



Computerpower meets (beats) Intelligence

An APL Solution for Combination Problems

Gerald Dittrich

APL-GSE Meeting

ERGO Köln, April 16-17, 2018



The Problem : Lyndon x B = Johnson

- Combination problem discussed in 1971
- From 1963-22-11 to 1969-20-1 Lyndon Baynes Johnson was President of the United States of America
- How to solve the equation $LYNDON \times B = JOHNSON$?
- Solution is time consuming (not really difficult but boring) with lots of if-then decisions
- Solution found on the internet



- Solution found on the internet

- 10-08-2005, 07:22 PM

- ..LYNDON
.....x.B
JOHNSON

....._ _ _ _ _
.....
_ _ _ _ _

1--If B = 2....3....4....5.....6.....7....8....9
.....N can be..0...0,5...0..0,5.....0,2,4,8...0,5...0...0,5



- 2--If $N = 0$, then there is no carryover from BxD
 - 3--Then, $DONxB = SON$ stands alone.
 - 4--If $N = 0$, then O cannot = 0.
 - 5--If $B = 3, 5, 7, \text{ or } 9$, O must = 5
 - 6--Then
- | N..... | B..... | D..... | S |
|--------|--------|--------|------------|
| 0..... | 3..... | 1..... | 4.... |
| 0..... | 3..... | 2..... | 7.... |
| 0..... | 5.. | NG | as $O = 5$ |
| 0..... | 7..... | -..... | -..... |
| 0..... | 9..... | -..... | -..... |



■ 7--Two possible results

(1)..150...(2)..250

.....x.3.....x.3

.....450.....750

8--Then, $LY \times 3 = J5H$

9-- $J = 2$ max. = 2 as $D = 1$

10--In (1) 6, 7, 8, and 9 are available for L, Y, J, and H.

11--If $Y = 6$, $H = 8$ but L must be 8 also - NG

....If $Y = 7$, $H = 1$ but $D = 1$ - NG

....If $Y = 8$, $H = 4$ but $S = 4$ - NG

....If $Y = 9$, $H = 7$, $3L + 2$ cannot = 25 - NG

12--In (2) 1, 4, 6, 8, and 9 are available for L, Y, J, and H.



- 13--If $Y = 1$, $N = 3$ - NG
 -If $Y = 4$, $H = 2$ - NG
 -If $Y = 6$, $H = 8$ but L must = 8 to make $3L + 2 = 25$ - NG
 -If $Y = 8$, $H = 4$ and $3L + 2$ cannot = 25 - NG
 -If $Y = 9$, $H = 7$ and $3L + 2$ cannot = 25.
- 14--Therefore, B is not 3, 5, 7, or 9 and must be 6.
- 15-- N can therefore be 0, 2, 4, 6, or 8



■ 16--Then, for B = 6

N.....O.....D.....S

0.....2.....1.....7

0.....4.....1.....8

0.....8.....-.....-

2.....-.....-.....-

4.....-.....-.....-

8.....-.....-.....-

17--Two candidates fall out (1)..120.and (2)..140

.....x.6...x..6

.....720....840

18--In (1) 3, 4, 5, 8, and 9 are available.

19--J can be a max. of 5 and = 3, 4, or 5



- 20--This path leads to no solutions.
- 21--In (2) 2, 3, 5, 7, and 9 are available
- 22--J can be a max. of 5 and = 2, 3, or 5.
- 23--J = 2 and 5 lead nowhere.
- 24--For J = 3, we have LY
.....x..6
.....34H
- 25-- $340/6 = 56.6666$ so LY must be 57 or greater.
For LY = 57, $57 \times 6 = 342$ What do you know. Home free.

