FURTHER MAPLE EXPLORATIONS OF FORBIDDEN PATTERNS IN COIN TOSSING

Dr. Joel C. Fowler
Mathematics Department, Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060-2896
email: jfowler@spsu.edu

We use Maple, at a student project level, to continue our exploration of recurrence relations for enumerating bit strings not containing certain sub-strings. By using two differing theoretical approaches, and Maple for computations, we find unexpected patterns and theorems for some collections of forbidden sub-strings.

```
> restart;
```

Let a(n, k) = the number of binary strings of length n without a run of k consecutive 1's. For example: a(3, 2)=5 (000,001,010,100,101). a(n, k) has been extensively studied. A nice introduction to the problem and the literature on its solution can be found in [M. Schilling, The Longest Run of Heads, The College Mathematics Journal, Vol. 21, No. 3 (1990) 196 - 207].

The most elementary approach to the problem involves recurrence relations. We may divide the a(n, k) strings of length n without k consecutive 1's into cases based on the first appearance of 0 from the left end of the string. If it occurs in bit:

1, the string is: "**0 [any length n-1 string w/o a run of k 1's]**" , hence a(n-1, k) **;**

2, the string is: "**1 0 [any length n-2 string w/o a run of k 1's]**", hence a(n-2, k) **;**

3, the string is: "**1 1 0 [any length n-3 string w/o a run of k 1's]**", hence a(n-3, k) **;**
and so on, until the last case, k, since the string does not contain k consecutive 1's:

the string is "**1111 ... 10 [any length n - k string w/o a run of k 1's]**" , hence a(n-k, k)**.**
Thus we have: a(n, k) = a(n - 1, k) + a(n - 2, k) + a(n - 3, k) + ... + a(n - k, k) ; and the initial conditions: a(0, k) = 1, a(1, k) = 2, a(2, k) = 4, ... , a(k - 1, k) = 2^(k-1) .

We can use a(n,k) to determine the number of binary strings without a run of k consecutive 0's or 1's. Let b(n, k) = the number of length n binary strings without a run of k 0's or 1's . We use a matrix approach for the case k = 3, which generalizes to a recurrence relation for all values of k .

Define the column vector **v**(n) = $\begin{bmatrix} \mathrm{v0}(n) \\ \mathrm{v1}(n) \\ \mathrm{v2}(n) \\ \mathrm{v3}(n) \end{bmatrix}$, where v0(n) = the number of length n strings ending in 00 without a run of 3 zeros or ones; v1(n) = the number of length n strings ending in 01 without a run of 3 zeros or ones; v2(n) = the number of length n strings ending in 10 without a run of 3 zeros or ones; v3(n) = the number of length n strings ending in 11 without a run of 3 zeros or ones.

We then have, by considering each case in turn:

v0(n+1) = v2(n), since these strings of length n+1 must to end in 100;
v1(n+1) = v0(n) + v2(n), since these strings of length n+1 must end in 001 or 101;
v2(n+1) = v1(n) + v3(n), since these strings of length n+1 must end in 010 or 110;
v3(n+1) = v1(n), since such a string of length n+1 would have to end in 011 .

Hence we have the matrix relationship **v**(n+1) = **B** ***v**(n), where **B** = $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$. Note that **v**(2) = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, and thus **v**(n) = **B** ^(n-2) $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$. And, since b(n, 3) = v0(n) + v1(n) + v2(n) + v3(n) = [1 1 1 1] **v**(n) , we then have, for n > 1 :

$$b(n, 3) = [\,1\ \ 1\ \ 1\ \ 1\,]\ \mathbf{B}\,\hat{}\,(n-2) \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

A quick route to a recurrence relation for b(n, 3) is to note that the minimal polynomial for **B** yields a linear recurrence relation that the sequence must satisfy.

```
> with(LinearAlgebra):
> B := <<0,1,0,0>|<0,0,1,1>|<1,1,0,0>|<0,0,1,0>>;
```

$$B := \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

```
> MinimalPolynomial(B,x);
```

$$-1 - 2\,x - x^2 + x^4$$

Hence we have the linear, homogenous recurrence: b(n, 3) = b(n - 2, 3) + 2*b(n - 3, 3) + b(n - 4, 3), for n > 5 , which, along with the initial conditions: b(0, 3) = 1, b(1, 3) = 2, b(2, 3) = 4, b(3, 3) = 6, b(4, 3) = 10, and b(5, 3) = 16, determines the sequence. Use of the minimal polynomial generates a linear recurrence, but not always one of lowest order. Using Maple, we can interate the powers of the matrix B to hunt for a pattern.

```
> seq(<<1>|<1>|<1>|<1>>.B^i.<1,1,1,1>, i=1..10);
```

