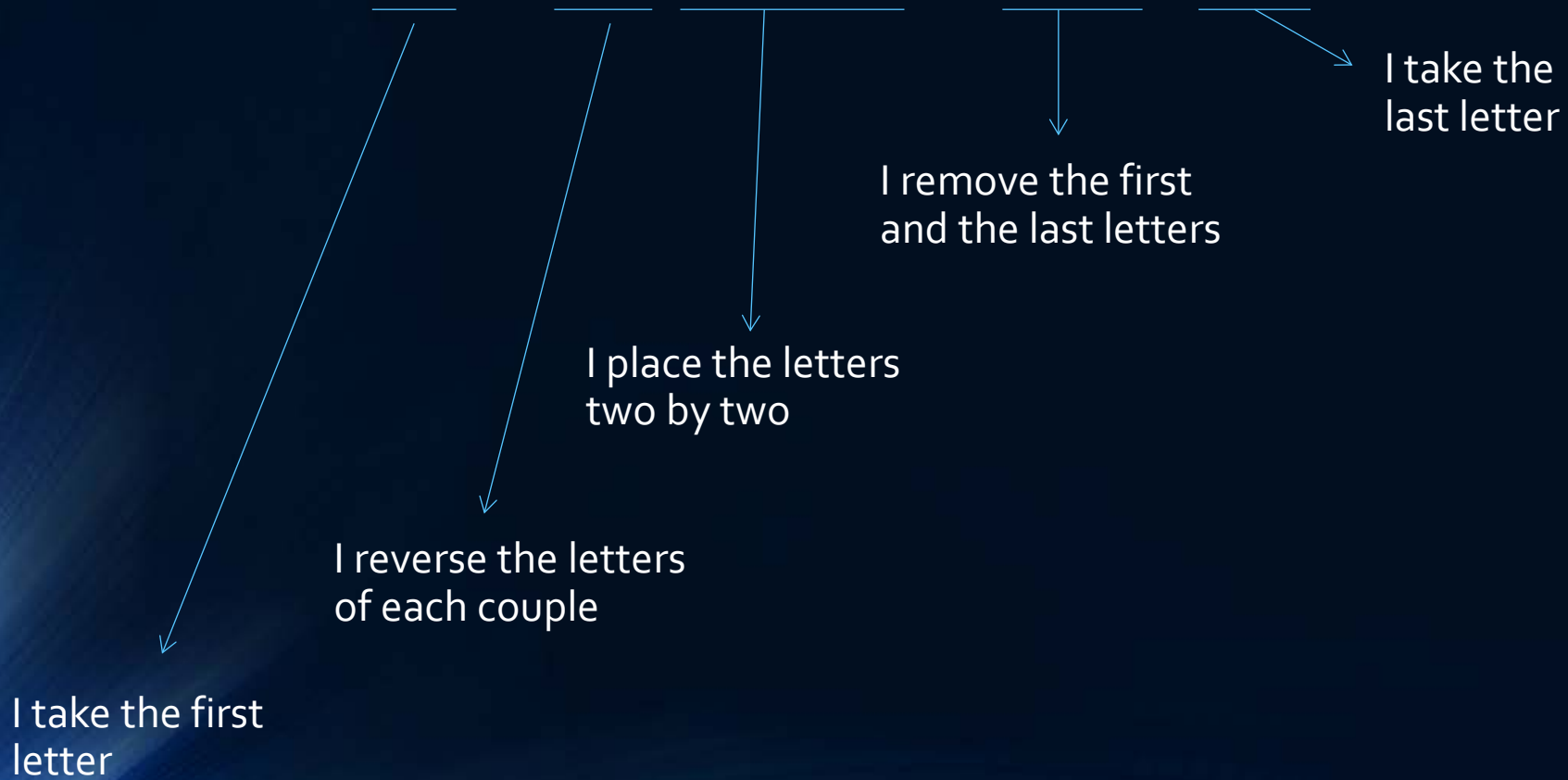
# I'D LIKE MY SCRAMBLED PLEASE (phase I)

Write a dfn that takes a character vector as its left argument and returns the word's letters inverted two by two, except the first and the last ones.

For example, 'argument' becomes 'agrmunet'.

# THE FUNCTION

$\text{solg} \leftarrow \{ \supset, / (1 \uparrow \omega), (\{ \Phi \text{ " } \omega \subset \sim 1 \text{ op } \sim \rho \omega \} 1 \downarrow \text{ } ^{-} 1 \downarrow \omega), \text{ } ^{-} 1 \uparrow \omega \}$



# PHASE II

Find Locs

Bioinformatics Problem 2 (medium difficulty)

The problem is to write an Apl function which:

-takes an integer vector representing  $L$   $L=\{l_1, l_2, l_3, \dots, l_x\}$

-returns an integer vector representing  $X$

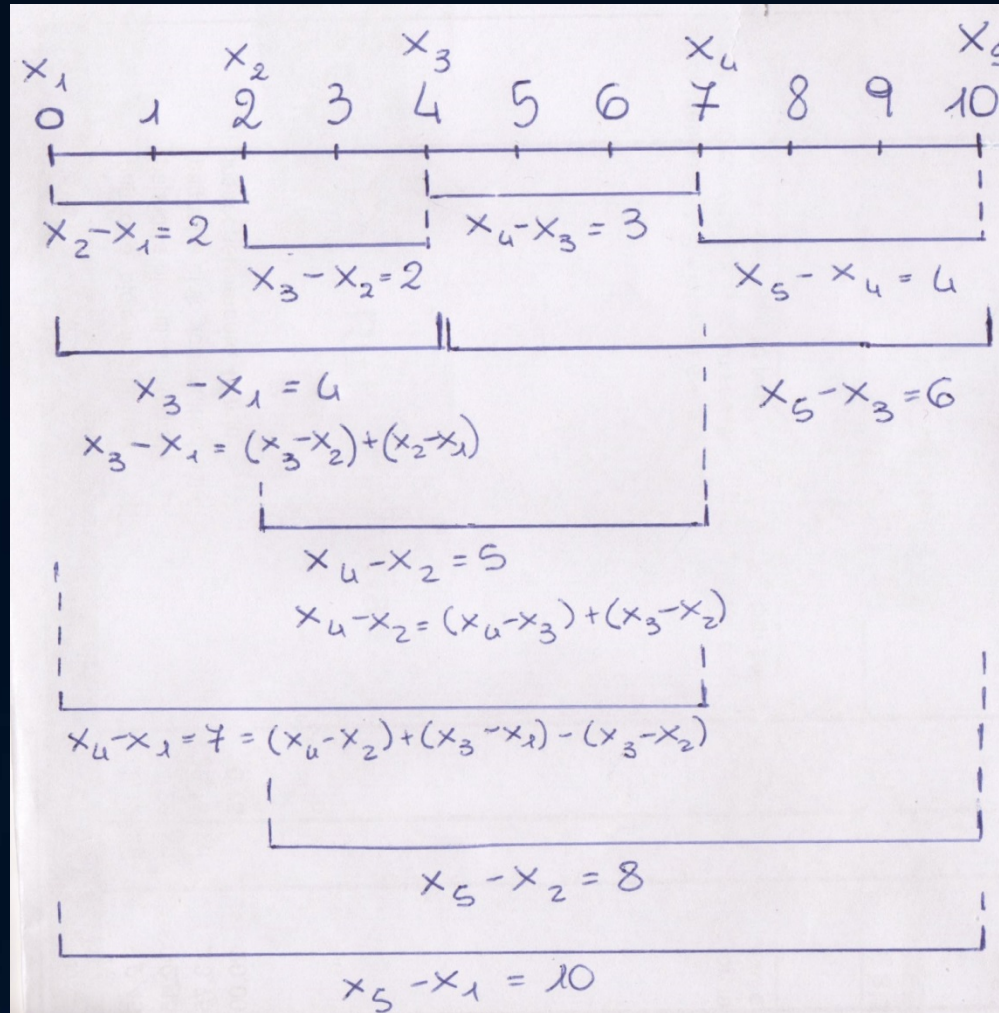
$X$  contains numbers  $X=\{x_1, x_2, x_3, x_4, \dots\}$  and  $L$  is a collection of all possible difference between all possible couple of  $X$ .

$$L=\{x_2-x_1, x_3-x_2, x_4-x_3, x_5-x_4, \dots\}$$

Knowing  $L$ , we have to find  $X$



We can see this example in order to have a graphical idea of what the set of differences is:



This is the final function:

```
x ← findLocs l; n
A Stub function for Bioinformatics Problem 2 Task 1
n ← 0.5 × 1 + 0.5 × √(1 + 8 × ρ l)
x ← n ↑ 0, + \ 1
```

I take the first n elements

I make the cumulative sum

I put a 0 before because I suppose that the first element is 0

$$n = \frac{1 + \sqrt{1 + 8k}}{2}$$

Firstly, knowing L, we have to understand how many elements X contains. If X contains n elements, then L contains an element for each pair of elements of X, so that

$$k = \frac{n(n-1)}{2}$$
$$2k = n^2 - n$$
$$n^2 - n - 2k = 0$$
$$n = \frac{1 + \sqrt{1 + 8k}}{2}$$



# Reversal Distance

## Bioinformatics Problem 3 (high difficulty)

One of the most studied problems in the field of computational biology is the string matching problem.

Why?

When trying to understand how genetic sequences mutate at the chromosome level we need to consider more global operations, rather than the basic ones. One of the most common global mutation is the reversing of a substring, operation also known as a reversal.

In order to represent the sequence of genes on a chromosome, we are going to use permutations. More clearly, we are given the order of  $n$  genes in two related organisms; we only consider the genes that appear in both organisms. The order of the genes are represented by a permutation  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ , where by  $\sigma_i$  we denote the position where the gene  $i$  appears.

A reversal of a permutation creates a new permutation by inverting some interval of the permutation;  $(5, 2, 3, 1, 4)$ ,  $(5, 3, 4, 1, 2)$ , and  $(4, 1, 2, 3, 5)$  are all reversal of  $(5, 3, 2, 1, 4)$ .



The first difficulty was understanding what a reversal is, in fact on Rosalind web site there was not a definition but an example. So I have examined the document 'Sorting by reversals' by Bogdan Pa\_saniuc.

A Reversal is a transformation that, taken a vector of ten elements from 1 to 10, creates a second permutation according to the following criterion:

A Reversal of  $[i,j]$  interval  
 $g(i, \dots, j-1, j) = (j, j-1, \dots, i)$   
With  $g(k) = k$  for  $k \notin [i,j]$



The function that makes the reversal is:

$\text{reversal} \leftarrow \{i \leftarrow (\bar{1} + \omega[1]) \downarrow \omega[2] \diamond a \leftarrow \alpha \diamond a[i] \leftarrow a[\ominus i] \diamond a\}$

This is the part  
that I want to  
modify (i)

Here I modify i (the interested part)  
with his reverse

Here we are using the Apl function 'reverse'  $\ominus$

So, applied to our example, it is

3 10 8 2 5 4 7 1 6 9 reversal 4 9

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

Let's talk about the problem:

The problem is to find the minimum number of reversals needed to transform one string into another, so the distance between two permutations.

In fact, the minimum number of reversals used to transform one string into another has proven to be a very good estimate for the evolutionary distance between organisms.