So we have 570140

.....x..6

.....3420840





Encode (\top) and (\perp) Decode: my favorite functions

- A pair of the most powerful functions in APL
- Unfortunately very rarely used by APL-programmers
- Reason 1: We do not need these functions (in reality: we do not know how that works)
- Reason 2: Most of the APL-programmers do not understand the power of these functions
- Reason 3: ?
- Allows you to generate all possible solutions of a problem
- Disadvantage: Needs a huge amount of workspace !
- Possibility to use loops when you get WS FULL (but ugly programming!)
- Need to eliminate generated solutions that do not behave like they should

Solving problems with encode and decode

- Method to generate results:
- Generate all possible solutions
- Reduce the starting basis to reasonable combinations
- Eliminate solutions that do not match your conditions
- Reduce possible solutions by logic reasons
- Try do avoid loops greater than 1000 (very time consuming)
- do not hesitate to make tiny loops (less than 100)
- Reduce amount of work in the inner most loop to be as small as possible
- Try to avoid outer product (use only for small vectors !)
- Replace outer product by other powerful functions if possible

Problems that can be solved by encode and decode

- Use a 2-dimensional matrix instead of a high dimensional matrix with lots of zeros to reduce the workspace required (avoids WORKSPACE FULL!)
- however required programs to use groupsumming in 2-dimensional matrices by keys instead of summing over axis
- Generate all permutations of a given vector and reduce results to logic restraints (the ASPIRIN problem)
- Use a idioms-generator to generate new idioms (this is just for fun!) (my speech from 2003: Hunting Crazy Idioms)
- Solve the LYNDON x B = JOHNSON problem with an APL-program or similar problems like for example MORTEN x ? = KROMBERG without dealing with annoying combination problems (I really hate if- then questions! I assume that my brain is not fitted for difficult questions – or does not like them)

Problems in examples

- The aspirin problem (no problem for today's computers)
- Playing Gomoku (5 points horizontal, vertical and diagonal)
- The Lyndon x Baines = Johnson problem (can be solved in a few minutes on a Intel 7-based computer)
- The phrase palindrom

S	A	T	O	R
A	R	E	P	O
T	E	N	E	T
O	P	E	R	A
R	O	T	A	S



The ASPIRIN Problem

- The ASPIRIN problem (no problem for today's computers)
- The number of combinations to produce is 823543 (7^7)
- Can be reduced to 279936 by eliminating columns containing 7
- Can be reduced to 2520 by deleting most of the lines that do not meet what we want (e.g. the vocal "I" should occur twice, the consonant "P" should only occur once etc.)



APL-Program to generate all Permutations of ASPIRIN (quick but not really dirty and working!)

```
ERG←ASPI STR;A;I;M;MA;R;R1;RED;STR1;Z
⎕Ⓞ DPC/GD,24.12.2017
⎕Ⓟ GENERATE ALL PERMUTATIONS OF A GIVEN STRING STR
⎕Ⓡ A2← ASPI A1
⎕Ⓡ
⎕Ⓡ
⎕Ⓡ
⎕Ⓡ APL2 VERSION
⎕Ⓡ
⎕Ⓡ e1 STR←'ASPIRIN'

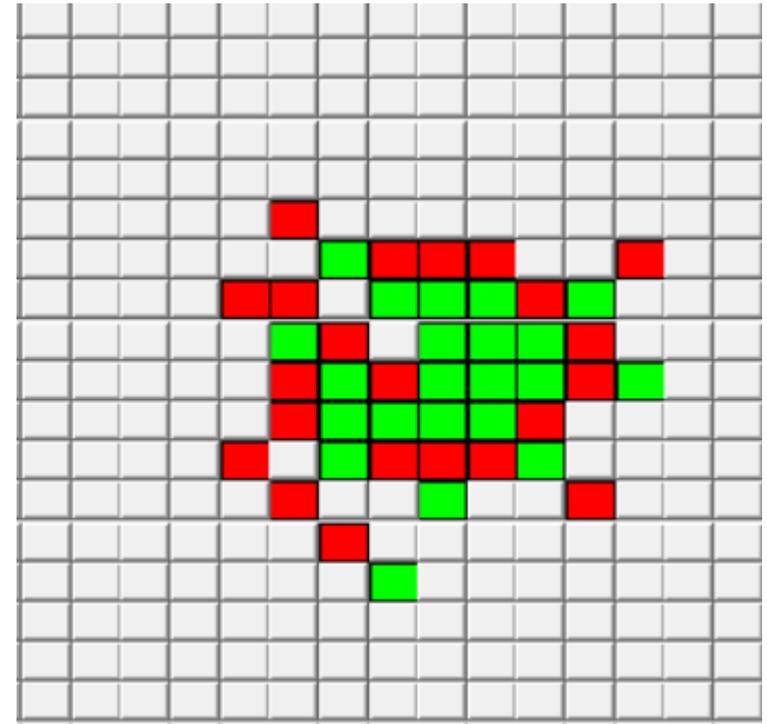
R←pSTR

R1←pSTR1←((STR⊖STR)=⊖pSTR)/STR ⎕ ALL DIFFERENT OCCURENCES OF CHARACTERS IN STRING
Z←0,~1⊖⊖R*R
M←1+(RⓅ)⊖Z
A←+/STRⓄ.=STR1 ⎕ NUMBER OF OCCURENCES OF A CHARACTER IN STRING
M←(~v/M>pA)/M ⎕ REDUCE M TO MAX DIFFERENT CHARACTERS
MA←STR1[M] ⎕ INDEXING VECTOR STR1 BY MATRIX M
I←0
LIE:→LIE×⊖R1<I←I+1
RED←A[I]=+/MA=STR1[I] ⎕ REDUCE TO NUMBER OF OCCURENCES OF A CHARACTER IN STRING
MA←RED/MA
→LI
LIE:
MA←ⓄMA
MA←MA[ⓄAVⓄMA;] ⎕ SORT MA ASCENDING
ERG←MA
```

