MULTI-STRAND NON REGULAR KNOT

(using a THK cordage route but multi-strand – GCD >1) :

how to built their BIGHT SEQUENCE AND BIGHT ALGORITHM.

(the hard core of the method is ***THE BRAIDER*** ; that is mainly SCHAAKE – adaptation is mine)

We have seen (my web page Turkshead_12) the BIGHT ALGORITHM FOR A 'TRUE' THK (in fact this goes for all coding on a THK –standard regular knot– cordage route or shadow knot.
Those were SINGLE STRAND knots

Now we are going to look at FALSE THK : MULTIPLE STRANDS (semi-regular knots for Schaake).

STANDARD PINEAPPLE knots are in fact assemblies of true THK albeit using several 'BIGHT BOUNDARY' on each side. Those THK COMPONENTS are distributed in one or two SETS with the same number of BIGHT on all components of a given set (that will form BIGHT-NEST), and ODD number of LEAD.
Number of LEAD is different in the two SETS which are separated by 2 LEADS.

Multi-strands 'FALSE' THK use a single BIGHT BOUDARY on each side so there is no nested BIGHT.
Just as Herringbone Knot DO NOT have nested bights.
All the components are identical in number of BIGHT and NUMBER of LEAD.
Each of the THK COMPONENTS obey the (GDC of L & B = 1) rule but the whole knot does not and the GDC of its full complement of LEAD and BIGHT is the number of STRAND needed to make it.

Let us look at the **DIFFERENT WAYS THE STRANDS MAY BE DISTRIBUTED** along a **BIGHT RIM (bight boundary for SCHAAKE).**

This distribution of STRANDS may be **REGULARLY PERIODIC** (see ANNEXE)

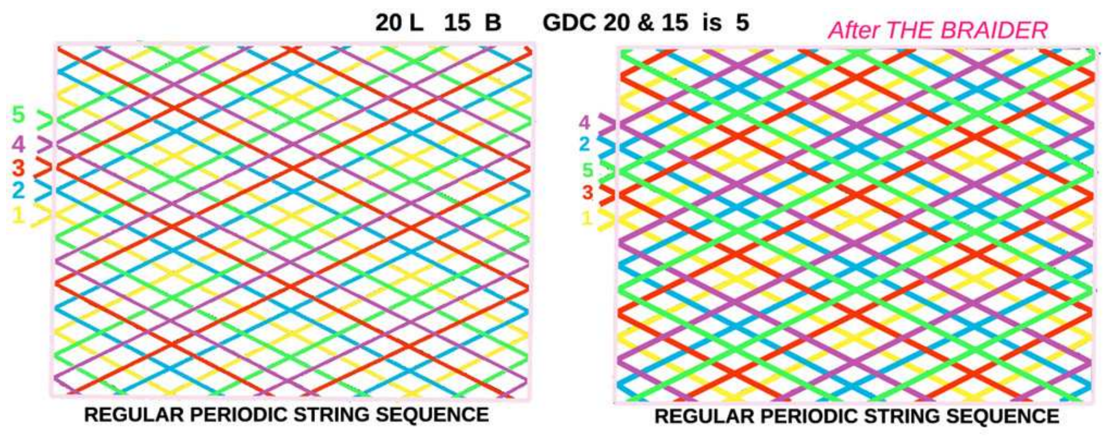

20 L 15 B GDC 20 & 15 is 5
After THE BRAIDER
5
4
3
2
1
4
2
5
3
1
REGULAR PERIODIC STRING SEQUENCE
REGULAR PERIODIC STRING SEQUENCE

FIG 1

```
1                       1  2                      1  2  3                   1  2  3  4
x  x  x  x  x   step 1  x  x  x  x  x   step 1    x  x  x  x  x   step 1    x  x  x  x  x   step 1

1  2  3  4 5
x  x  x  x  x    if you go on, ALL THE WHILE thinking circular and using modulo, then →

1  2  3  4 5  6  7  8 9 10 11 12 13 14 15 16 17 18 19 20
x  x  x  x  x  x  x  x  x  x  x   x  x   x  x  x  x   x  x  x
```

if using [MODULO (number of STRANDS)] ,(here number of STRANDS ==5) becomes :

```
            0              0             0            0
1  2  3  4 5  1  2  3 4  5  1 2 3 4 5 1 2 3 4 5
x  x  x  x  x  x  x  x  x  x  x x x  x x x x  x x x
```

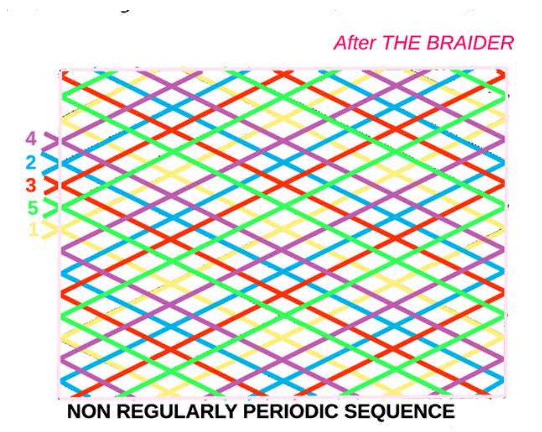
The right side diagram in ***FIG 1*** : **1 3 5 2 4** **HAS** a REGULAR PERIOD despite what you may intuitively believe.
1 + step 2 = 3 3 + step 2 = 5…….
1 *step 2* (one bight-step) **3** *step 2* **5** *step 2* **7** *step 2* **9**

Do you get the regular periodicity now you know it exists ?
No !
Then you are forgetting to think circular / modular ; just apply [MODULO Number of STRAND] (5 in this case) and you get :

1 *step 2* **3** *step 2* **5** *step 2* **2** *step 2* **4**

So only **ONE** BIGHT SEQUENCE leading to only **ONE** BIGHT ALGORITHM is necessary. This algorithm will be applied successively, periodically to each STRAND.

BUT it can also be **NOT REGULARLY PERIODIC** :



1 5 3 2 4 whatever you do is NOT periodic

Just try :
1 *step 4* (one bight step) **5** *step 3* (one bight step) **3** *step 4* (one bight step) **2**
2 *step 2* (one bight step) **4**

Still it is easy to realise that there may exist **SUB**-periods and go searching for them :

1 *step 1* **2** one **sub**-period : there are 2 BIGHT between BIGHT 1 and BIGHT 2 making 3 bight steps.

5 NO bight in-between and step minus 2 **3** ONE BIGHT in-between and step +1 **4**

FIG 2
no sub-period ? Are YOU SURE OF THAT ?
May be circular / modular was plainly forgotten ?

imagine **5 3 4** distributed clockwise on the perimeter of a circle (just as they are in fact on the circular BIGHT BOUNDARY) then you will be able to «commence the running of the round track » anywhere you want.
Start at **3** then you can read **3 4 5** so one **sub**-period with step 1 does exist (2 bight steps) .