The first fundamental idea is that solving this problem

3 10 8 2 5 4 7 1 6 9 reversalDistance 5 2 3 1 7 4 10 8 6 9

Is exactly the same of solving this one

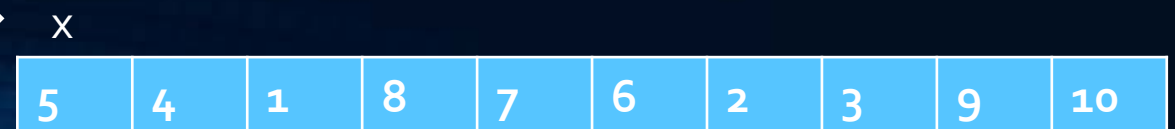
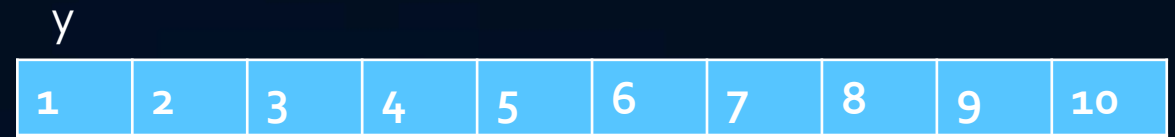
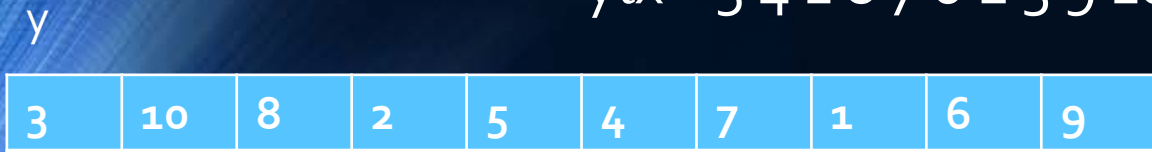
1 2 3 4 5 6 7 8 9 10 reversalDistance 5 4 1 8 7 6 2 3 9 10

This is helpful because now the left string is ordered and it is easier to pass from one permutation to the other.

$t \leftarrow y[x]$

I put instead of x the positions that numbers of x have in y

Y 3 10 8 2 5 4 7 1 6 9  
X 5 2 3 1 7 4 10 8 6 9  
y[x] 5 4 1 8 7 6 2 3 9 10



At this point, we have to understand how to solve the problem.

In other words we have to understand how to pass from a disordered permutation like 5 4 1 8 7 6 2 3 9 10 to the ordered one.

First of all,

We add two fictitious numbers at the beginning and at the end so we force in some way permutation to be increasing.

0	5	4	1	8	7	6	2	3	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----



A breakpoint of a permutation  $\sigma$  is a pair of adjacent positions  $[i, i + 1]$  such that  $\sigma[i]$  and  $\sigma[i+1]$  are consecutively decreasing or increasing.

The function that calculates the number of bk is:

$$bk \leftarrow \{ +/1 = |2 - /0, \omega, 1 + \rho\omega \}$$

It adds a number at the end

It makes the module of the difference between each number and his following

it counts the 1



These are all the breakpoints, in total they are 5

We have to use a reversal that improves the total number of bk



Now we have 6 breakpoints

The idea is to improve each time the number of bk until we arrive to 10 (this is a greedy algorithm)



So, at each step we try to maximize the number of "breakpoints".

The final function:

$z \leftarrow y$  revd  $x; t; bk; r; all; newt$

$t \leftarrow y \cup x$

$bk \leftarrow \{+ / 1 = | 2 - / 0, \omega, 1 + \rho \omega \}$

$r \leftarrow \{ i \leftarrow ( - 1 + \omega [ 1 ] ) \downarrow \omega [ 2 ] \diamond a \leftarrow \alpha \diamond a [ i ] \leftarrow a [ \Phi i ] \diamond a \}$

$all \leftarrow \{ a \leftarrow , ( \omega , \omega ) \times ( \omega ) \circ . < \omega \diamond a / \sim 0 < + / " a \} \rho t$

$z \leftarrow 0$

It finds all the couples in which the first element is lower than the second

It takes these couples

:Repeat

$newt \leftarrow t \cup r \cdot all \longrightarrow$  It makes all the possible reversal

$t \leftarrow \sup newt [ \sup \nabla bk \cdot newt ]$

It orders the reversals putting at the beginning the ones with the highest number of breakpoints, and takes the first one

$\square \leftarrow t$

$z \leftarrow z + 1$

:Until  $10 < bk \ t$

# LONGEST SUBSEQUENCES

## Bioinformatics Problem 3 (high difficulty)

Write an APL function, which:

-takes an integer permutation vector of length  $n$

-returns a 2-element vector where:

- [1] is a longest increasing subsequence of the permutation vector
- [2] is a longest decreasing subsequence of the permutation vector

Example:

longestSubsequence 5 1 4 2 3

1	2	3	5	4	2
---	---	---	---	---	---

So we have to find the increasing and the decreasing subsequences of the permutation vector.

However finding the increasing one is the same of finding the decreasing one, because if we change sign at each number and we find the increasing subsequence, then we change again the sign we find the decreasing subsequence.

How can I find the increasing subsequence?

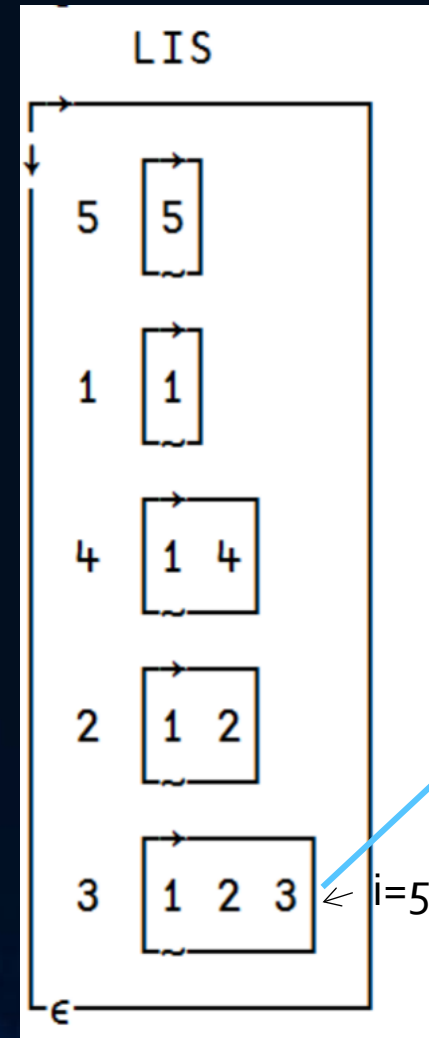
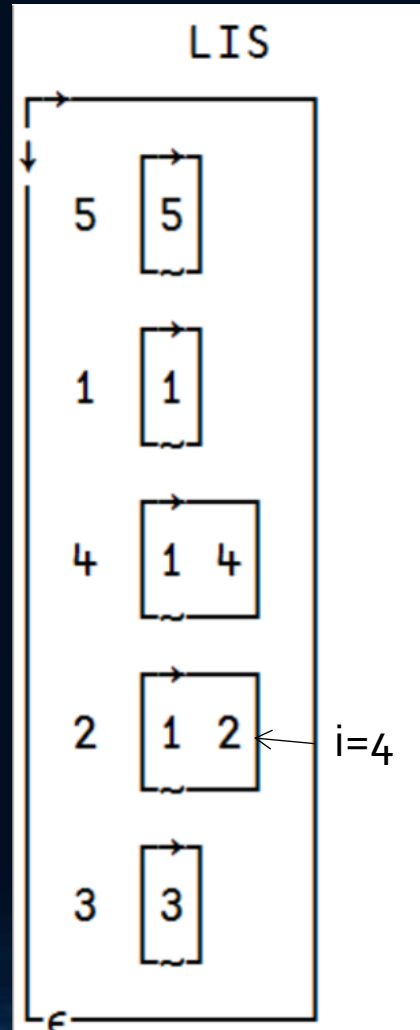
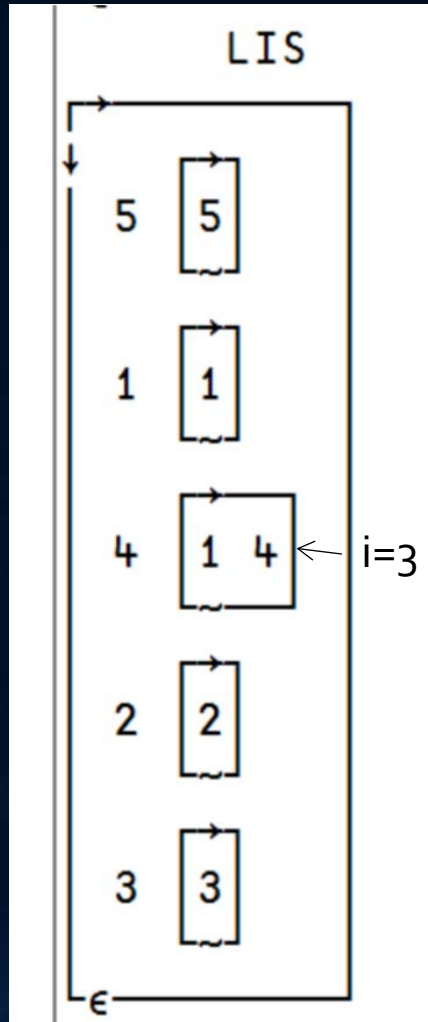
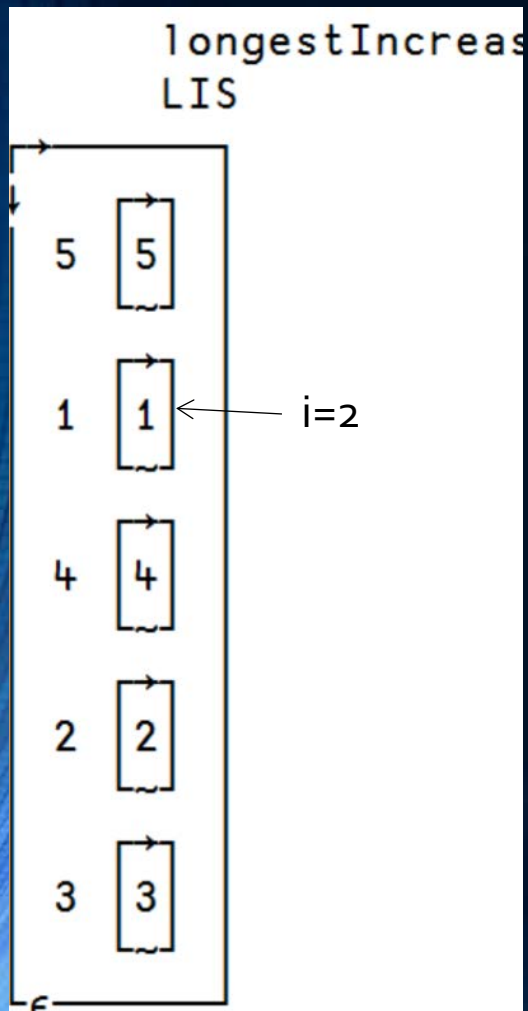
5	1	4	2	3
---	---	---	---	---

We can focus on the last number (3) and search the subsequences that contain 3 as the last element.

In this case 1-3 and 1-2-3, where the longest is the second one.

However it's easier to start from the beginning and go ahead, with dynamic programming, by searching for each number all the previous numbers lower than the number itself.