Playing Gomoku

- Solution: numbering unused fields with 0, computer uses fields with 1, player uses fields with 2 and boarder fields with 3
- Decode strings of length 6 to a system with basis 4
- Lookup all unused fields on a table with strategic values
- Sum up all strategic values for all unused fields
- Computer moves to the field with highest combined strategic values
- No loops necessary: just using decode and dyadic iota
- Example shown on next page for a 15x15 matrix

Example of playing Gomoku with APL2000





Solving LYNDON x B = JOHNSON

- Idea to solve this problem:
- Generate all numbers from 1 to 999999 (M1, corresponds to ,LYNDON')
- Multiply by factors 2 to 9 (factor)
- Keep result (product M2: normally numbers with 6 to 7 digits, corresponds to ,JOHNSON')
- Use encode to basis 10 for generated numbers and product to get digits
- In this case a 6 x 999999 matrix(M1) and a 7 x 999999 matrix(M2)
- Catenate M1,factor, and M2 (M1,[1]factor,[1]M2) (matrix M)
- Guarantee that all characters match the same digit (Forename,factor,Familyname) compared with M



Still $LYNDON \times B = JOHNSON$

- If you are lucky you get remaining columns:
- In this case you get the result:

$$\begin{array}{l} | LYNDON \times B = JOHNSON \\ | 570140 \times 6 = 3420840 \end{array}$$

MORTEN x ? = KROMBERG

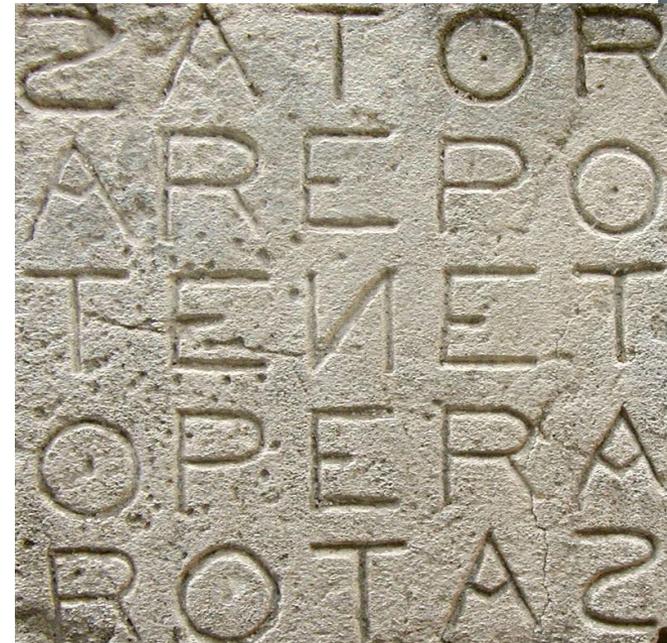
- Another example:
- $MORTEN \times ? = KROMBERG$
- Normally, the length of the family name should be equal to or more than the length of the first name
- However there exists an exception as you can see
- Only one possible solution if $? = B$ and $K=0$