That is the moment a bright brain will : *"OH NO ! that does not work ! look :*
OK between ***3*** *and* ***4*** *there is one BIGHT (Bight number* ***2*** *) but between* ***4*** *and* ***5*** *there are 11 BIGHTS "*
Yes indeed.
So what ?
I insist it is *step 1* (2 bight-steps)
Once again the ***MODULO*** *(here 5)was lost from sight.*

Looking at it like that : *1* **5** **3** *2* **4** *x x x x x x x x x x x* **5** you will believe your interjection is right.Yet it is wrong !
You have to look at it so :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 *1 2* *BIGHTS going round and round…*
1 ***5*** ***3*** ***2*** ***4*** ***x x x x x x x x x x*** *1* 5 the two highlighted ***5*** 5 are the same **5** just as ***1*** *1* are the same **1** (circularity == perimeter , the « closing of the path" = circuit , the 're-entering' of the sequence)

1 **5** ***3*** *2* ***4*** *1* ***5*** ***3*** *2* ***4*** *1* ***5*** ***3*** *2* ***4*** *1* **5**   the step between each component of **3   4   5** is indeed *step 1* (2 'bight-steps' of one step)

2 bight steps, there is 1 BIGHT between any two of them considered as « **ORD**INAL », in their numerical **ORD**ER or RANK : 3, 4 , 5 .   So **3   4   5** is another sub-period.

We now have **2** sub-periods sequences
1 2   and   **3   4   5**   …….   so we need to built **2** 'BIGHT SEQUENCE' and **2** 'BIGHT ALGORITHM'

BUILDING THE BIGHT SEQUENCE.

Knots in ***FIG 1*** are   **20 LEAD 15 BIGHT   5 STRAND**

GDC of 20 (number of LEADS) and 15 (number of BIGHTS) is **5**
20 = 2 * 2 * **5** (factoring with prime factors)
15 = 3 * **5**
FIVE SINGLE STRAND COMPONENTS THAT WILL BE EACH 4 LEAD 3 BIGHT

To get the BIGHT SEQUENCE we need to provide for each BIGHT hence : '**B'** signs '@' IN LINE
(may be we would have done better putting them in a circle : chariots defence style !)

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1   2   3                                           B-3 B-2 B-1  B
@   @   @   @   @   @   @   @   @   @     @    @    @    @    @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

That would be OK as it is, for a single strand knot that is, but there are FIVE different STRANDS in this case. We better keep that in mind. Each single strand making on each bight rim (**B / GDC**) or (**B / Number of Strand** (NS)) == 15/5 = 3 BIGHT PERIODICALLY DISPOSED :
It is now time for an important note and another bout of my expounding on the absolute necessity of never confusing logical planes under penalty of being lost and some times even plain stupid.
Here two logical planes are at play : SPACE & TIME.
FIG 1 and ***FIG 2*** are showing the **SPACE distribution** of the components knots and theirs BIGHTS.
The illustration here under, a BIGHT SEQUENCE, show the **TIME distribution** of the STRANDS and BIGHTS. If you confuse one representation with the other you will be lost in deep dark woods !

```
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
```

Lets us do the **20L 15B** periodic repartition that is in the left side diagram of ***FIG 1***
First find the « step » : it is still **(-L) modulo B**   so (-20) mod 15 = 10
The writing of the first STRAND @ will be

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
    1   2   3   4   5   6   7   8   9   10
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0                                       1
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
5   6   7   8   9   10                      1   2   3   4
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0                   2                   1
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Now we need to distinguish between the different STRANDS, so the first one having « used up » the DIGITS we now have to go « LETTERS » :
second STRAND is denoted **A** , third STRAND is **B**, fourth STRAND is **C** and so on…

How can we now dispose the BIGHT SEQUENCE WITH THE SECOND STRAND '**A**' ?

Err…… Well….err……

Here is what is written in *THE BRAIDER* Volume **I** Issue N° 8 PAGE 174

[open quote]
…/ first we lay down string #1 ; for this string the above complementary cyclic night-number scheme applies. Then we lay down string #2, but since string #1 already occupies the bight-points immediately ***below*** those which string #2 is going to occupy, we mark the bight-points immediately to the ***left*** (in cyclic fashion of course) of those who possess numerical value with **A.** Hence for string #2 we obtain the complementary cyclic scheme !

0 A 2 A 1 A
∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

Now we lay string#3, but since strings #1 and #2 already occupy the two consecutive bight points immediately **below** those which string #3 is going to occupy, we mark the bight-points immediately to the **left** (in cyclic fashion of course) of those who possess the letter A with B. Hence for string #3 we obtain the complementary cyclic bight-number scheme :

0 B A 2 B A 1 B A
∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

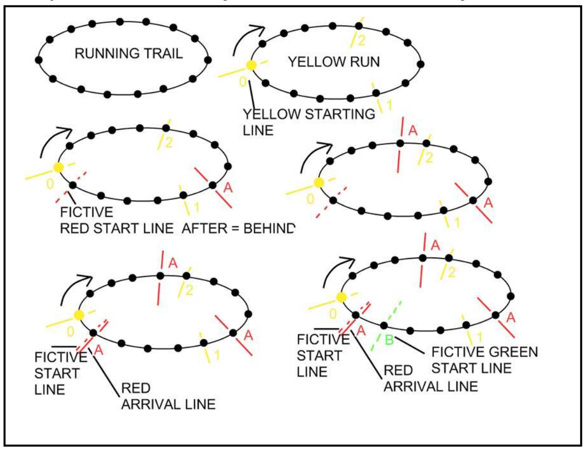
/…
[end quote]

As I don't think that everyone will get the above first time around **here is my own explanation.**

0 **2** **1**
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @

In fact you should not accept to see there a STRAIGHT LINE but chose to see a CURVED LINE MAKING A FULL CIRCLE :

Imagine you take the « bar » of @ with one hand on each extremity then you bend it downward on your knee, making a circle (mind experience only !) like the bights rim does.
Keep that circularity in mind each time you find difficulty in understanding the explanations.



Consider a running track :
first 'runner' (runner==strand) to start will have yellow O line as starting line, then will run for **(-L)**modulo **B**, that will be till mark 1, then will run again for
(-L)modulo **B** (mark 2), then will run again for
(-L)modulo **B** which bring it back at 0 ; the start line is also its arrival line 'closing the circuit'.