Let's see an example (5, 1, 4, 2, 3)



# THE FINAL FUNCTION

$z \leftarrow \text{lis } x; \text{LIS}; j; i; \text{lmax}$

$\text{LIS} \leftarrow \{\omega, [1.1], \omega\}x$

$\text{lmax} \leftarrow \{\sup \omega [\sup \psi \sup \rho \omega]\}$

:For  $i : \text{In } 1 \downarrow \uparrow x$

$j \leftarrow (\uparrow x) / \sim (i > \uparrow x) \wedge x[i] > x$

:If  $0 < \sup j$

$\text{LIS}[i; 2] \leftarrow \subset x[i], \sim \text{lmax LIS}[j; 2]$

:EndIf

:EndFor

$z \leftarrow \text{lmax LIS}[:, 2]$

⊙ initial matrix

⊙ it finds the longest subsequence

⊙ I start from the second item

⊙ it finds all the indices lower than  $x[i]$

⊙ if there is at least one  $j$

⊙ it updates the matrix

⊙ it takes the longest subsequence

# HtmlTable

## Applications Problem 1 (low difficulty)

The task is to write an Apl function that:

- takes a matrix of data as its right argument

- returns a character vector of HTML to render the matrix as an HTML table

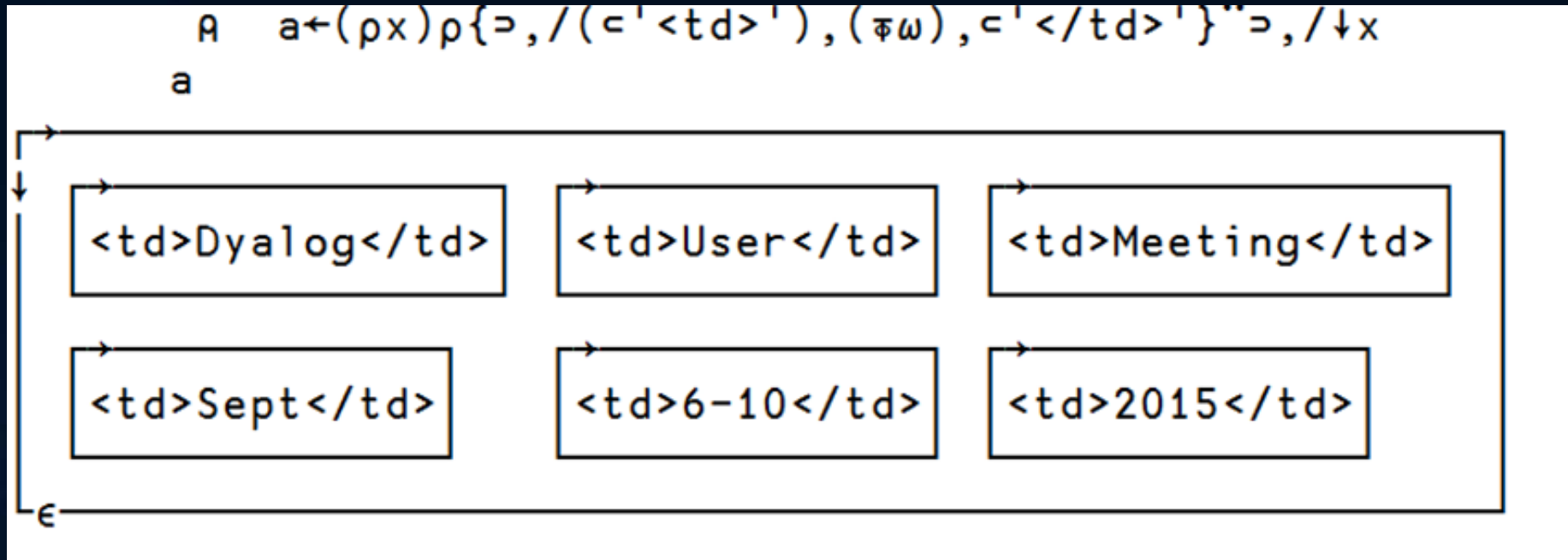


This should be the result:

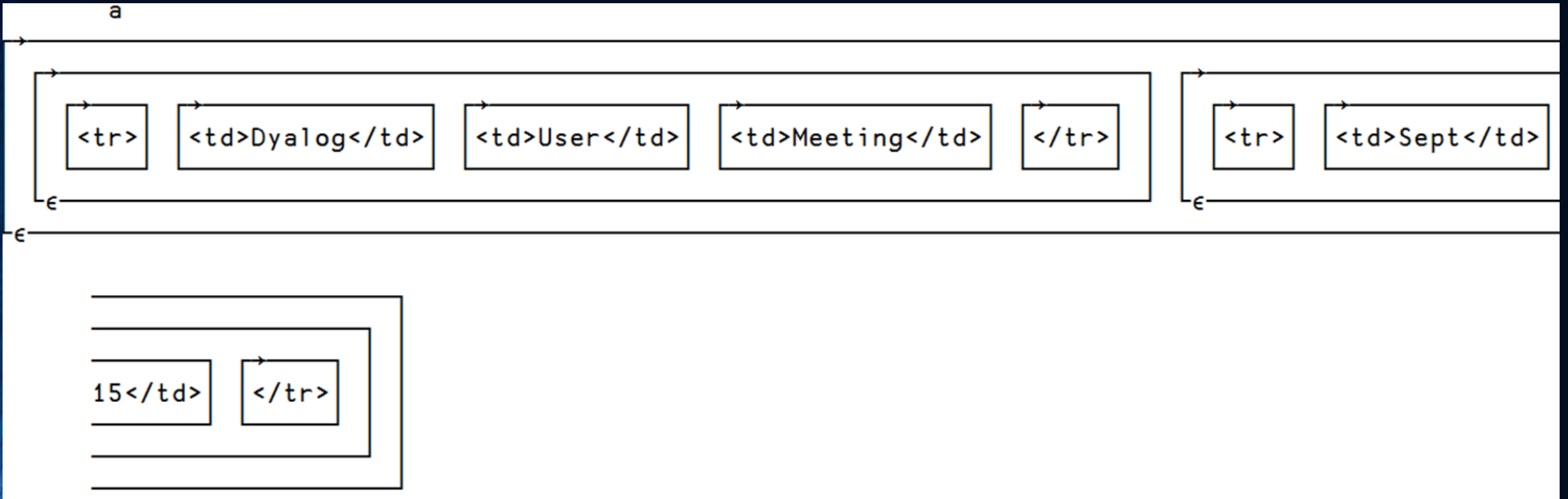
```
htmlTable 2 3p'Dyalog' 'User' 'Meeting' 'Sept' '6-10' 2015
→
<table>
  <tr>
    <td>Dyalog</td>
    <td>User</td>
    <td>Meeting</td>
  </tr>
  <tr>
    <td>Sept</td>
    <td>6-10</td>
    <td>2015</td>
  </tr>
</table>
```



The first operation we do is to put a 'td' before and '</td>' after each cell of the matrix



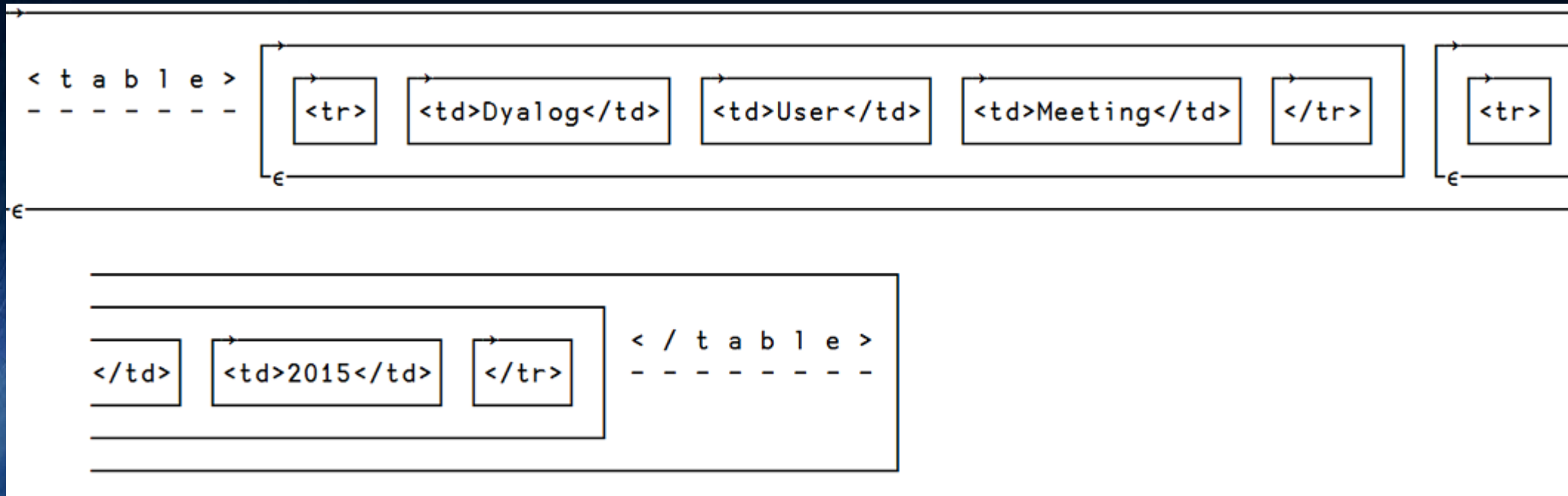
Now we pack the lines, by putting a 'tr' before and a '</tr>' after



Then we pack the entire table

$$A \ a \leftarrow \{ ' \langle \text{table} \rangle ' , \omega , ' \langle / \text{table} \rangle ' \} a$$

a

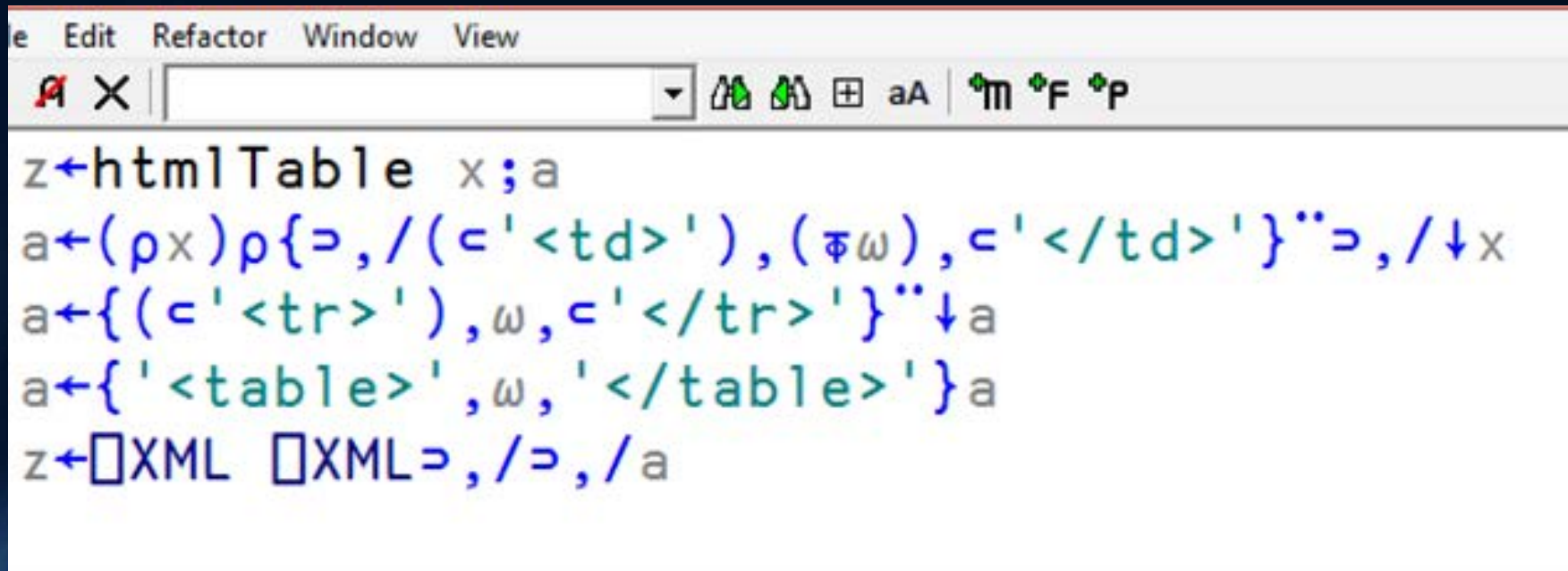


And finally we clean the nested data

```
=,/=,/a
```

```
<table><tr><td>Dyalog</td><td>User</td><td>Meeting</td></tr><tr><td>Sept</td><td>6-10</td><td>2015</td></tr></table>
```

And we obtain the final function



```
e Edit Refactor Window View  
z←htmlTable x;a  
a←(ρx)ρ{=,/(c'<td>'),(⊖ω),c'</td>'}"=,/=,↓x  
a←{(c'<tr>'),ω,c'</tr>'}"↓a  
a←{'<table>',ω,'</table>'}a  
z←⊠XML ⊠XML=,/=,/a
```

# KENKEN

## Recreation and Games Problem 3 (high difficulty)

Your task is to write a program which will solve a KenKen puzzle.