■ Result:

$$\begin{array}{r}
 \mathbf{MORTEN \times B = KROMBERG} \\
 \mathbf{493517 \times 8 = 03948136}
 \end{array}$$

Palindrom: the history

Palindromes date back at least to 79 AD, as a palindrome was found as a graffito at [Herculaneum](#), a city buried by ash in that year. This palindrome, called the [Sator Square](#), consists of a sentence written in Latin: "[Sator Arepo Tenet Opera Rotas](#)" ("The sower Arepo holds with effort the wheels"). It is remarkable for the fact that the first letters of each word form the first word, the second letters form the second word, and so on. Hence, it can be arranged into a [word square](#) that reads in four different ways: horizontally or vertically from either top left to bottom right or bottom right to top left. As such, they can be referred to as palindromatic.^[from Wikipedia.org]



The Phrase Palindrom

- The phrase palindrom
- The greek verb palindromos means running backwards
- (Παλίνδρομος *palíndromos*)

S	A	T	O	R
A	R	E	P	O
T	E	N	E	T
O	P	E	R	A
R	O	T	A	S

- This leads to a tremendous amount of computer operations ($n=11881376$ for each column) (26 power 5)
- leads to $n \times (n-1) \times (n-2) \times (n-3) \times (n-4)$ ca. $2.36E35$ operations !!
- can be reduced to some $4.5E29$ operations by avoiding senseless combinations, but still far to high to be solved



Possible reductions for the phrase palindrom

- Let's take a closer look to the phrase palindrom:
- You can take advantage from the fact that once you have chosen five characters for the first line, the second character of these five characters will reduce the combinations for the second line to all combinations starting with the second character of the first line (reduction from 600000 to some numbers between 500-3000)
- This gives the possible combinations of the inner circle of the square
- The innermost point can be from A-Z (but no need to look at it during generation of palindromes)
- These ideas will reduce the number of possible solutions (and loops you have to go) to about $600000 \times 2000 = 1.2 \times E9$
- Still 1.2 billions of operations to perform !!!! WOW !!!!! What an enormous challenge !!!

Finding the SATOR phrase palindrom

- my Intel i7 needed about 3 hours to generate 10 % of the phrase palindromes (24 GB of workspace needed in Dyalog APL-64)
- It would take 30 hours to produce all phrase palindromes (not to bad!)
- The SATOR Palindrom was found after 96.5 million iterations when I started in the interval between 65 % to 75 %
- The produced Palindromes are saved in a nested vector.
- Before choosing a character for the innerst point, you can look at other lines to determine if a result would make sense



Using a List of 5-character Semi-Palindromes (Semordnilap)

- The real challenge is not to produce the phrase palindromes, but to evaluate if the generated palindromes make sense
- Semordnilaps are words that can be read backwards even if the meaning is different (e.g. SPOOL LOOPS, LEBEN NEBEL, SMART TRAMS)
- Using only semordnilaps will reduce the number of words to some 100
- Semordnilaps guarantee that produced palindroms can be read
- But: the list of known semordnilaps would NEVER produce the solution for our phrase palindrom!!
- The reason: the name of the SATOR : 'AREPO' is artificial and of course not in the list of Semordnilaps

Using multiprocessor capabilities

- Intel i5 has 4 cpus (2 virtual) (up to 4 GHz)
- Intel i7 has 8 cpus (4 virtual) (up to 4 GHz)
- Intel i9 has 10-16 cpus (the same virtual) (up to 4 GHz)
- Distribution possible if problem can be partitioned:
- Distribute workload to available cpus (do not use virtual cpus : that may slow down computer speed due to swapping and partage memory!)
- Distribute workload to available (internet) computers
- Distribute workload to a cascade of computers
- Ideas for more possibilities ?



This is the end!

- I thank you very much for listening to me and
- working with me in this tutorial !