What cannot be seen on the straight linear representation is that the second runner (strand) is starting **after, in time,** so is **behind** (relative to the direction of running) ,**in space, to represent it being 'behind' in time.**

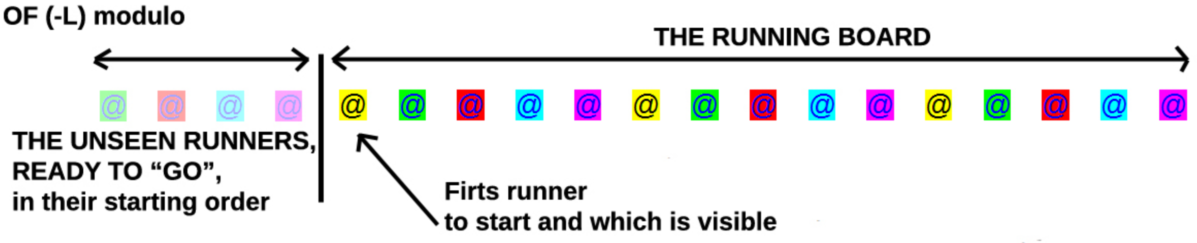
The segmented run made by STRAND #2 or **A** will be exactly identical in segment length, (-**L**)modulo B**,** and be marked by **A**.

Then the third runner (strand **B**) will be placed on the running track just behind the last runner **A** : that will be **B** fictive start line …..

It happen that the "behind" is worth in this case ONE Bight position, in some other cases it will be worth, TWO, or THREE, or **NS – 1** (Number of Strand less one).

I feel that it is easier to grasp **Later = Behind** than **Below / Left** but it is essentially identical.
You may also choose to look at the situation as in the following illustration

ALL CONCURRENT MUST RUN SUCCESSIVE SEGMENTS OF (-L) modulo


THE RUNNING BOARD
@ @ @ @
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @
THE UNSEEN RUNNERS, READY TO "GO", in their starting order
Firts runner to start and which is visible

The writing goes on like that :

```
-------------------------------------------------------------
0               A   2               A   1               A
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
-------------------------------------------------------------
0           B   A   2           B   A   1           B   A
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
-------------------------------------------------------------
0       C   B   A   2       C   B   A   1       C   B   A
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
-------------------------------------------------------------
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
-------------------------------------------------------------
```

We now have the Complementary Bight Sequence.

We have **20 L** so we need to provide for each of them by writing **(L + 1)** signs (here signs are #)

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
1   2   3   4                                                       L-2 L-1 L   L+1
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

You will remember from the standard THK study that we need to 'differentiate' the BIGHT RIM (border, boundary….) with for example : # **left** BIGHT RIM   # **right** BIGHT RIM

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

the **(L – 2)** # signs stand for the "column" where the crossings happen in this COLUMN-CODED knot.

We have the **complementary**

```
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A
```

we get the **cyclic** just by "reversing " the complementary

```
A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
```

then we write the BIGHT ALGORITHM

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
                        A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

ALL columns are NOT yet covered so we just go CIRCULAR / PERIODIC / CYCLIC

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A   0   D   C   B   A   Φ
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
Φ   A   B   C   D   0   A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Note that there is an empty slot Φ above the RIGHT side bight rim and under the LEFT side bight rim.
The reason is quite evident : there is no crossing on a BIGHT RIM.

There is a special way to read all that : Remember ?

```
CONCERN THE ODD NUMBERED HALF-PERIOD
TO BE READ FROM LEFT TO RIGHT
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A   0   D   C   B   A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
    A   B   C   D   0   A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
                                              TO BE READ FROM  RIGHT TO LEFT
                                   CONCERN THE EVEN NUMBERED HALF-PERIOD
```

Add now the appropriate code signs in place of the # signs as in the following example :

```
CONCERN THE ODD NUMBERED HALF-PERIODS  going low left to up right (mandrel)
TO BE READ FROM LEFT TO RIGHT
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A   0   D   C   B   A
#   /   /   /   \   /   /   \   \   /   /   \   \   /   /   \   \   /   /   /   #
    A   B   C   D   0   A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
                                              TO BE READ FROM  RIGHT TO LEFT
CONCERN THE EVEN NUMBERED HALF-PERIODS going low right to up left (mandrel)
```

/ is seen by an **ODD** numbered HP running from low **left** to up **right** this is an **OVER**
/ is seen by an **EVEN** numbered HP running from low **right** to up **left** this is an **UNDER**

\ is seen by an **ODD** numbered HP running from low **left** to up **right** this is an **UNDER**
\ is seen by an **EVEN** numbered HP running from low **right** to up **left** this is an **OVER**

so we could also chose to write :

```
CONCERN THE ODD NUMBERED HALF-PERIODS going low left to up right (mandrel)
TO BE READ FROM LEFT TO RIGHT
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A   0   D   C   B   A   (¶)
     O   O   O  U   U    O   O  U   U    O   O  U   U    O   O  U   O    O   O
#   /   /   /   \   \   /   /   \   \   /   /   \   \   /   /   \   /   /   /   #
    U   U   U   O   U   U   O   O   U   U   O   O   U   U   U   O   U   U   U
(¶) A   B   C   D   0   A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
                                              TO BE READ FROM  RIGHT TO LEFT
 CONCERN THE EVEN NUMBERED HALF-PERIODS going low right to up left (mandrel)
```

---

which can even be written as (that is what I prefer and what I do, as it is IMO quite superior to the / and \ in ease of use and lower mistake-rate : H(igh) is Over L(ow) is Under just because H and L are less prone to mistakes than u and o or U and O)

CONCERN THE **ODD** NUMBERED **HALF-PERIODS** going **low** left to **up** right (mandrel)
TO BE **READ** FROM **LEFT** TO **RIGHT**

```
0   D   C   B   A   2   D   C   B   A   1   D   C   B   A   0   D   C   B   A   (¶)
    H   H   H   L   L   H   H   L   L   H   H   L   L   H   H   L   H   H   H
#   xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx   #
    L   L   L   H   H   L   L   H   H   L   L   H   H   L   L   H   L   L   L
(¶) A   B   C   D   0   A   B   C   D   1   A   B   C   D   2   A   B   C   D   0
```

TO BE **READ** FROM **RIGHxT** TO **LEFT**
CONCERN THE **EVEN** NUMBERED **HALF-PERIODS** going **low** right to **up** left **(mandrel)**

It "*just happened so*" that the line of code # / / \ ….. \ / / #
is COMMON to EVEN and ODD HP **but it may happen that you need one line of code for each sort of even / odd HP.**

The making of different algorithms is needed for **COLUMN** CODED knots that have their STRANDS in a **NON** REGULARLY PERIODIC disposition or for ***NEITHER*** COLUMN-CODED ***NOR*** ROW-CODED knots.

ALWAYS DISTINGUISH BETWEEN LOGICAL PLANES :
Here the one of THE CORDAGE ROUTE
and the one of the CODING that is just 'put on' the cordage route.
With the same cordage route may come several different coding

Examples taken from ***THE BRAIDER*** *Volume **I** issue N° 9*
On a **25 L 15 B** cordage route you may have (a few of the possible !)