[6], [10], [16], [26], [42], [68], [110], [178], [288], [466]

It appears that b(n, 3) is determined by: b(n, 3) = b(n-1, 3) + b(n-2, 3), for n > 2; b(0, 3) = 1, b(1, 3) = 2, b(2, 3) = 4 . We can show this recurrence relation holds in general:

Suppose a sequence, $\{a(n)\}$, satisfies a linear, homogeneous recurrence relation, R, of order k. Let R' be a linear, homogeneous recurrence relation of order $p < k$. It can be shown by induction that: If the first k terms of $\{a(n)\}$, after the initial conditions of R', satisfy R', then all of $\{a(n)\}$ must satisfy R'. Now we have already noted that, since b(n,3) is produced by powers of a matrix, its minimal polynomial yields the 4th order linear, homogenous recurrence relation $b(n, 3) = b(n - 2, 3) + 2*b(n - 3, 3) + b(n - 4, 3)$. Thus we need only note that the first 4 terms (past the first two) computed above for b(n, 3), satisfy $b(n, 3) = b(n-1, 3) + b(n-2, 3)$ to establish that the entire sequence satisfies this recurrence relation, which we have already seen by inspection.

Note that this is the same recurrence relation for no 2 consecutive ones, i.e. a(n, 2). In fact, by comparing the initial conditions for each sequence we have:

$$b(n, 3) = 2 * a(n - 1, 2), \text{ for } n > 1 .$$

This surprising result holds in general. That is,

$$b(n, k) = 2 * a(n - 1, k - 1), \text{ for } n > 1 .$$

To see why this is true in general, we first note that every binary string of length n can be uniquely encoded as a string of length n of the form:

$$\underset{1}{\underline{0 \text{ or } 1}}\ \ \underset{2}{\underline{+0 \text{ or } +1}}\ \ \underset{3}{\underline{+0 \text{ or } +1}}\ \ \underset{4}{\underline{+0 \text{ or } +1}}\ \ \ldots\ldots\ \ \underset{n}{\underline{+0 \text{ or } +1}}\quad \pmod 2$$

where the +0 or +1 indicates the operation to be used on the previous bit to produce the next. Thus a string with no k consecutive 0's or 1's corresponds directly to an encoding that begins 0 or 1 in the first position, and then contains no k-1 consecutive +0's in the last n-1 positions. This argument is the approach used in [M. Schilling, The Longest Run of Heads, The College Mathematics Journal, Vol. 21, No. 3 (1990) 196 - 207].

Note that this matrix approach, is easily adaptable to enumerating strings with various pattern avoidances. For example, we find a recurrence relation for length n binary strings without the substrings 000 or 110. Let f(n) = # binary strings of length n without 000 or 110 .

Then proceeding as before: Define the column vector $\mathbf{v}(n) = \begin{bmatrix} v0(n) \\ v1(n) \\ v2(n) \\ v3(n) \end{bmatrix}$, where v0(n) = the number of length n strings ending in 00 without 000 or 110; v1(n) = the number of length n strings ending in 01 without 000 or 110; v2(n) = the number of length n strings ending in 10 without 000 or 110; v3(n) = the number of length n strings ending in 11 without 000 or 110 . We then have, by considering each case in turn:

$v0(n+1) = v2(n)$, since these strings of length n+1 must end in 100;

$v1(n+1) = v0(n) + v2(n)$, since these strings of length n+1 must end in 001 or 101;

$v2(n+1) = v1(n)$, since these strings of length n+1 must end in 010;

$v3(n+1) = v1(n) + v3(n)$, since these strings of length n+1 must end in 011 or 111 .