The program:

-takes a 3-column matrix right argument

- Column 1 contains the integer target number
- Column 2 contains a character scalar representing the operation (one of + - x :)
- Column 3 contains a vector of cell coordinates that make up the cage

-returns an integer matrix representing the solution to the puzzle

A 4x4 KenKen puzzle grid. The grid is divided into cages by thick black lines. The clues are as follows:

16x		7+	
2-			4
	12x	2÷	
		2÷	

Solving this problem consist in solving a sudoku, and the solution starts from the tecnique thought by John Scholes, presented in the YouTube video "A Sudoku Solver in APL".

However this tecnique solves a generical sudoku, while we have to solve a KenKen.

The problem of a KenKen is the fact that there is a strong condition on each cage:

-there is a target number  $t=16$

-there is an operation  $o='x'$

-there are involved cells  $(1\ 1),(1\ 2),(2\ 2)$  that tell us the dimension of the cage  
 $k=3$

16x		7+	
2-			4
	12x	2÷	
		2÷	

The first problem is: can I write a function, the FillCage, which tries to fill with numbers the indicated cells so that they respect the conditions?

So the embryonic function of this solution is the FillCage which has:

-Input

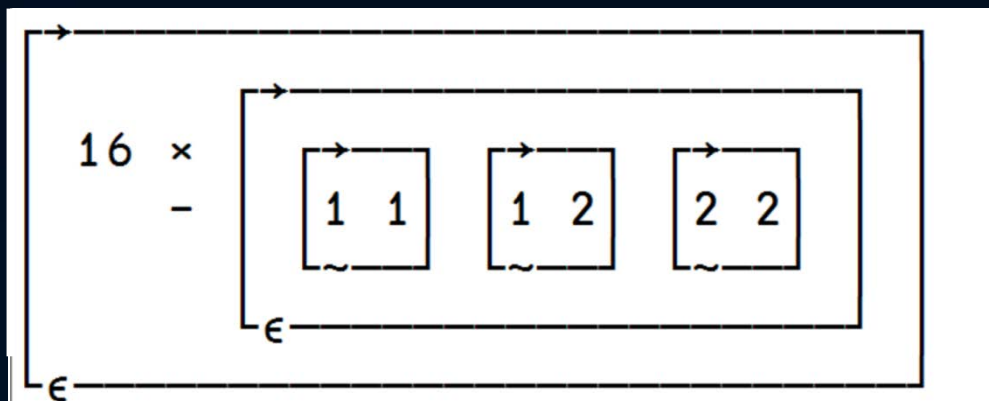
- n dimension of KenKen
- t integer target number
- o operation + - x :
- v vector of cells coordinates that make up the cage

-Output:

- the matrix filled only in the cells indicated by the Input

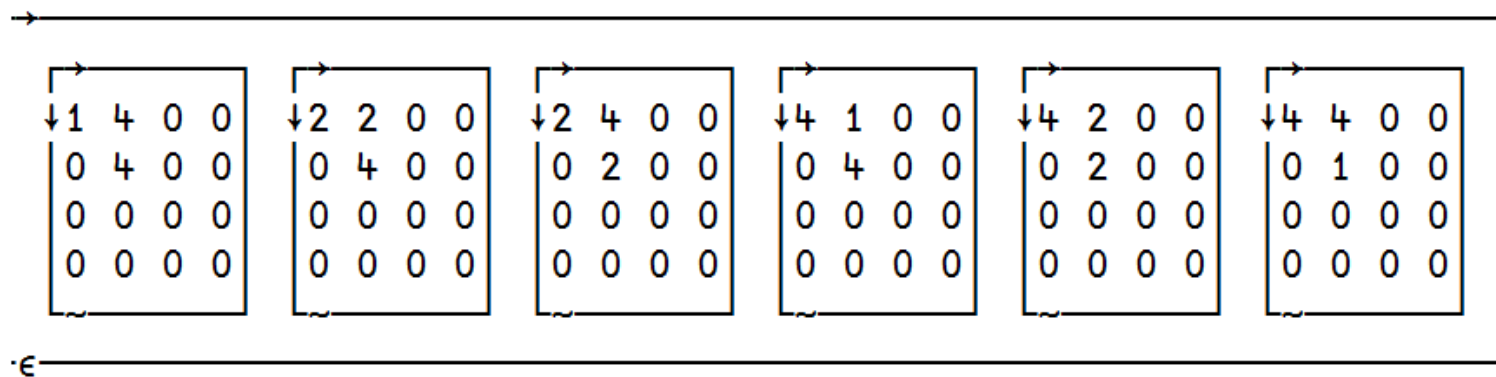
<b>16x</b>		<b>7+</b>	
<b>2-</b>			<b>4</b>
	<b>12x</b>	<b>2÷</b>	
		<b>2÷</b>	

# Let's see the FillCage



As you can see there are solutions that are not correct, but we are going to remove them with the 'sum' function in the kenken function

4 fillcage 16, 'x', c(1 1)(1 2)(2 2)



Above it's shown the structure of the argument, as you can see the third element of the vector is a vector that contains the coordinates of the cells to fill. This makes the input a nested vector. Also the output is a vector of matrices, so it's nested too.



# I explore all the possible combinations

$((14) \circ \{\omega \circ . \times \alpha\} * 2) 14$

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
2	4	6	8
4	8	12	16
6	12	18	24
8	16	24	32
3	6	9	12
6	12	18	24
9	18	27	36
12	24	36	48
4	8	12	16
8	16	24	32
12	24	36	48
16	32	48	64

I check where the combinations are equal to the target number

$16 = ((14) \circ \{\omega \circ . \times \alpha\} * 2) 14$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0
0	0	0	1
0	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1
0	1	0	0
0	0	0	0
1	0	0	0

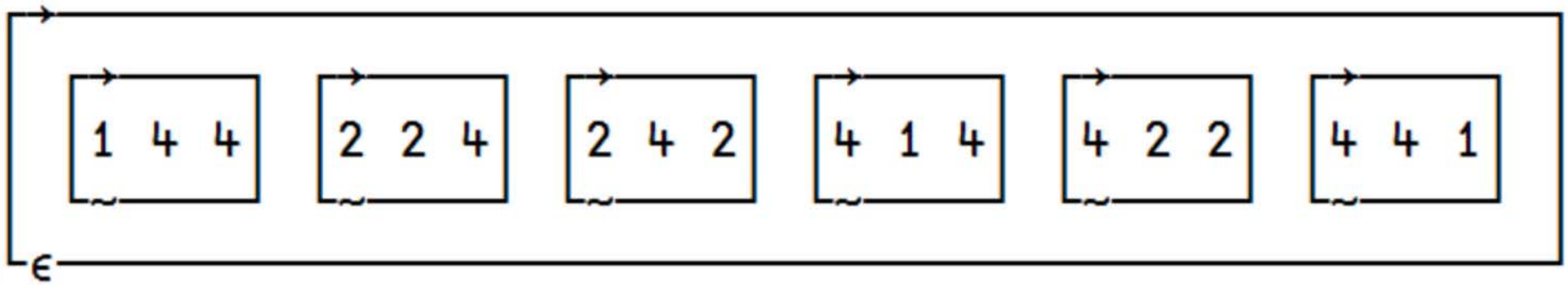
# I discover the coordinates of the cube where there is 16

$$\text{where } \leftarrow \{(\omega \neq 0) / \rho \omega\}$$

It takes all the 1 (searches where  $\omega \neq 0$ )

It gives me the vector of all the coordinates

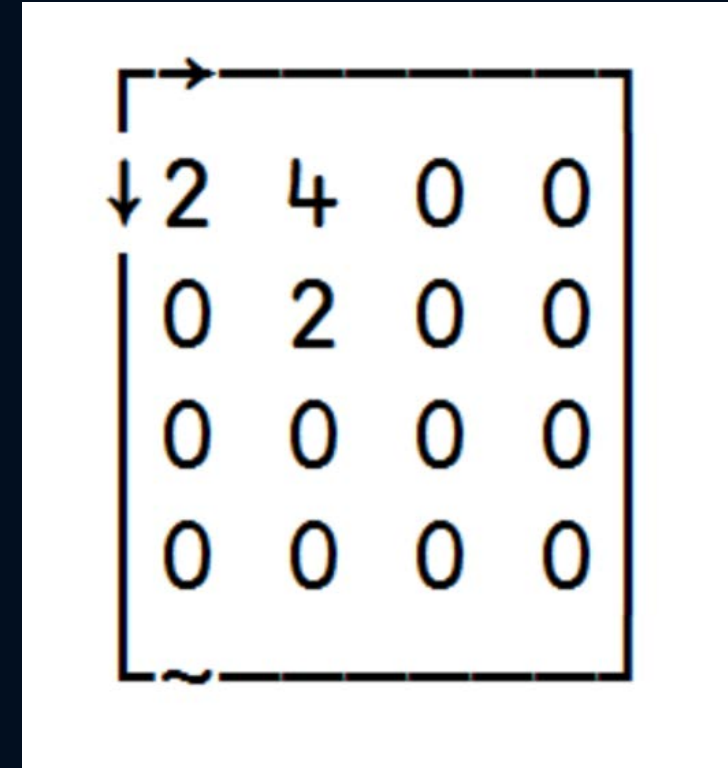
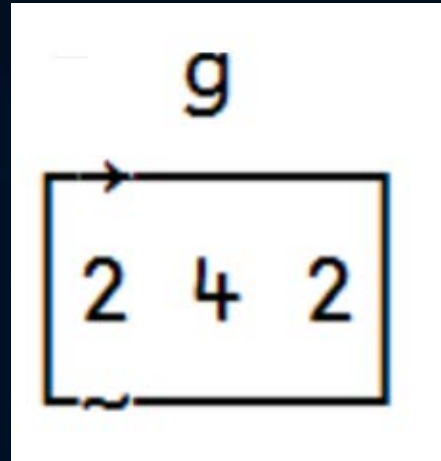
$$\text{where } 16 = ((14) \circ \{\omega \circ \cdot \times \alpha\} * 2) 14$$



In this way I find the sets of number whose product is 16

After we have found the possible solutions, we can place them into the matrix, with the function 'at'

For example:



$at \leftarrow \{a \leftarrow (n \times n) \rho \circ \diamond a[(, in n) \iota \alpha] \leftarrow \omega \diamond n \ n \rho a\}$

It makes the matrix

It places the numbers ( $\omega$ ) in the cells ( $\alpha$ )

# THIS IS THE FINAL FILLCAGE

Z ← n fillcage x;t;o;c;k;f;m;at    Ⓢ we have to search and place the number

Ⓢ n dimension of kenken

t o c ← x

f ← { | ⊥ ⊕ α, o, ω }    Ⓢ it executes char operation in a function

at ← { a ← (n × n) ρ ◊ a [ (, in n) i α ] ← ω ◊ n n ρ a }

k ← ⊃<sup>-1</sup> + ρ c

:If o = ' '

z ← C c at t

:Else

m ← (in) ( { α ◊ f ω } \* k ) in    Ⓢ I make all the possible combinations

z ← c ◊ at ( t = , m ) / , i ρ m    Ⓢ I take the coordinates of the cells with numbers that make the target number

:EndIf

Now I repeat this process for the second cage, and I obtain 8 possibilities, so in total they are 16, among which I can take only the ones that don't have the same number repeated on a row or a column.