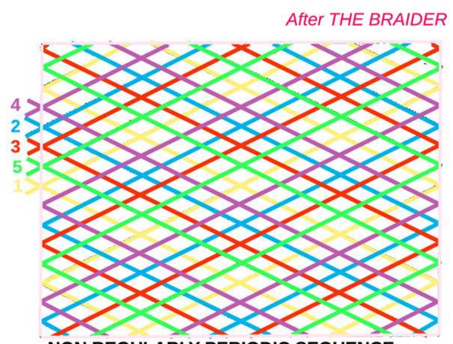
[open quote]
/ / \ \ / / \ \ / / \ \ / / \ \ / / \ \ / / \ \ …. or …. / / / \ \ \ / / / \ \ \ / / / \ \ \ / / / \ \ \... or… / / / / \ \ \ \ / / / / \ \ \ \ / / / / \ \ \ \….or….. / / \ \ \ \ / / / / \ \ \ \ / / / / \ \ \ \ / / /
[end quote]

Now, just to be more knowledgeable thanks to **SCHAAKE** and ***THE BRAIDER*** pioneering work allowing us to be "dwarves on the giant's shoulder " we will see the cases of

- non regularly periodic **column-coded** semi-regular knot
- **neither column-coded nor row-coded** semi-regular knot

PLEASE ATTACH YOUR SEAT BELT AND EXTINGUISH CIGARETTE
AND TURN OFF ANY PORTABLE THEY BE PHONE OR PC **….;-)**

THE CASE OF THE NON REGULARLY PERIODIC COLUMN-CODED



NON REGULARLY PERIODIC SEQUENCE

Lets us go back to ***FIG 2***

You will remember that we found that there were TWO sub-periods **1 2** and **3 4 5** hence 2 'Bight Sequence' and 2 'Bight Algorithm' to make

One COMMON to STRAND #1 and STRAND #2

The other one, different from the first one is COMMON to STRANDS #3 , #4 and #5
20L 15 B it imply (GDC = 5)
(-L)modulo B == (-20)modulo 15 = 10

```
****************FOR STRAND # 1 and STRAND # 2**************************************************
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Note that now the “lateness” of A on DIGIT is not equal to ONE but to THREE BIGHTS

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
@     @     @     @     @     @     @     @     @     @     @     @     @     @     @
      1     2     3     4     5     6     7     8     9     10

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0                             2                             1
@     @     @     @     @     @     @     @     @     @     @     @     @     @     @
5     6     7     8     9     10                                  1     2     3     4
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0                             2           A                 1
@     @     @     @     @     @     @     @     @     @     @     @     @     @     @
                                          -3    -2    -1
8     9     10                                  1     2     3     4     5     6     7

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           A                 2           A                 1
@     @     @     @     @     @     @     @     @     @     @     @     @     @     @
                                          -3    -2    -1
8     9     10                                  1     2     3     4     5     6     7

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           A                 2           A                 1           A
@     @     @     @     @     @     @     @     @     @     @     @     @     @     @
                                          -3    -2    -1
                  1     2     3     4     5     6     7     8     9     10
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

We now have the complementary bight sequence for STRANDS **#1** and **#2**

It is easy to get the **BIGHT ALGORITHM** now

```
-                                                                                  -
0       A           2       A           1       A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #

-                                                                                  -

0       A           2       A           1       A           0       A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #

-                                                                                  -
0       A           2       A           1       A           0       A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
            A       0           A       1           A       2               A   0
-                                                                                  -
```

```
-----------------------------------------------------------------------------------------------
```

For ODD numbered HP read left to right

```
0       A           2       A           1       A           0       A
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #

            A       0           A       1           A       2               A   0
```

For EVEN numbered HP read right to left

```
-----------------------------------------------------------------------------------------------
```

Just replace the # signs by CODE signs and work out the code for each Half-Period for STRAND **#1** and STRAND **#2**

****************FOR STRAND # 3, STRAND # 4 and STRAND # 5**

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
@   @   @   @   @   @   @   @  @   @   @   @  @  @   @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Note that
the "lateness" of A on 0 is not equal to ONE but to FOUR BIGHT
the "lateness" of B on 0 that was equal to TWO is still (happenstance) of TWO BIGHT
the "lateness" of C on 0 is not equal to THREE but to ONE
the "lateness" of D on 0 is not equal to FOUR but to THREE BIGHT

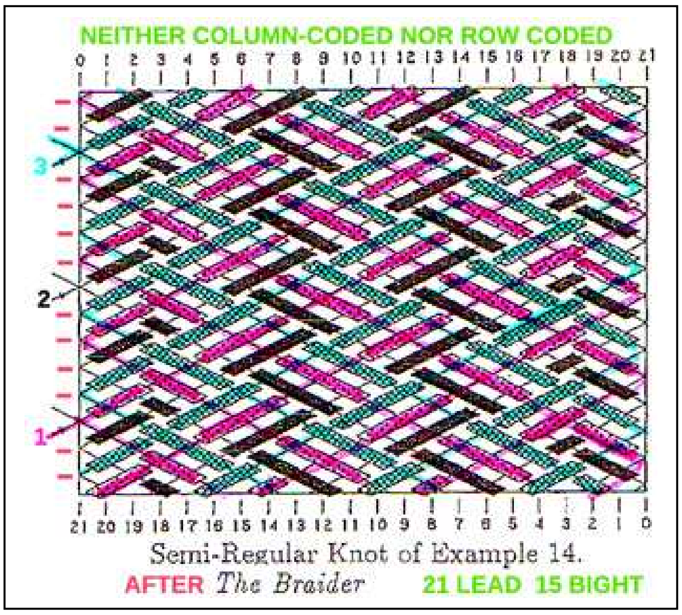
so you get the complementary Bight sequence for STRAND #3, #4 and #5

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                       0   A   D   B   C   2   A   D   B   C   1   A   D   B   C
   -4  -3  -2  -1      @   @   @   @   @   @   @   @   @   @   @   @   @   @   @
    A   D   B   C   ( waiting behind the scene in "starting order" )
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

I leave you working the rest by yourself as you had more than enough explanation to do it alone.

Look in the help note or the annexe if necessary

THE CASE OF THE *NEITHER* COLUMN-CODED *NOR* ROW-CODED



It is NOT that there is no REGULARITY in the PERIODICITY **1 2 3** but the point is that
THE CODING IS DIFFERENT FOR EACH ONE OF THE THREE STRANDS

this IMPLY :

THREE 'BIGHT SEQUENCE'
AND
THREE 'BIGHT ALGORITHM'
ONE FOR EACH STRAND

MORE ! FOR TWO STRANDS THE CODING IS DIFFERENT FOR THE EVEN NUMBERED HALF-PERIOD AND THE ODD-NUMBERED HALF-PERIOD OF THE SAME STRAND.

This ask for "adaptation and intelligence" (useful traits) so let us leave hand skill aside and chose brain skill instead.

(-L)modulo B == (-21) modulo 15 = 9 B / GDC = NS (number of BIGHT for each of the 3 components)
There are 5 BIGHT STEPS from strand #1 to strand #2 (letter **A**) and 10 BIGHT STEPS from strand #1 to strand #3 (letter **B**) but 5 modulo **NS** is 2 and 10 modulo **NS** is 1
So **A** lag 2 places behind and **B** lag 1 place behind, so for each STRAND it will be for 15 BIGHTS :

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

STRAND # 1

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0                                    1
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
    1   2   3   4   5   6   7   8   9

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           2                        1
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
6   7   8   9                           1   2   3   4   5

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

finished it will be

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           2           4           1           3
@   @   @   @   @   @   @   @   @   @   @   @   @   @   @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

This COMPLEMENTARY BIGHT SEQUENCE WILL BE USED FOR **21L** IN THE FOLLOWING BIGHT ALGORITHM

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           2           4           1           3           0           2
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

The PERIODIC (CYCLIC for SCHAAKE) is

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
            2           0           3           1           4           2           0
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

which leads to

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0           2           4           1           3           0           2
#   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #   #
            2           0           3           1           4           2           0
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Just ‘polish it’ to