Thus the transition matrix from $\mathbf{v}(n)$ to $\mathbf{v}(n+1)$, for $n > 1$, is:

```
> F := <<0,1,0,0>|<0,0,1,1>|<1,1,0,0>|<0,0,0,1>>;
```

$$F := \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

```
> MinimalPolynomial(F,x);
```

$$1 - x^2 - x^3 + x^4$$

Hence, past the first few terms, the sequence must satisfy f(n) = f(n-1) + f(n-2) - f(n-4). To see if there is a lower order recurrence relation for this sequence, we can generate a list of terms using powers of the matrix F, and then use Maple's linear regression routines to find the lowest order linear, homogeneous recurrence relation that exactly fits the terms. Using the routine below with TryOrder=1, 2, 3, ... we find the first fit with 0 error is:

```
> TryOrder := 4 ; Seque := i -> <1 | 1 | 1 | 1>.F^i .
<1,1,1,1> : MatEnt2 := (i,j) -> Seque(TryOrder+i-j-1) :
SolveMat := Matrix(4,TryOrder, MatEnt2) : VectEnt := (i,j)
-> Seque(TryOrder+i-1) : SolveVect := Matrix(4,1,
VectEnt): Sol := LeastSquares(SolveMat,SolveVect);
Norm(SolveMat.Sol - SolveVect,2); a(n) = (Transpose(<seq(
a(n-i), i=1..TryOrder)>) .Sol)[1];
```

$$TryOrder := 4$$

$$Sol := \begin{bmatrix} 1 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

$$0$$

$$\mathrm{a}(n) = \mathrm{a}(n-1) + \mathrm{a}(n-2) - \mathrm{a}(n-4)$$

We can automate the entire algorithm for any set, S, of forbidden substrings (all with the same length) using a few basic routines and a Maple procedure.

```
> restart;
> with(StringTools): with(LinearAlgebra):
> FullBin := (n,k) -> cat( Fill( "0", k - length( convert(
convert(n,binary), string) )), convert( convert(n,binary),
string) ):
> MinRecRelBinary := proc(S::set)    description "Minimal
Recurrence Relation for enumerating binary strings without
any element of S as a substring";
slength := length(S[1]) :

MatEnt := (i,j) -> `if`( Drop( FullBin(j-1,slength-1),1) =
Take(FullBin(i-1,slength-1),slength-2)   and   not(   cat(
```

```
FullBin(j-1,slength-1),     FullBin(i-1,slength-1)[slength-
1]) in S ) , 1, 0):

M := Matrix(2^(slength-1),2^(slength-1),MatEnt):

for i from 0 to max(2^slength,12) do Seque(i) :=
(Matrix(1,2^(slength-1),1) . M^i . Matrix(2^(slength-
1),1,1))[1,1] end do:

NormResult := 1 :

for TryOrder from 1 while NormResult<>0 do
       MatEnt2 := (i,j) -> Seque(TryOrder+i-j-1) :
       SolveMat := Matrix(2^(slength-1),TryOrder,MatEnt2):
       VectEnt := (i,j) -> Seque(TryOrder+i-1) :
       SolveVect := Matrix(2^(slength-1),1, VectEnt):
       Sol := LeastSquares(SolveMat,SolveVect):
       NormResult := Norm(SolveMat.Sol - SolveVect,2) :
end do:

op([S,a(n) = (Transpose(<seq( a(n-i), i=1..(TryOrder-1))>)
.Sol)[1], [seq(Seque(k),k=0..10)]]);

end proc;
```

We can investigate the recurrence relations for various sets, S, of forbidden substrings, and search for more general theory. Here are two examples with predictable results:

```
>MinRecRelBinary({"01","11"});
```

$$\{"01", "11"\}, a(n) = a(n-1), [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$$

```
>MinRecRelBinary({"000", "111", "011","110","101"});
```

$$\{"000", "011", "101", "110", "111"\}, a(n) = a(n-1), [4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]$$

For a bit more complexity, compare these two, and explain:

```
>MinRecRelBinary({"0000", "0001"});
```

$$\{"0000", "0001"\}, a(n) = a(n-1) + a(n-2) + a(n-3),$$
$$[8, 14, 26, 48, 88, 162, 298, 548, 1008, 1854, 3410]$$

```
>MinRecRelBinary({"000"});
```

$$\{"000"\}, a(n) = a(n-1) + a(n-2) + a(n-3),$$
$$[4, 7, 13, 24, 44, 81, 149, 274, 504, 927, 1705]$$

However, not all recurrence relations exhibit so much regularity. Consider:

```
>MinRecRelBinary({"1000","0100","0010","0001"});
```

$\{"0001", "0010", "0100", "1000"\}$,

$$a(n) = 2\,a(n-1) - a(n-3) + a(n-4) - a(n-5) - a(n-6) + a(n-7),$$

$[8, 12, 20, 34, 58, 98, 167, 286, 490, 839, 1437]$

One avenue of exploration is to find sets of forbidden substrings that yield "regular" recurrence relations admitting some type of theoretical explanation. For example, what about alternating substrings?

```
> MinRecRelBinary({"10101","01010"});
```