So I have to define a sum function that doesn't accept repetitions.

$$\text{sum} \leftarrow \{a \leftarrow, \alpha \circ . + \omega \diamond f \leftarrow \{\wedge / \{\{\omega \equiv \cup \omega\} \omega \sim 0\} \downarrow \omega\} \diamond a / \sim (f \circ \circ a) \wedge f \circ a\}$$

It makes all the combinations of the sum of two cages

it controls that there aren't two numbers repeated on the same row

It applies f on the transpose of a, so makes the test also on the columns

I repeat this process for each cage and i find the final solution.

# THIS IS THE FINAL FUNCTION

$z \leftarrow \text{kenken } x; n; \text{sum}$

$n \leftarrow 0.5 * \sim \supset \rho \supset, /x[:3]$      $\rho$  dimension of the KenKen

$\text{sum} \leftarrow \{a \leftarrow, \alpha \circ . + \omega \diamond f \leftarrow \{ \wedge / \{ \{ \omega \equiv \cup \omega \} \omega \sim 0 \} \downarrow \omega \} \diamond a / \sim (f \circ \ominus \circ a) \wedge f \circ a \}$

$z \leftarrow \supset \supset \text{sum} / n \circ \text{fillcage} \circ \downarrow x$

# Identifying Maximal Repeats

## Bioinformatics Problem 3 (high difficulty)

The task is to write an Apl function which:

- takes a character vector representing a DNA string
- returns a vector of character vectors containing all maximal repeats in the DNA string having a length greater or equal to 20





In order to simplify the problem I considered a short string used as an example by the Rosalind website.

So I took the DNA string 'TAGTTAGCGAGA', in which I have to find a 2-long substring, not a 20-long one. This economizes everything

$L \leftarrow 2$   $\odot$  minimum length of maximal repetition

The problem is divided in two points, in fact I have to find all the substrings which:

1) repeat, so there must be at least two of them

2) are not extensible (so they can't be done in a better way), because I have to find the maximal substring, the longest repeated one

"For example, "AG" is a maximal repeat in "TAGTTAGCGAGA" because even though the first two occurrences of "AG" can be extended left into "TAG", the first and third occurrences differ on both sides of the repeat; thus, we conclude that "AG" is a maximal repeat. Note that "TAG" is also a maximal repeat of "TAGTTAGCGAGA", since its only two occurrences do not still match if we extend them in either direction."

The idea I followed is the "Chrochemore algorithm" because it perfectly combines with the Apl function 'find' ( $\epsilon$ ). So:

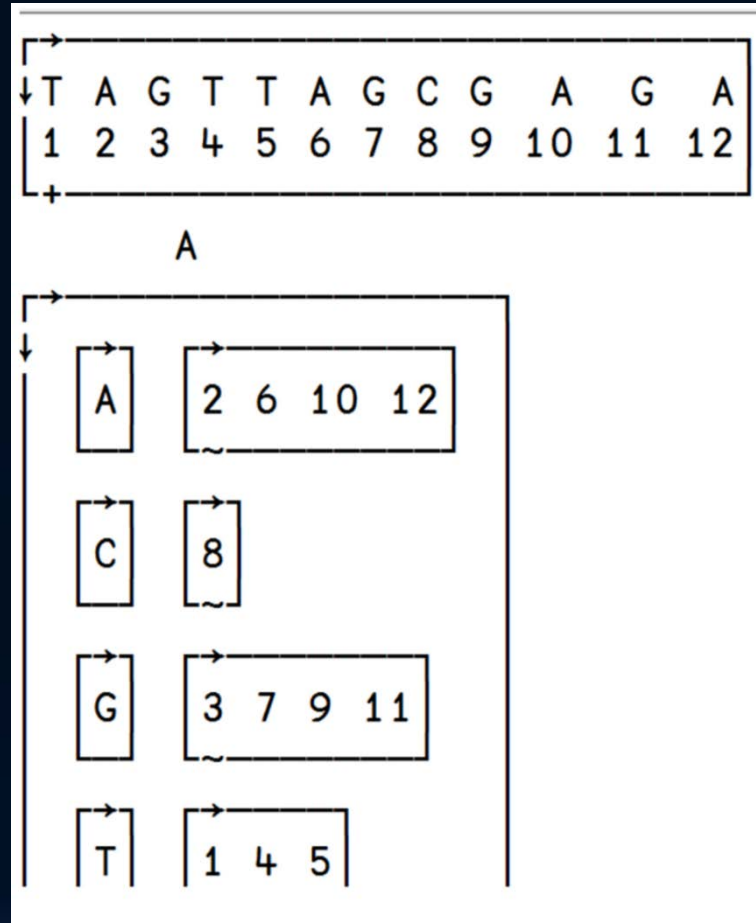
- For each letter of the Dna I consider how it can be extended
- I obtain the substrings of length 2 and I try to extend them
- I go on with this process until I obtain the longest repeated substring

This algorithm seems slow, but it 's not like that.



$A \leftarrow M \leftarrow (, 'ACGT'), [1.1] \times \{(\alpha = \omega) / \rho \alpha\} 'ACGT'$

With this function I search where the letters 'ACGT' are in the dna string, and at the beginning this is the result:



This is the first turn. Now A and M coincide, but M will remain the same, while A will evolve

Now, I try to extend the substrings, by creating that of two elements in this way

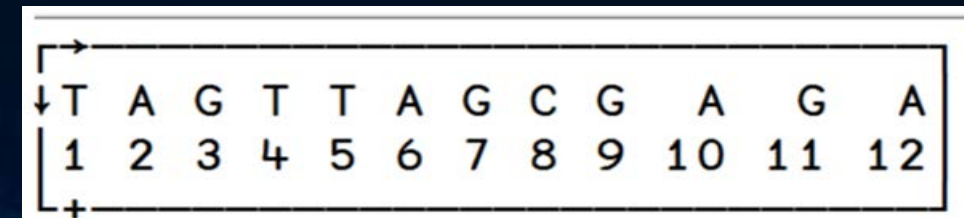
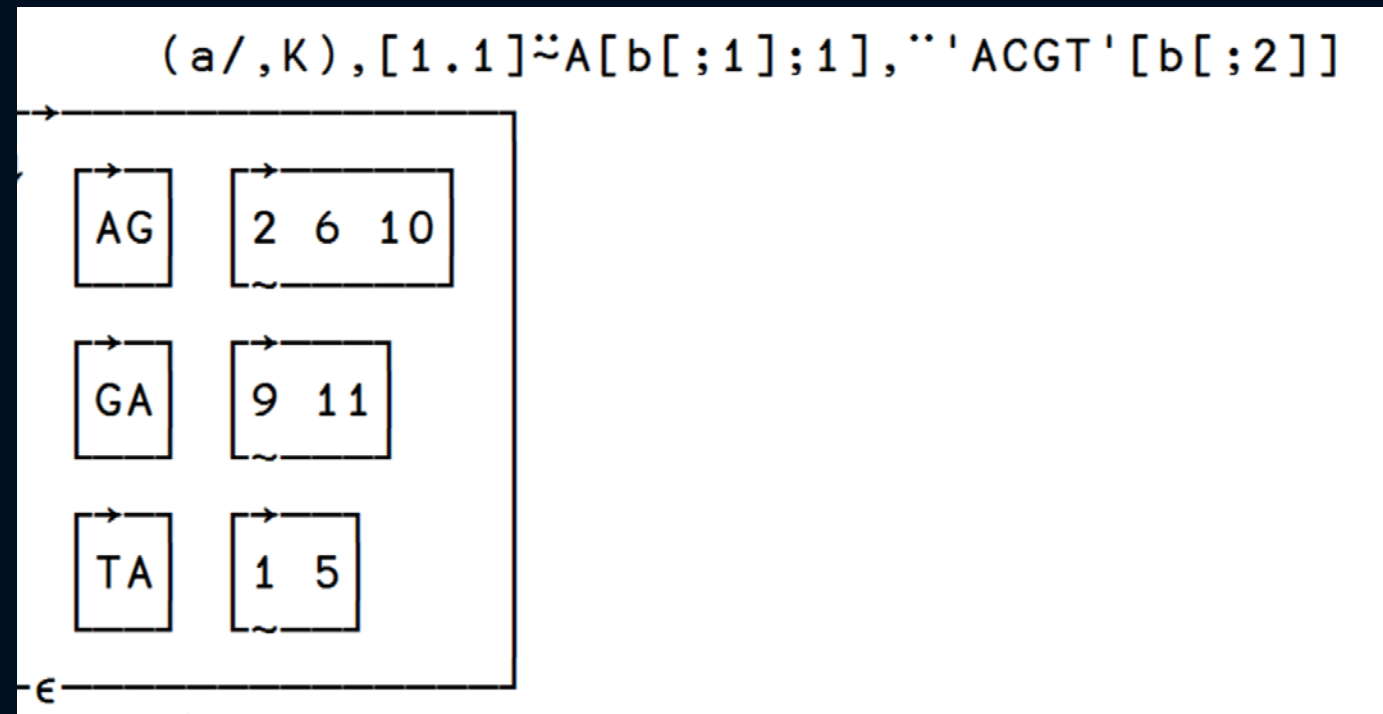
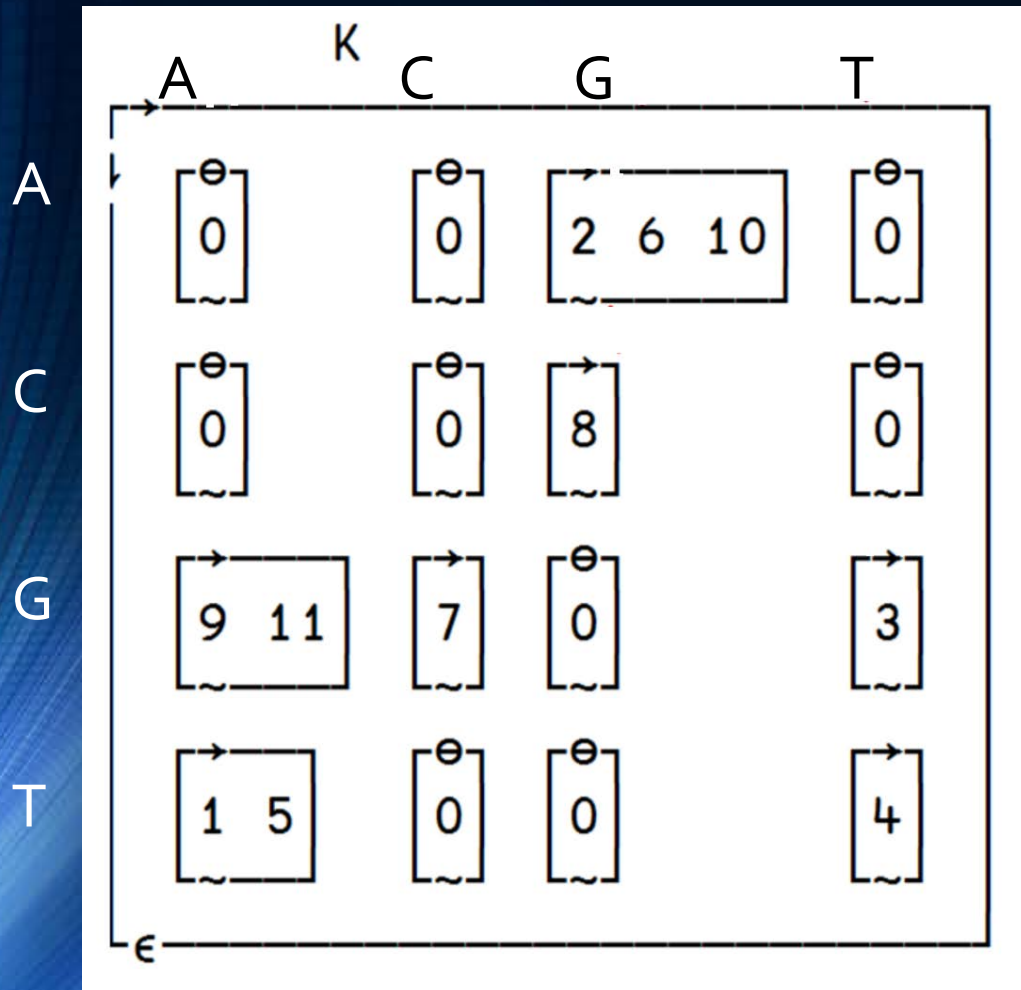
I find the positions of the couples