```
-----------------------------------------------------------------------------------------------------------------
Read left to right  (ODD numbered HP )
0           2           4           1           3           0           2
#   /   /   \   \   /   /   /   \   \   \   /   /   /   \   \   \   /   /   \   \   #
            2           0           3           1           4           2           0
                                                (EVEN numbered HP )  Read right to left
-----------------------------------------------------------------------------------------------------------------
```

STRAND # 2

This BIGHT SEQUENCE is the start ;

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0              2              4              1              3
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 lag  -2   -1   0
```

we have to add STRAND # **2** which is denoted by letter **A**
finished it is :

```
------------------------------------------------------------------------------------------------------------
Read left to right  (ODD numbered HP )
0   A       2 A       4   A       1   A       3   A       0   A       2   A
#   /   /   \ \   \   /   /   /   \   \   \   /   /   /   \   \   \   /   \   \   #
------------------------------------------------------------------------------------------------------------
#   /   /   \   /   /   /   \   \   \   /   /   /   \   \   \   /   /   /   \   \   #
        A   2       A   0       A   3       A   1       A   4       A   2       A   0
                                     (EVEN numbered HP )  Read right to left
------------------------------------------------------------------------------------------------------------
```

STRAND # 3

This BIGHT SEQUENCE is the start :

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0    A          2    A          4    A          1    A          3    A
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

we have to add STRAND # **3** which is denoted by letter **B**
finished it is :

```
------------------------------------------------------------------------------------------------------------
Read left to right  (ODD numbered HP )
0   A   B   2 A   B   4   A   B   1   A   B   3   A   B   0   A   B   2   A   B
#   /   /   \ \   \   /   /   /   \   \   \   /   /   /   \   \   \   /   \   \   #
------------------------------------------------------------------------------------------------------------
#   /   /   \   /   /   /   \   \   \   /   /   /   \   \   \   /   /   /   \   \   #
    B   A   2   B   A   0   B   A   3   B   A   1   B   A   4   B   A   2   B   A   0
                                     (EVEN numbered HP )  Read right to left
------------------------------------------------------------------------------------------------------------
```

IF YOU KEEP IN MIND THAT IT IS TIME AND NOT SPACE THAT IS THE “KING OF THE REALM” HERE YOU WILL UNDERSTAND THE SUCCESSION FROM #1 to #3 WITHOUT ANY TROUBLE.

HELPING NOTE

so you get the complementary Bight sequence for STRAND **#3**, **#4** and **#5**

```
                     0   A   D   B   C   2   A   D   B   C   1   A   D   B   C
 -4   -3   -2   -1   @   @   @   @   @   @   @   @   @   @   @   @   @   @   @
  A    D    B    C   ( waiting behind the scene in “starting order” )
```

this is where I left you till I heard crying in the woods !

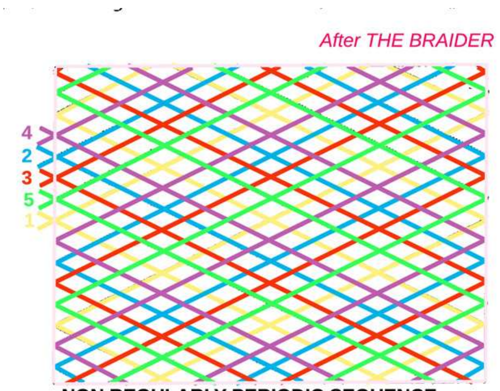
Having a big heart here I bring help before the big bad wolf gulps you down !

I will attempt to give some help.

I suppose (anyway I will not come again on that point as I do believe that people must make some personal real hard effort at some point) that there is no need to explain again neither the PRINCIPLE of the “lateness” of “phase shift”, of “lag” nor the detection of PERIODIC OR NOT PERIODIC DISTRIBUTION OF THE STRANDS ON THE BIGHT RIM.

Yet being a patient soul I will, once more, try to show how to “measure” or quantify this “lagging” between STRANDS.

STRAND #1 is denoted by DIGITS
STRAND #2 is denoted by **A**
STRAND #3 is denoted by **B**
STRAND #4 is denoted by **C**
STRAND #5 is denoted by **D**



NON REGULARLY PERIODIC SEQUENCE

The five strands are immediately adjacent one to another in this upward laying order **1 5 3 2 4**

No need to expound again on the fact that there are two sub-periods or on how to determine those.

one sub-period for **STRAND #1** and **STRAND #2**

another one for **STRAND #3**, **STRAND #4** and **STRAND #5**

Knot is a 20L 15B so (-20)modulo 15 == 10 as step for the complementary.
GDC L & B is 5, that is NS (number of STRANDS)
15B / 5 = 3 that is the Number of BIGHTS for each THK component that are each **4L 3B**

DO NOT FORGET TO KEEP IN MIND « CIRCULARITY » (modularity), try and forget the «false» straight line aspect in the illustration above which is a SPATIAL DISTRIBUTION and that we are searching for a TIME DISTRIBUTION of the laying of STRANDS.