$\{"01010", "10101"\}$, $a(n) = a(n-1) + a(n-2) + a(n-3) + a(n-4)$,

$[16, 30, 58, 112, 216, 416, 802, 1546, 2980, 5744, 11072]$

```
> MinRecRelBinary({"0101","1010"});
```

$\{"0101", "1010"\}$, $a(n) = a(n-1) + a(n-2) + a(n-3)$,

$[8, 14, 26, 48, 88, 162, 298, 548, 1008, 1854, 3410]$

It appears that strings without an alternating substring of length k have recurrence relations of the form: $a(n) = a(n - 1) + a(n - 2) + a(n - 3) + ... + a(n - (k - 1))$.
To see why this is true in general, recall that every binary string of length n can be uniquely encoded as a string of length n of the form

0 or 1	+0 or +1	+0 or +1	+0 or +1	....	+0 or +1	(mod 2)
1	2	3	4		n	

A string with no alternating substring of k consecutive 0's or 1's corresponds to an encoding that begins 0 or 1 , and then contains no k-1 consecutive +1's in the last n-1 positions. Since 0 and 1 are interchangeable, we have the number of strings of this type is twice the number of stings of length n-1 that do not have k-1 consecutive 0's. We have already established the above recurrence relation for that sequence.
Carrying this idea further, consider binary strings without a run of k identical <u>or</u> alternating bits. Such a string would correspond to an encoding without any run of k-1 consecutive +0's or +1's in the last n-1 positions. We have already established that such a string is governed by the recurrence relation:

$$a(n) = a(n - 1) + a(n - 2) + a(n - 3) + ... + a(n - ((k-1) - 1)).$$

This is easily verfied:

```
> MinRecRelBinary({"00000","11111","01010", "10101"});
```

$\{"00000", "01010", "10101", "11111"\}$, $a(n) = a(n-1) + a(n-2) + a(n-3)$,

$[16, 28, 52, 96, 176, 324, 596, 1096, 2016, 3708, 6820]$

It is worth noting that the connection between the regularity of the strings and recurrence relation is fairly sensitive. Consider:

```
> MinRecRelBinary({"0000", "1010", "0101"});
```

$\{"0000", "0101", "1010"\}$,

$$a(n) = a(n-1) + 2\,a(n-3) + a(n-4) - a(n-6) - a(n-7),$$

$[8, 13, 23, 40, 70, 122, 213, 372, 650, 1135, 1982]$

This procedure for finding recurrence relations over binary strings is easily generalized to the q-ary alphabet {0, 1, 2, ... , q-1} using Maple.

```
> QaryRep := (n,q) -> Remove( IsPunctuation or IsSpace,
convert(
[seq(convert(n,base,q)[(nops(convert(n,base,q))+1)-i],
i=1..(nops(convert(n,base,q))))],string)):
>QaryRep(31,3);
```

"1011"

```
> FullQaryRep := (n,k,q) -> cat( Fill( "0", k - length(
QaryRep(n,q))), QaryRep(n,q) ):
>FullQaryRep(100,6,5);
```

"000400"

```
>MinRecRelAnyBase := proc(S::set,q)   description "Minimal
Recurrence Relation for enumerating q-ary strings over {0,
1, 2, ..., q-1} without any element of S as a substring";

slength := length(S[1]);

# "Populate the matrix M"
MatEnt := (i,j) -> `if`( Drop( FullQaryRep(j-1,slength-
1,q),1) = Take( FullQaryRep(i-1,slength-1,q),slength-2)
and not( cat( FullQaryRep(j-1,slength-1,q), FullQaryRep(i-
1,slength-1,q)[slength-1]) in S ) , 1, 0): M :=
Matrix(q^(slength-1),q^(slength-1),MatEnt);

# "Compute sequence terms from powers of M"
Temp := (Matrix(1,q^(slength-1),1)) . M :   Seque(0) :=
q^(slength-1) :
for i from 1 to max(q^slength,12) do
    Seque(i) := ( Temp . Matrix(q^(slength-1),1,1))[1,1]:
    Temp := Temp . M :
end do:

# "Find recurrence relation fit, by first regressing
against short vector of terms, and then checking against
the full list of terms."
CheckFit := 1 :
for SeqDepth from 1 while CheckFit<>0 do
   NormResult := 1 :
        for TryOrder from 1 while NormResult<>0 do
            MatEnt2 := (i,j) -> Seque(TryOrder+i-j-1) :
SolveMat := Matrix(SeqDepth*slength,TryOrder, MatEnt2) :
            VectEnt := (i,j) -> Seque(TryOrder+i-1) :
SolveVect := Matrix(SeqDepth*slength,1, VectEnt):
```

```
            Sol := LeastSquares(SolveMat,SolveVect):
            NormResult := Norm(SolveMat.Sol-SolveVect,2) :
        end do:

   FullMat := Matrix(q^(slength-1),TryOrder-1, MatEnt2) :
FullVect := Matrix(q^(slength-1),1, VectEnt):
   CheckFit := Norm( FullMat.Sol - FullVect) :
end do:

op([S,a(n) = (Transpose(<seq( a(n-i), i=1..(TryOrder-1))>)
.Sol)[1], [seq(Seque(k),k=0..10)]]);

end proc;
```