$$K \leftarrow A[:,2] \circ (d f) M[:,2]$$

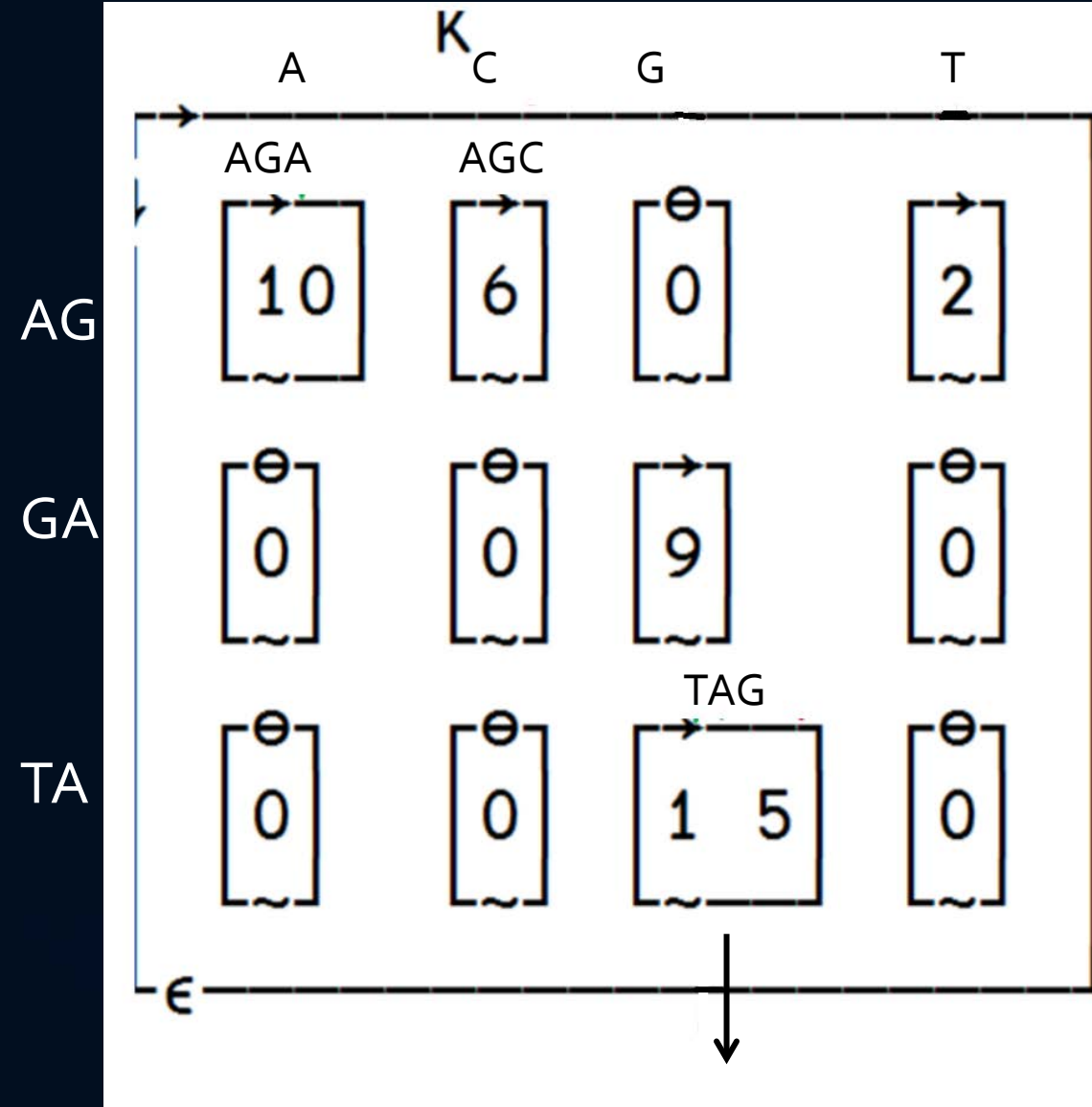
I find the repeated strings and their positions

$$A \leftarrow ((a/,K), [1.1] \sim A[b[:,1];1], 'ACGT'[b[:,2]])$$

$$(a/,K), [1.1] \sim A[b[:,1];1], 'ACGT'[b[:,2]]$$



I go on with this process to find the substrings of three letters



Only this survives because we have found it twice

# FINAL FUNCTION

$z \leftarrow \max_{R, x; M; A; f; K; L; b; a; d; g}$

$L \leftarrow 20$   $\hat{\rho}$  number of minimum occurrences

$f \leftarrow \{ \alpha / \sim ( \rho \alpha ) \in ( \alpha \alpha = a ) / ( \rho \alpha \leftarrow , \omega \circ . - \alpha ) \rho \rho \alpha \}$   $\longrightarrow$  It will be used to search the positions in which the substrings can be extended

$g \leftarrow$  \*next slide

$A \leftarrow M \leftarrow ( , "ACGT" ) , [ 1 . 1 ] \times \{ ( \alpha = \omega ) / \rho \alpha \}$  "ACGT"

$d \leftarrow 0$   $\hat{\rho}$  length of the string that I want to extend

$z \leftarrow 0$   $2 \rho \theta$

:Repeat

$d \leftarrow d + 1$

$K \leftarrow A [ ; 2 ] \circ . ( d \ f ) M [ ; 2 ]$

$b \leftarrow \uparrow \{ ( , \rho \omega ) / \sim , \omega \}$   $a \leftarrow 1 < \supset \rho \text{ " } K \longrightarrow$  it finds the positions of the repeated substrings

$A \leftarrow ( ( , a ) / , K ) , [ 1 . 1 ] \sim A [ b [ ; 1 ] ; 1 ] , "ACGT" [ b [ ; 2 ] ] \longrightarrow$  it finds the repeated substrings

$z \leftarrow z \ g \ A \longrightarrow$  it removes the shorter substrings that are contained in the longer substring that I have found

$z \leftarrow z \ \bar{\rho} \ A \ / \ \sim , \supset \text{ " } L \leq \rho \text{ " } A [ ; 1 ] \longrightarrow$  it puts together the obtained results

:Until  $0 = + / , a$

$z \leftarrow \bigoplus z [ ; 1 ] \longrightarrow$  It reverses the results to put at the beginning the most repeated substring

g ← {

a ← { (ρ ω) ρ ∨ / " , ω } α [ ; 1 ] ° . ∈ ω [ ; 1 ]

It searches which substrings for example of 2 elements are contained in that of 3 elements

b ← ↑ , ln ← ( ∃ ρ α ) , ∃ ρ ω

b ← n ρ , ∃ " ( ρ " α [ b [ ; 1 ] ; 2 ] ) = ρ " ω [ b [ ; 2 ] ; 2 ]

It controls if the shorter substring occurs the same number of times as the longer substring in which it is contained

α / ~ ~ ∨ / b ∧ a

It removes the shorter substrings that are always contained in the longer one

}

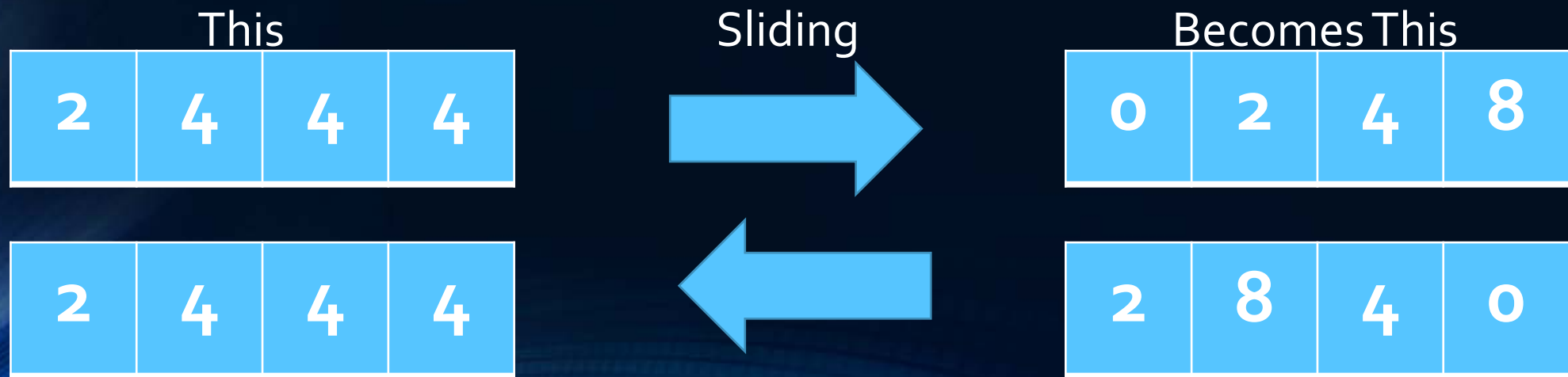
# 2048: Task 1 and 2

## Recreation and Games Problem 2 (medium difficulty)

These problems are based on the game 2048, first implemented on the web during March 2014 by Gabriele Cirulli.

At the start of the game, 2 random cells of  $4 \times 4$  grid are assigned a value of 2; the user has to indicate a direction (up, down, left, right) to slide the cells as far as possible in the chosen direction until they are stopped by another cell or the edge of the grid. If two cells of the same number collide, they will merge into a cell with the total value of the two cells that collided.

For example:





# 2048:Task 1 – Shifty Thinking

The task is to write a function, `shift2048`, which:

- takes a right argument which is a 4 element integer vector representing 4 cells (0 indicates a blank cell)
- takes a Boolean scalar left argument which indicates the direction to shift (1 for shift to the right, 0 for shift to the left)
- returns a 4 element integer vector representing the result after the shift.