For **STRAND #1** and **STRAND #2**

Between **STRAND #1** and **STRAND #2 or A** TWO bights so **THREE BIGHT STEPS** with modulo 5 applied is THREE BIGHT STEPS so **‘LAGGING’ is 3**

Hence
FROMxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

0 **1**
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @
1 2 3 4 5 6 7 8 9 10

TO xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

0 **2** **1**
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @
1 2 3 4
5 6 7 8 9 10

THEN PUTTING ON THE FIRST “A”xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

-3 -2 - 1 0
0 **2** A **1**
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @

THEN THE NEXT « A » xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
0         A         2         A         1
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
                                             1    2    3    4    5    6    7
8    9    10
|
```

AND NEXT « A » AGAIN xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
0         A         2         A         1         A
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Now we need to adopt anther perspective :
- the diagram is a SPACE SPREAD of the knot
- the BIGHT SEQUENCE is a TIME SPREAD of the laying of the strands
- what is represented here with a straight line is in fact a curved line closing into a circle so that

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
0
@    @    @    @    @    @    @    @    @    @    @    @    @    @    @
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

may be seen as

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
                         0
@    @    @    @    |    @    @    @    @    @    @    @    @    @    @    @    @    @    @    @    |
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

with the | @ @ @ @ | being the area for gluing when transforming in a circle by bending downwards the two extremities (so reading clockwise)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
0                                                  1
@    @    @    @    |    @    @    @    @    @    @    @    @    @    @    @    @    @    @    @    |
                         0    1    2    3    4    5    6    7    8    9    10
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
                         0                        2                        1
@    @    @    @    |    @    @    @    @    @    @    @    @    @    @    @    @    @    @    @    |
                                                                           0    1    2    3    4
1    2    3    4         5    6    7    8    9    10
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

now we do the trick of putting in the A for strand #2 that has a 'lateness » of 3:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
    -3   –2   –1         0
     A1                  0                        2                        1         A1
@    @    @    @    |    @    @    @    @    @    @    @    @    @    @    @    @    @    @    @    |
     0    1    2         3    4    5    6    7    8    9    10
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

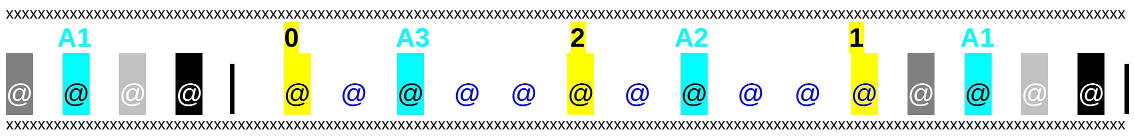
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
     A1                  0         A3             2         A2             1         A1
@    @    @    @    |    @    @    @    @    @    @    @    @    @    @    @    @    @    @    @    |
4    5    6    7         8    9    10                        0    1    2    3    4    5    6    7
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

This will be a useful perspective for the treatment of the second sub-period.

Had all been nicely regularly periodic we would have used the trick already known for that situation.
Now we will have to start for STRAND **#3**, **#4**, **#5** with a TIME SPREAD that will looks like



IFF all had been nice and regular periodic we would have had :
digit – **strand #4** – **strand #2** – **strand #3** – **strand #5** – digit…. which translate into

digit – **C** – **A** – **B** – **D** – digit …..

suppose we take **off** the letter immediately following after **A** which letter is **B** :

digit – **C** – **A** – – **D** – digit letter **A** can not yet gently 'slides" next to a digit, so we take **off** the next letter **C**

digit – – **A** – – **D** – digit now **A** still cannot slides to the right but can slides leftward

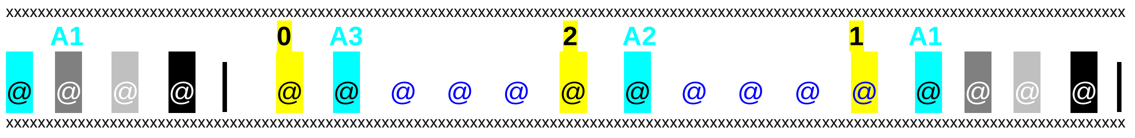
digit – **A** – – – **D** – digit take **off** the last letter that is **D** to get at the new starting situation

digit – **A** – – – – digit

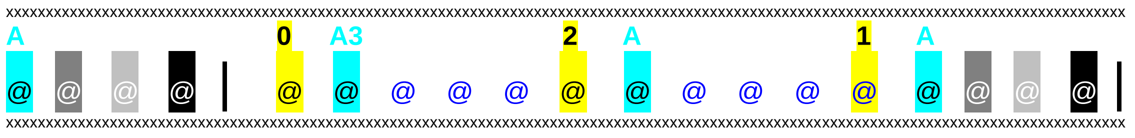
May be it could even have been : let the letter **D** slides towards its nearest LETTERED neighbour, that would have been digit – **A** – **D** – – – digit with "only" the placing of **B** and **C** to be determined…..
We could even have hoped for the placing following the alphabetical order to