A larger alphabet brings much more variation. The recurrence relations can be messy.

```
> MinRecRelAnyBase({"0000", "1010", "0101", "2222", "1212",
"2121", "1111"},4);
```

{"0000", "0101", "1010", "1111", "1212", "2121", "2222" }, $a(n) = 3\,a(n-1) + 2\,a(n-2) + 5\,a(n-3) + 6\,a(n-4) - 4\,a(n-5) - 4\,a(n-6) - 5\,a(n-7) - 4\,a(n-8) - a(n-9)$, [64, 249, 975, 3816, 14938, 58470, 228867, 895843, 3506560, 13725570, 53725386]

We can check a familiar forbidden string over several alphabets.

```
> for q from 3 to 4 do MinRecRelAnyBase( {"000"} , q) end
do;
```

{"000" }, $a(n) = 2\,a(n-1) + 2\,a(n-2) + 2\,a(n-3)$,
[9, 26, 76, 222, 648, 1892, 5524, 16128, 47088, 137480, 401392]

{"000" }, $a(n) = 3\,a(n-1) + 3\,a(n-2) + 3\,a(n-3)$,
[16, 63, 249, 984, 3888, 15363, 60705, 239868, 947808, 3745143, 14798457]

```
> for q from 3 to 4 do MinRecRelAnyBase( {"0000"} , q) end
do;
```

{"0000" }, $a(n) = 2\,a(n-1) + 2\,a(n-2) + 2\,a(n-3) + 2\,a(n-4)$,
[27, 80, 238, 708, 2106, 6264, 18632, 55420, 164844, 490320, 1458432]

{"0000" }, $a(n) = 3\,a(n-1) + 3\,a(n-2) + 3\,a(n-3) + 3\,a(n-4)$, [64, 255, 1017, 4056, 16176, 64512, 257283, 1026081, 4092156, 16320096, 65086848]

It appears that, in general, the recurrence relation for strings over a q-ary alphabet without a run of k 0's is given by:

$$a(n) = (q-1)*a(n - 1) + (q-1)*a(n - 2) + (q-1)*a(n - 3) + \ldots + (q-1)*a(n - k) .$$

This is easily verified by elementary counting. Divide the strings of length n into cases based on how far from the left the first non-zero character appears. If that position is i, then there are (q-1)*a(n - i -1) such strings, since positions 1, 2, 3, . . . , i must all be 0's; position i +1 must be any of the q - 1 non-zero characters; and the remaining n - i - 1 positions must be filled by any length n - i - 1 string not containing a run of k 0's. Summing over i = 1, 2, ... , k-1 yields the recurrence relation.

What about strings that contain no length k run of any of their q characters? We can easily gather some particular cases:

```
>          MinRecRelAnyBase({"000","111",          "222"},3);
MinRecRelAnyBase({"0000","1111",                 "2222"},3);
MinRecRelAnyBase({"00000","11111", "22222"},3);
```

{"000", "111", "222" }, $a(n) = 2\ a(n-1) + 2\ a(n-2)$,

[9, 24, 66, 180, 492, 1344, 3672, 10032, 27408, 74880, 204576]

{"0000", "1111", "2222" }, $a(n) = 2\ a(n-1) + 2\ a(n-2) + 2\ a(n-3)$,

[27, 78, 228, 666, 1944, 5676, 16572, 48384, 141264, 412440, 1204176]

{"00000", "11111", "22222" },

$a(n) = 2\ a(n-1) + 2\ a(n-2) + 2\ a(n-3) + 2\ a(n-4)$,

[81, 240, 714, 2124, 6318, 18792, 55896, 166260, 494532, 1470960, 4375296]

```
>      MinRecRelAnyBase({"00","11",      "22",       "33"},4);
MinRecRelAnyBase({"000","111",       "222",       "333"},4);
MinRecRelAnyBase({"0000","1111", "2222", "3333"},4);
```