Examples:

```
1 shift2048 2 4 4 8
```

```
0 2 8 8
```

```
0 shift2048 2 4 4 8
```

```
2 8 8 0
```

Our function will work this way:

1 shift 2048 2 4 4 8

0 2 8 8 8

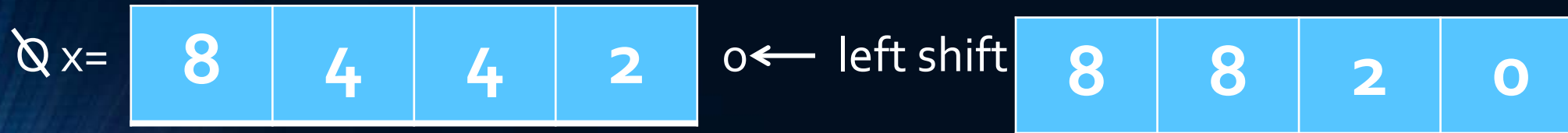
In fact not only it will return the vector, but also the sum 8 and this will be useful for the second part of the problem.

The main idea of the function is that if I find the way to shift to the left, I will be able to shift to the right, too. This simplifies everything because I can do all the reasonings at once.

I will explain better



I reverse the starting vector



I reverse again and the result is the same



# Now let's see the logic of the solution:

2	4	4	8
---	---	---	---

o ← left shift

2	8	8	0
---	---	---	---

2	4	4	8
---	---	---	---

x

4	4	8	0
---	---	---	---

s0(x)

0	1	0	0
---	---	---	---

$a \leftarrow \{ \omega = s_0(x) \} x$

It takes the couple of equal numbers

2	8	4	8
---	---	---	---

$z \leftarrow x + ax$

It makes the sum of x and ax

0	0	1	0
---	---	---	---

$b \leftarrow s_1 a$

It makes the shift to the right of a

1	1	0	1
---	---	---	---

$\sim b$

It denies b

2	8	8
---	---	---

$(\sim b) / x + ax$

It adapts b on z

Finally I add the sum (8)

$s \leftarrow + / 2 \times X \times a$

2	8	8	0
---	---	---	---

$z \leftarrow (px) \uparrow (\sim b) / z$  It adds a zero at the end

2	8	8	0	8
---	---	---	---	---

The final function is:

$z \leftarrow y \text{ shift}_{2048} x; s_0; s_1; a; b$

$s_0 \leftarrow \{1 \downarrow \omega, 0\}$        $\odot$  left shift (I remove the first number and I add a 0)

$s_1 \leftarrow \{\bar{1} \downarrow 0, \omega\}$        $\odot$  right shift

:If  $y=1 \diamond x \leftarrow \bigoplus x \diamond$  :EndIf

$a \leftarrow \langle x = s_0 \ x$        $\odot$  the result is 0 1 0 0     $\langle$  takes only the 1 at left if there are more 1

•  $z \leftarrow x + x \times a$        $\odot$  the result is 2 8 4 8

$b \leftarrow s_1 \ a$        $\odot$  the result is 0 0 1 0

•  $z \leftarrow (\rho x) \uparrow (\sim b) / z$        $\odot$  the result is 2 8 8 0

•  $z \leftarrow (\rho x) \uparrow z / z \ \ddot{\sim} z \neq 0$        $\odot$  if there are 0 at the beginning, in the middle or at the end I remove them

:If  $y=1 \diamond z \leftarrow \bigoplus z \diamond$  :EndIf

$s \leftarrow + / 2 \times x \times a$        $\odot$  s is the sum

$z \leftarrow z, C, s$

## 2048:Task 2 – All a board

Now the problem becomes two-dimensional, because there is a Board, but if we imagine the single rows of the board as vectors and we apply the shift to each row, it is clear that applying the shift to the matrix is equivalent to apply it to each row.

The two vertical shifts (up and down) added to the horizontal ones (left and right)

Also, the vertical shifts can be traced to the horizontal shifts, after rotating the matrix and the result

This is what happens graphically:

current

0	0	4	2
2	4	16	4
2	8	8	32
64	2	2	4

↑<sub>2 up</sub>

4	4	4	32
64	8	16	4
0	2	8	32
0	0	2	4

0 shift 2 0 4 8 ~~0~~ current

~~0~~

~~0~~ current

0	2	2	64
0	4	8	2
4	16	8	2
2	4	32	4

0 ←  
left shift

4	64	0	0
4	8	2	0
4	16	8	2
2	4	32	4

4	4	4	32
64	8	16	4
0	2	8	32
0	0	2	4

# FINAL FUNCTION

$z \leftarrow y \text{ board}_{2048} \ x; s_2; k;$

$\odot \ 0 \leftarrow$

$\odot \ 1 \rightarrow$

$\odot \ 2 \uparrow$

$\odot \ 3 \downarrow$

$\odot \ 0 \leftarrow \Leftrightarrow 2 \uparrow$

$\odot \ 1 \rightarrow \Leftrightarrow 3 \downarrow$

$k \leftarrow 2|y \ \odot \ (2| \text{ calculates the residue) if } y=1 \text{ it remains } 1; \text{ if } y=0 \text{ it remains } 0; \text{ if } y=2 \text{ it becomes } 0; \text{ if } y=3 \text{ it becomes } 1$

$:\text{If } y>1 \ \diamond \ x \leftarrow \bigcirc x \ \diamond \ :\text{EndIf}$

$z \leftarrow \uparrow k \text{ shift}_{2048} \text{bis} \ \downarrow x$

$\odot \ I \text{ turn the matrix into a vector of rows, I apply shift}_{2048}, \text{ then I make the matrix again}$

$:\text{If } y>1$

$\odot \ \text{vertical shift}$

$z \leftarrow \bigcirc z$

$\odot \ I \text{ rotate the matrix}$

$Z \leftarrow \{(-1 \downarrow \omega), -1 \uparrow \omega\} z$

$\odot \ I \text{ put the sum next to the matrix, in fact when the matrix rotates, even the sums rotate}$

$:\text{EndIf}$

$s \leftarrow +/, \supset \ ^{-1} \uparrow [2] z$

$\odot \ \text{sum of sums of all rows}$

$z \leftarrow (\subset \ ^{-1} \downarrow [2] z), s$

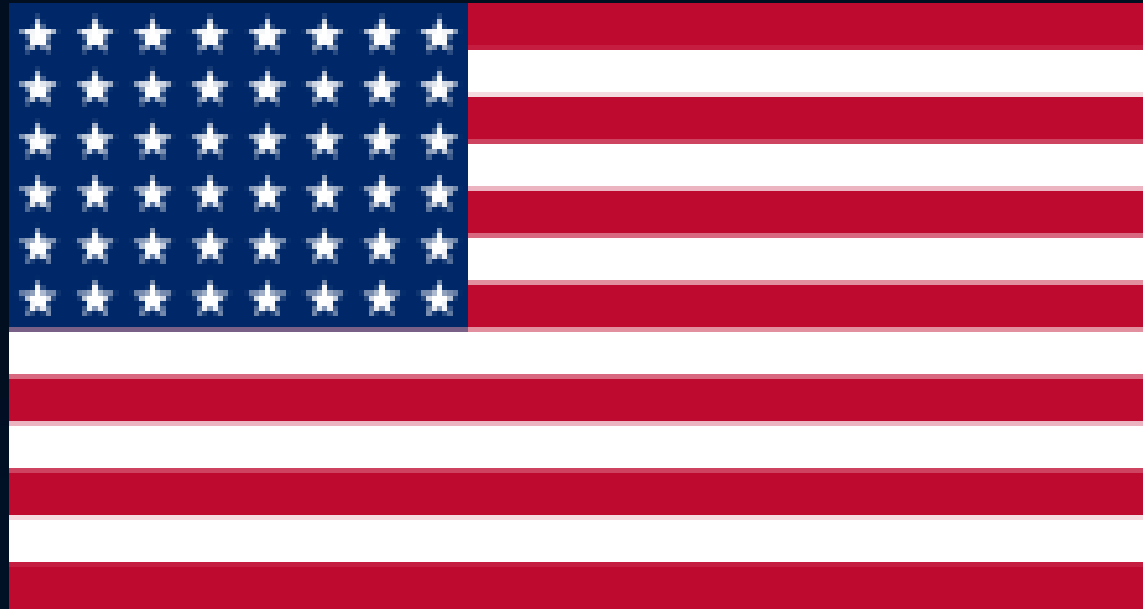
$\odot \ I \text{ put the sum next to the matrix and I remove the last column}$



# Arrange Stars

Applications Problem 3 (high difficulty)

Imagine that a new state is admitted to United States. How could the USA's flag change?



Given a number of stars, we have to choose a new armonious arrangement of the stars.

The problem is to understand what 'armonious' means.

To solve this problem we relied on the work of Skip Garibaldi, American mathematician, who defined all the possible patterns to arrange the stars.

Let's see them:

- 1) Wyoming pattern single;
- 2) Wyoming pattern double;
- 3) Short-long pattern;
- 4) Long-short pattern;
- 5) Alternate long-short;
- 6) Equal.

# WYOMING PATTERN SINGLE

With a single shorter line in the center



# WYOMING PATTERN DOUBLE

With two shorter lines in the center



# SHORT-LONG PATTERN

With short and long lines alternated



# LONG-SHORT PATTERN

With long and short lines alternated



EQUAL



Another important resource to solve this problem is the widget 'nextbigfuture' that allows you to have a graphical image of the modified flag.

This is useful, in particular, to control if the solution obtained is correct.



But let's see the details of the problem.

The task is to write a function that :

-takes a right argument integer singleton representing the number of stars to place

-returns a three items vector describing a 'reasonable' star arrangement where

- The first item is a vector of the number of stars in each row
- The second item is the horizontal spacing between columns of stars
- The third item is the vertical spacing in between rows of stars

We have to assume that the ratio of height to base is 1:1.4

```
#.Problems.arrangeStars 10
```

```
3 2 3 2 | 0.23333333333 0.2
```