digit – **A** – **D** – **B** – – digit
and then
digit – **A** – **D** – **B** – **C** – digit

ALWAYS KEEP IN MIND THAT YOU ARE SHIFTING "TIME SPREAD" AND NOT "SPACE SPREAD" AND YOU WILL BE LESS CONFUSED (confusion is born from that you are inscribing a shifting time sequence on a linear space (the piece of paper)

We put adjacent the 'nearest POSSIBLE' neighbour and that leads to



For **STRAND #3**, **STRAND #4** and **STRAND #5**
We must start again but not anew BUT with an existing situation

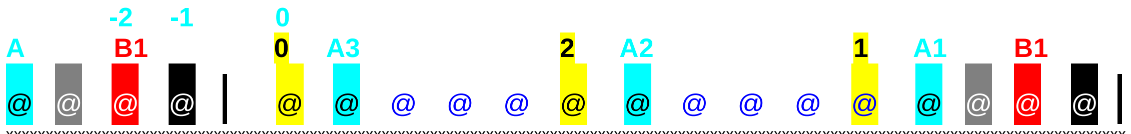


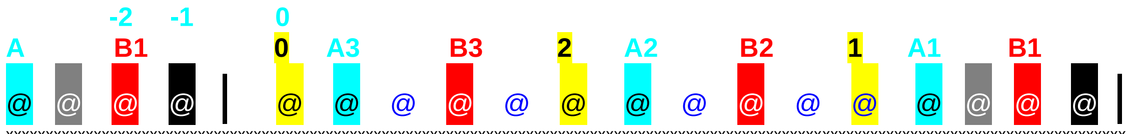
STRAND **#3** is denoted '**B**' STRAND **#4** is denoted '**C**' STRAND **#5** is denoted '**D**'

We have now to 'introduce' those strands in the TIME sequence respecting their RELATIVE position to one another in order to respect the correct TIME SPREAD.
we can go
B C D **D B C** **C D B** **B D C** **D C B** **C D B**
1 on 6 is not much for "try and fail" attempts so we abandon randomness and use some pondering.

Observe that in the diagram of SPACE SPREAD **strand #3** or **B** is 2 BIGHT STEPS above **strand #1** that is denoted by **digits** so it must be on the **LEFT** of it LAGGING on it by 2

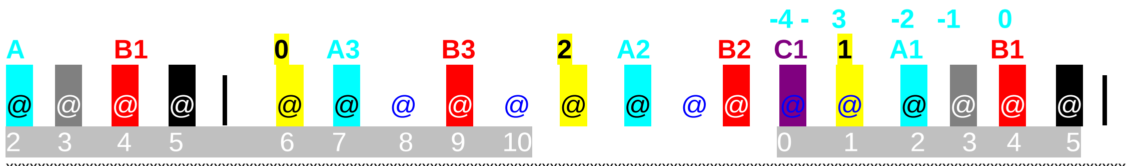
let us try



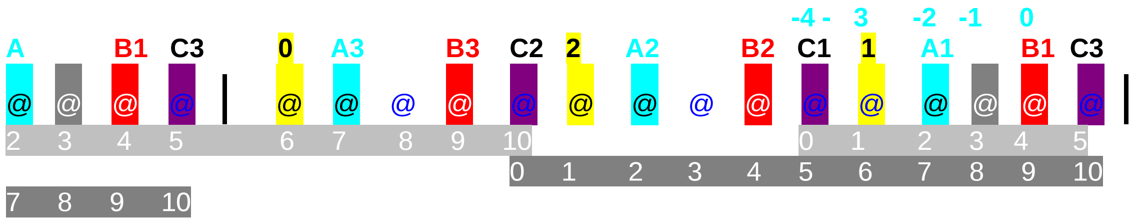


strand #4 (**C**) is **2** 'bight step' above **strand #3** or **B** ;
4 bight steps **above #1** and 1 bight step **below** strand #1 (think periodic / cyclic / circular) but may we use periodic process ? Not as we did before I think.

so either 2 on the **left** of the « **new zero strand** », that is **B,** but then the place is already occupied by **A ;** next again on the left is the place occupied by a **DIGIT** ; next again on the **left** is free.
4 bight steps **above** #1 is also **4** on the **left** of the « **new zero strand** » which is **B**
Let us go for that

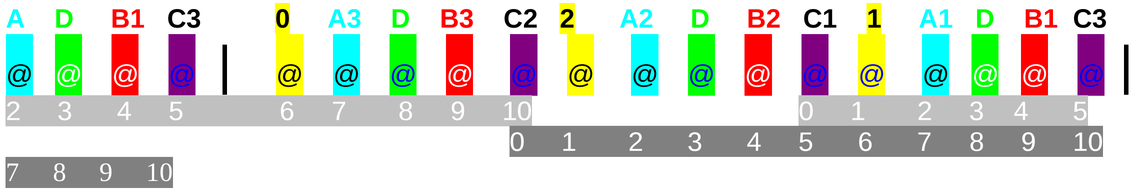


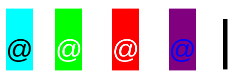
going on



Now the only « free room » is for strand #5 or **D**
This strand on the SPACE SPREAD is one bight step above strand #1 and so we can suppose one bight step on the LEFT of the new strand zero that is **B**

That seems to works



We can now suppress the  part that has served its usefulness

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0  A3  D  B3  C2  2  A2  D  B2  C1  1  A1  D  B1  C3
@  @   @  @   @   @  @   @  @   @   @  @   @  @   @
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

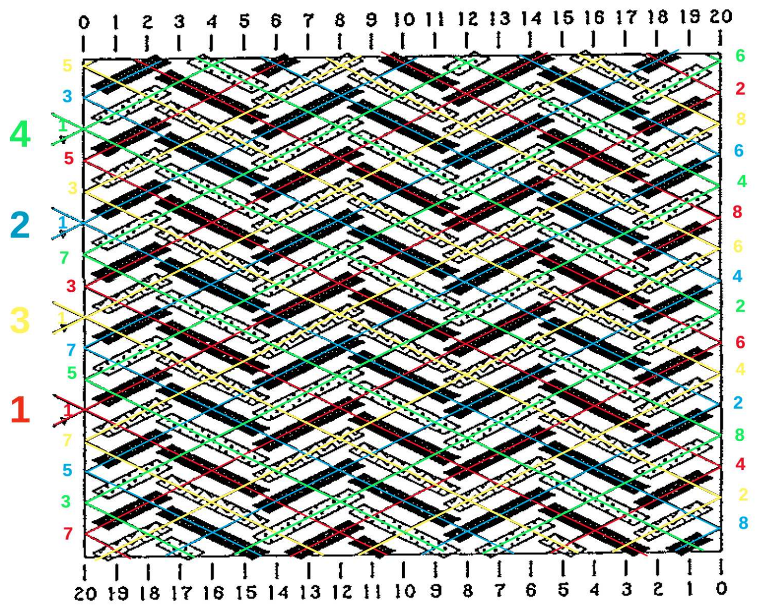
with this BIGHT SEQUENCE OF 15 we can do the full BIGHT ALGORIGHTM for 25 LEAD

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONCERN ODD NUMBERED HALF-PERIODS
TO BE READ LEFT TO RIGHT
0 A D B C 2 A D B C 1 A D B C 0 A D B C 2 A D B C
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
x x x x x x x x x x x x x x x x x x x x x x x x x x
  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
  C B D A 2 C B D A 0 C B D A 1 C B D A 2 C B D A 0
CONCERN EVEN NUMBERED HALF-PERIODS
TO BE READ RIGHT TO LEFT
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

It just remains to replace the x x x ..... x x x x by the appropriate code signs.

Now another exercise that you will do all by yourself if you want the training and to « verify » if what decoded of SCHAAKE's instruction (given in full in the quote) is correct.

EXERCISE



After THE BRAIDER Example 13 (page 194) 20L 16B

FIND THE TWO ALGORITHMS : ONE FOR STRANDS #1 and #2 and the OTHER FOR STRANDS #3 and #4
END RESULTS ARE

For STRAND #1 and STRAND #~2

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0 - A - 3 - A - 2 - A - 1 - A - 0 - A -
# / / \ \ \ / / / \ \ \ / / / \ \ \ / / #
  - A - 0 - A - 1 - A - 2 - A - 3 - A - 0
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

For STRAND #3 and STRAND #~4

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

0 B C A 3 B C A 2 B C A 1 B C A 0 B C A

/ / \ \ \ / / / \ \ \ / / / \ \ \ / /

A C B 0 A C B 1 A C B 2 A C B 3 A C B 0

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

JUST VERIFY WITH EVEN NUMBERED HALF-PERIOD OF N° 6 of STRAND B (remember that strand C does not YET exist at this precise moment in time so cannot participate in crossings.)

- -

/ / \ \ \ / / / \ \ \ / / / \ \ \ / /

A B 0 A B 1 A B 2 A B 3 A B 0

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

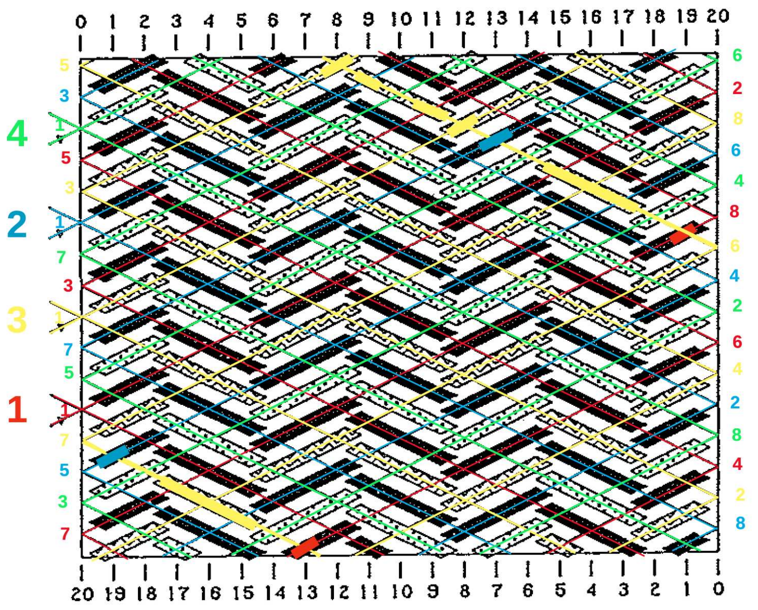
so it is **UNDER – OVER – OVER – OVER – UNDER – UNDER – OVER – OVER – UNDER – UNDER – OVER – OVER – OVER – UNDER**

or **U1 – O3 – U2 – O2 - U2 - O3 – U1**

Verify with the diagram here under (remember NOT TO COUNT ANY CROSSING WHERE THE **GREEN STRAND #4** or C IS AN ELEMENT)

On the next page is the beginning of a solution but don't look at it.

Cheating will only be cheating with yourself and destroying a good training verification of what I said starting from Schaake's words.



After THE BRAIDER Example 13 (page 194) 20L 16B