{"00", "11", "22", "33" }, $a(n) = 3\ a(n-1)$,

[4, 12, 36, 108, 324, 972, 2916, 8748, 26244, 78732, 236196]

{"000", "111", "222", "333" }, $a(n) = 3\ a(n-1) + 3\ a(n-2)$,

[16, 60, 228, 864, 3276, 12420, 47088, 178524, 676836, 2566080, 9728748]

{"0000", "1111", "2222", "3333" }, $a(n) = 3\ a(n-1) + 3\ a(n-2) + 3\ a(n-3)$, [64, 252, 996, 3936, 15552, 61452, 242820, 959472, 3791232, 14980572, 59193828]

From these examples, and others, it appears that, in general, the number of strings over a q-ary alphabet with no k consecutive identical characters has the recurrence relation:
a(n) = (q-1)*a(n - 1) + (q-1)*a(n - 2) + (q-1)*a(n - 3) + ... + (q-1)*a(n - k + 1) .

This can be proved by a more general form of the encoding method used previously. Every q-ary string may be represented uniquely as a string of the form:

<u>0 - (q-1)</u>	<u>+ b(1)</u>	<u>+ b(2)</u>	<u>+ b(3)</u>	. . . .	<u>+ b(n-1)</u>	(mod q)
1	2	3	4		n	

where the b(i) are from {0, 1, 2, . . . , q-1}. A string will contain a run of k identical characters if and only if its encoding contains a run of k 0's within the last n-1 positions. Thus we have established that

(# q-ary strings of length n without a run of k identical characters)

= q * (# q-ary strings of length n-1 without a run of k-1 0's) .

Since we have already established the recurrence relation for q-ary strings without runs of 0's, the result follows.

Finally we consider q-ary strings without any length k runs of identical characters or alternating characters.

```
> MinRecRelAnyBase({"00000", "11111", "22222", "01010",
"10101", "02020", "20202", "12121", "21212"},3);
```

{"00000", "01010", "02020", "10101", "11111", "12121", "20202", "21212", "22222"
}, $a(n) = 2\, a(n-1) + 2\, a(n-2) + 2\, a(n-3)$,

[81, 234, 684, 1998, 5832, 17028, 49716, 145152, 423792, 1237320, 3612528]

From this example, and others, it appears that, in general, the number of strings over a q-ary alphabet with no run of k consecutive identical characters or alternating characters has the recurrence relation

$$a(n) = (q-1)*a(n-1) + (q-1)*a(n-2) + (q-1)*a(n-3) + \dots + (q-1)*a(n-k+2).$$

We will illustrate the proof of this recurrence relation by examining the cases $q = 4$ and $q = 3$ with a more general encoding method.

For q = 4: Every 4-ary string may be represented uniquely as a string of the form

0 - 3	p(1)	p(2)	p(3)	....	p(n-1)
1	2	3	4		n

where each p(i) is one of the four permutations: A = (1)(2)(3)(4) , B = (1, 2)(3, 4) , C = (1, 3)(2, 4) , D = (1, 4)(2, 3) . Each permutation provides the mapping that transforms the previous character in the string into the next. A run of k-1 A's corresponds to a string with a run of k identical characters, while a run of k-1 B's, C's or D's corresponds to a string with an alternating run of length k. Thus we have:

(# 4-ary strings of length n without a run of k identical or alternating characters)

= 4 * (# 4-ary strings of length n-1 without a run of k-1 identical characters)

Since we have already established the recurrence relation for 4-ary strings without a run of k identical characters is $a(n) = (4-1)*a(n-1) + (4-1)*a(n-2) + (4-1)*a(n-3) + \dots + (4-1)*a(n-k+1)$, the result follows.

For q = 3: The same approach works as for q = 4. We need only use a different set of permutations for encoding. Every 3-ary string may be represented uniquely as a string of the form

0 - 2	p(1)	p(2)	p(3)	....	p(n-1)
1	2	3	4		n

where each p(i) is one of the three permutations: A = (1)(2, 3) , B = (2)(1, 3) , C = (3)(1, 2) . As before, length k-1 runs of A, B, or C correspond to runs of identical or alternating characters, and the recurrenc relation follows.

In general, the set of permutations to use for q is a generalization of the case q=4 above, for q even, and q=3 above, for q odd. It involves known graph theoretic results about decompositions of the complete graph on q vertices.