Input: 10

Output: 3 2 3 2      number of stars in each row

0.23333333333 horizontal spacing

0.2.      vertical spacing

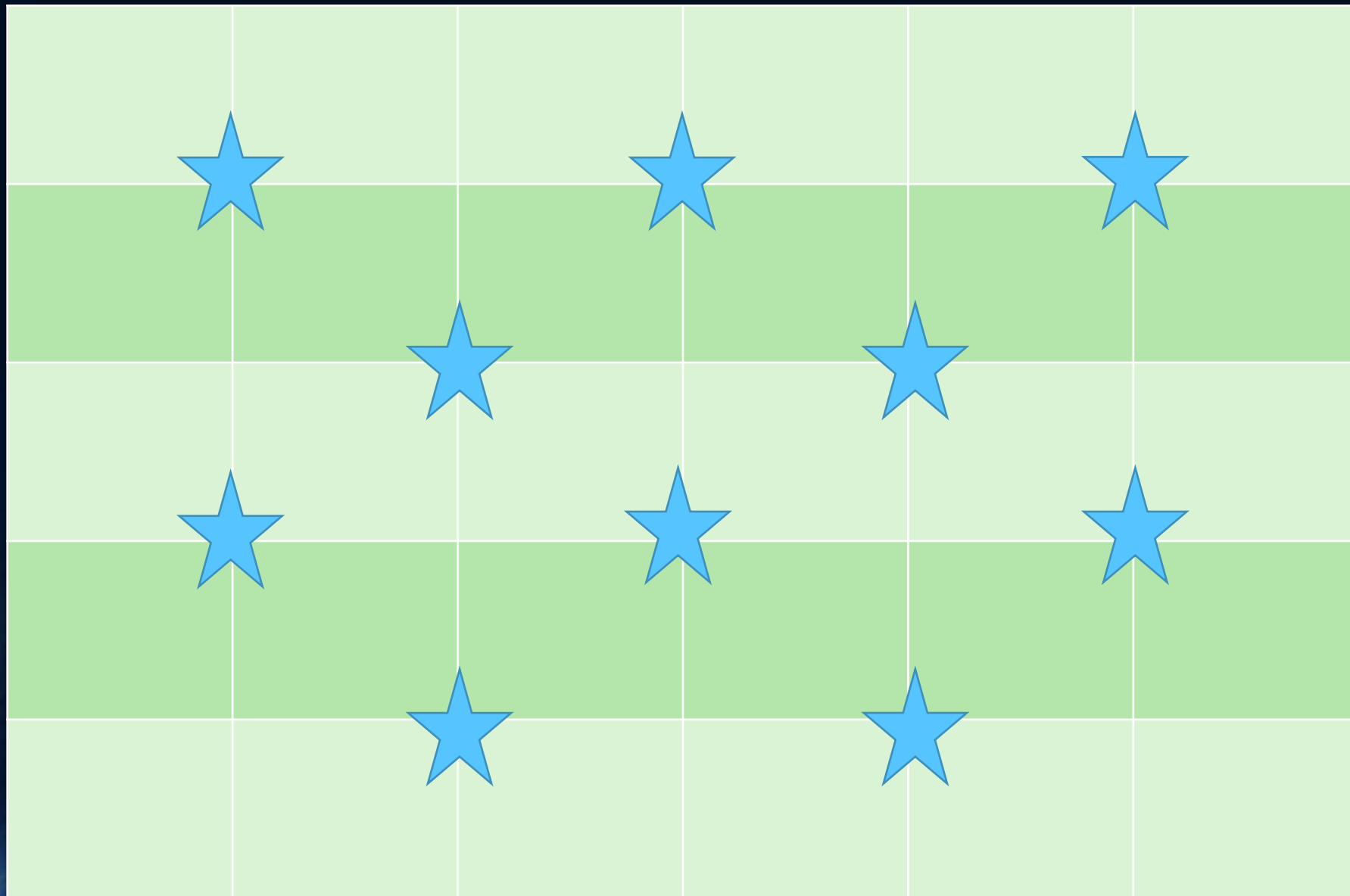
However, the second and the third elements of the output are derivable from the first one, and this can be easily understood by watching a drawing.



H H H H H H H

$1.4 = 6H$   
 $H = 1.4/6$

F  
F  
F  
F  
F



$1 = 5F$   $F = 1/5$

So, if my output is 3 2 3 2, the number of columns is  $c=3$  and the horizontal space is  $H=1.4/6=1.4/2c$ ; while the number of rows is  $r=4$  and the vertical space is  $F=1/5=1/r+1$ .

In this way we have found that  $H$  is always  $H=1.4/2c$ , and  $F$  is always  $F=1/r+1$ .

So the function of the alternated long.short pattern is:

$$ls \leftarrow \{(\underbrace{c(\alpha\rho\omega) - \alpha\rho 0}_1), (\underbrace{1.4 \div \omega \times 2}_2), (\underbrace{1 \div \alpha + 1}_3)\}$$

It gives me the first element (so the number of stars in each row) where  $\alpha\rho\omega$  is the long row and  $\alpha\rho\omega - 1$  is the short row.

It gives me the second element (so the horizontal spacing between columns of stars).

It gives me the third element (so the vertical spacing between rows of stars).

However our input is not a couple  $(r,c)$ , but a number, for example 22.

By starting from 22, we can only say that  $r$  and  $c$  must be between 1 and 22, and that  $r$  must be even, so it could be:

2 4 6 8 10 12 14 16 18 20 22 (11 possibilities)

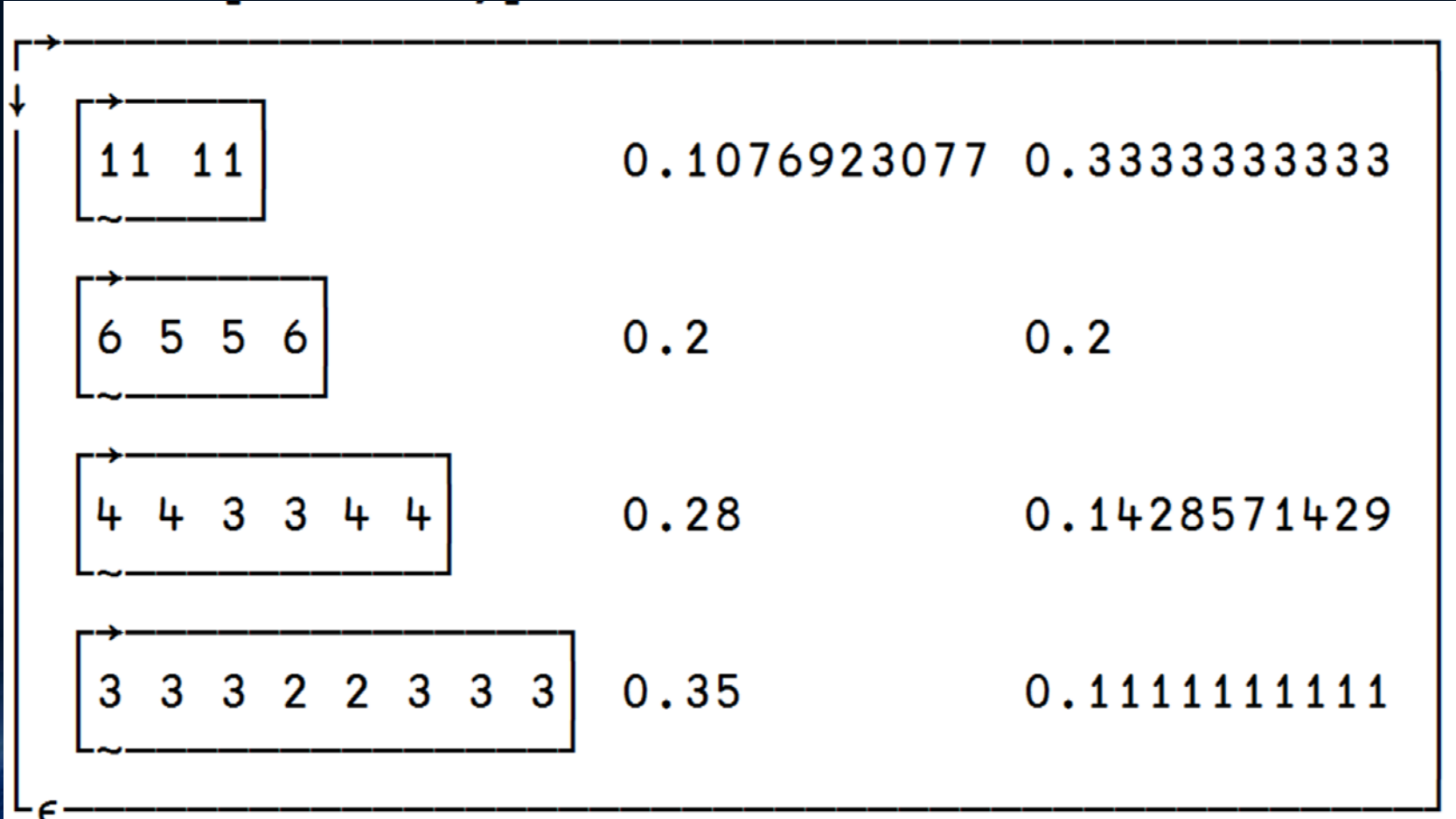
and  $c$  could be:

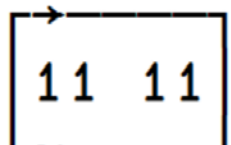
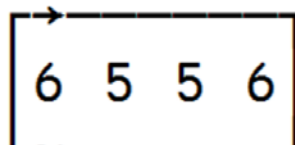
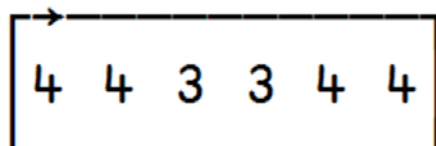
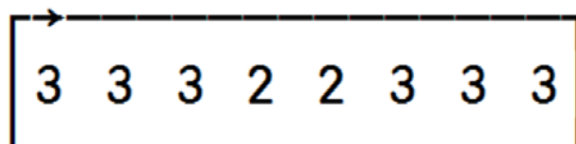
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 (22 possibilities)

We have to examine all the possible couple of  $r$  and  $c$  to reject those that are not correct ( for example  $(r,c)=(4,7)$ , because it would be 7 6 7 6; but the sum of stars is not 22) and in this particular example the only possible arrangement is 6 5 6 5.

However, we have considered only one pattern (alternate long-short) and so we have to repeat the reasoning for the other pattern and see if there are other possible arrangement.

In this case, there are 15 possible layouts, and these are some of them:



	0.1076923077	0.3333333333
	0.2	0.2
	0.28	0.1428571429
	0.35	0.1111111111

How can I decide which of them is the best?

The best is the one in which the difference between H and F is minimal, because in this way the stars will be bigger and the arrangement will be armoniuos.



# THE FINAL FUNCTION

$z \leftarrow \text{stars } x; e; ls; sl; als; ws; wd; reven; rodd$

$e \leftarrow \{(\subset \alpha \rho \omega), (1.4 \div \omega + 1), 1 \div \alpha + 1\} \circlearrowleft r \} c$

$ls \leftarrow \{(\subset (\alpha \rho \omega) - \alpha \rho 0 \ 1), (1.4 \div \omega \times 2), (1 \div \alpha + 1)\}$

$sl \leftarrow \{(\subset (\alpha \rho \omega) - \alpha \rho 1 \ 0), (1.4 \div \omega \times 2), (1 \div \alpha + 1)\}$

$\circlearrowleft als \leftarrow \{ \}$  it is a particular case of  $ls$  with even rows

$ws \leftarrow \{(\subset (\alpha \rho \omega) - \{ \{ \omega, 1, \omega \} \omega \rho 0 \} (\alpha - 1) \div 2), (1.4 \div \omega + 1), (1 \div \alpha + 1)\} \circlearrowleft \text{ odd rows}$

$wd \leftarrow \{(\subset (\alpha \rho \omega) - \{ \{ \omega, 1, 1, \omega \} \omega \rho 0 \} (\alpha - 2) \div 2), (1.4 \div \omega + 1), (1 \div \alpha)\} \circlearrowleft \text{ even rows}$

$reven \leftarrow (x \rho 0 \ 1) / \iota x$

$rodd \leftarrow (x \rho 1 \ 0) / \iota x$

$z \leftarrow, (\iota x) \circ . e(\iota x)$

$z, \leftarrow, (\iota x) \circ . ls(\iota x)$

$z, \leftarrow, (\iota x) \circ . sl(\iota x)$

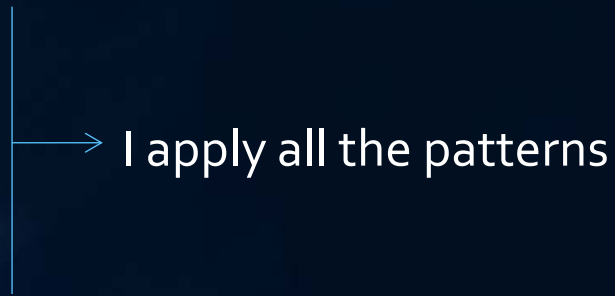
$z, \leftarrow, rodd \circ . ws(\iota x)$

$z, \leftarrow, reven \circ . wd(\iota x)$


$z \leftarrow \uparrow z$

$z \leftarrow (x = + / z[; 1]) \neq z \circlearrowleft$  I take all the patterns in which the sum of stars is equal to  $x$

$z \leftarrow \supset \downarrow (\{ \omega = [ / \omega \} ] - / z[; 2 \ 3]) \neq z \circlearrowleft$  I take the one in which the difference between horizontal and vertical space is minimum







The End

ANY QUESTIONS?