IFF all had been nicely regular periodic it would have been

digit – strand #4 - strand #2 - strand #3 - digit……….

digit – C - A - B - digit……….

digit – C - A - - digit……….
A may now immediately slides rightward toward the nearest digit accessible

digit – C - - A - digit……….

it could even be (regrouping of letters **)**
digit – - C - A - digit……….

which would complete to

digit – B - C - A - digit……….

we can hope ! let us verify in another way !

Starting situation is

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0   x  x  A  3   x   x  A  2  x   x  A  1  x   x  A
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

B or **strand #3** is 3 Bight steps above (so will be on the left) **strand #1**

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
     -3 –2  –1  0   -3   -2  -1  0  -3   -2  -1  0
0     B   x   A  3   B    x   A  2  B    x   A  1  B   x   A
      0   1   2  3   4    5   6  7  8    9   10  11 12
                                                   0   1   2
3     4    5   6  7   8    9   10  11 12

                                          0   1   2   3   4   5   6
7     8    9  10  11   12
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

why 12 ? have you forgotten (-L) modulo B == (-20) modulo 16 == 12 that is why !

Of course in : 0 B x A 3 B x A 2 B x A 1 B x A the only place left is for C

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
0  B  C  A  3  B  C  A  2  B  C  A  1  B  C  A
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

you have the bight complementary sequence you can get the periodic or cyclic sequence and then built up the BIGHT ALGORITHM then add the code and use to find the coding of each half-period.

ANNEXE

About **PERIODICITY** :

GDC of **LEAD** and **BIGHT** is 'NS' **number of STRANDS** needed to make the knot

Observe that (NS-1) ! -that read as "factorial NS-1"- will give the number of different periodic sequences that are possible :
Example NS = 3
I am sure that many will say that it is 3 ! == 3 * 2 * 1 = 6 and not (3 – 1)! = 2 ! == 2 * 1 = 2 that gives the correct result.
Well they are wrong and making a confusion between 'concepts'

There are indeed 6 ways to 'arrange' A ; B ; C :
A B C **B C A** **C A B** **A C B** **C B A** **B A C**
but observe that only make 2 sequences or in fact one circular sequence but that can be read once on clockwise direction and once in counter-clockwise direction

$$\begin{tikzcd}
& A \arrow[dl] & \\
B \arrow[rr] & & C \arrow[ul]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& A \arrow[dr] & \\
B \arrow[ur] & & C \arrow[ll]
\end{tikzcd}$$

CIRCULAR sequence **ONE**
READ **A B C / B C A / C A B** it
is still the same sequence

CIRCULAR sequence **TWO**
READ **A C B / C B A / B A C** it
is still the same sequence

NEVER FORGET **CIRCULAR / MODULAR** PLEASE !

So with NS = 5 4! == 4 * 3 * 2 * 1 = 24 different sequences but ALL WILL NOT BE REGULARLY PERIODIC in their distribution of the STRANDS on the BIGHT RIM.
Still if we still think "relatively prime" then from 1 to 4 there are only TWO numbers that are relatively prime with 5 : **1** and **3**

so TWO sequences will be regularly periodic
*** with step = **1**
(read that CIRCULAR) **1 2 3 4 5 == 2 3 4 5 1 == 3 4 5 1 2 == 4 5 1 2 3 == 5 1 2 3 4**
*** with step = **3**
(read that CIRCULAR) **1 4 7 10 13** but applying modulo NS (5 here) it is **1 4 2 5 3**
1 4 2 5 3 == 4 2 5 3 1 == 2 5 3 1 4 == 5 3 1 4 2 == 3 1 4 2 5
ONLY WITH REGULAR PERIODICITY IN THE REPARTITION OF THE STRANDS ON THE BIGHT RIM CAN WE USE ONE ALGORITHM ONLY.

Another calculation : **BIGHT / NS = S**
S is the number bight-points (the little peaks at the bight rim – their number is EQUAL to the number of BIGHTS of a COMPONENT MADE WITH A SINGLE STRAND – **NS** is the NUMBER OF SINGLE STRAND COMPONENT) on EACH BIGHT RIM made by EACH OF THE **NS** SINGLE STRAND.
It follow that the number of STEP from (zero) one BIGHT to the immediately next BIGHT made by the same single STRAND will be **BIGHT / S = NS**

20 **L** 15 **B** that imply **GDC** of **L** and **B** = 5
so **B / NS** == 15 / 5 = 3 **S** = 3

B / S = NS 15 / 3 = 5 so 5 steps of one bight apex of whatever colour (supposing that each of the strands is of a different colour) to get once more on an apex made by the same strand on the same bight rim.

To get a regular repartition of the STRANDS along the bight rim **NS** and **S** must be relatively prime with each other.

Remember also that you will have to calculate the index 'i' of each HALF-PERIOD and that this can be done with 2 formulas.
i = (HPodd –3) / 2 **i=(HPeven – 2) / 2**

example :

HP 1	i = negative –discard – free run flag	HP 2	i = 0
HP 3	i = 0	HP 4	i = 1
HP 5	i = 1		
….		HP 8	i = 3
HP 9	i = 3	HP 10	i = 4
HP 11	i = 4		
….			
		HP 30	i = 14
HP 31	i = 14	HP 32	i = 15

I BEG OF YOU :

PLEASE BE A WELL BEHAVE READER, EMAIL ME ABOUT ANY MISTAKE YOU MAY FIND IN THIS TEXT (help me by stating precisely why you believe it is a mistake.)