

Termination Proofs for the Ducci Programs

Program Verification & Analysis

June 25, 2026

Abstract

This file provides some additional information about the termination proofs for the programs based on Ducci-like iterations. Starting with the original Ducci sequence with tuples of four unsigned integers, we consider also variants using signed integers and other sizes of tuples.

1 Ducci Sequences

A Ducci sequence (aka the Diffy game) is an iteration of tuples of integers which are obtained by an arbitrary initial tuple and the iteration $(a_1, a_2, \dots, a_n) \rightarrow (|a_1 - a_2|, |a_2 - a_3|, \dots, |a_n - a_1|)$. Since the entries of the tuples are combined with their successors and the final value's successor is the first entry, one may also place the entries of the tuple on a circle so that the values are then replaced with their difference to the next value. Ducci sequences are named after Enrico Ducci (1864–1940), the Italian mathematician who discovered in the 1930s that every such sequence based on unsigned integers eventually becomes periodic. The reason for this matter of fact is not difficult to see since the maximum value of the tuples cannot increase. For this reason, the set of potential tuples that can occur in the sequence is finite, since it is contained in the set $\{0, \dots, \max(a_1, a_2, \dots, a_n)\}^n$. For this reason, any infinite sequence of such tuples must become eventually periodic. Ducci sequences often appear as mathematical puzzles and were considered in research papers such as [7, 5, 3, 4, 1, 2].

2 DucciNat4: 4-Tuples of Unsigned Numbers

The classic Ducci sequence is typically the one with 4-tuples of unsigned integers, i.e., the sequence computed by the following synchronous reactive program:

```
module DucciNat4(nat ?aIn, ?bIn, ?cIn, ?dIn, event !rdy) {
  nat a, b, c, d, m1, m2, m;
  a = aIn;
  b = bIn;
  c = cIn;
  d = dIn;
  while(a!=0 | b!=0 | c!=0 | d!=0) {
    m1 = (a<b ? b : a);
    m2 = (c<d ? d : c);
    m = (m1<m2 ? m2 : m1);
    next(a) = (a<b ? b-a : a-b);
    next(b) = (b<c ? c-b : b-c);
    next(c) = (c<d ? d-c : c-d);
    next(d) = (d<a ? a-d : d-a);
    p:pause;
  }
  m1 = (a<b ? b : a);
  m2 = (c<d ? d : c);
  m = (m1<m2 ? m2 : m1);
  emit(rdy);
}
```

It is not difficult to prove that after four iterations, all numbers a, b, c, d will be even numbers, and after further four iterations, they are all divisible by 4, and in general, after $4k$ iterations, the numbers are divisible by 2^k . Since the numbers cannot grow beyond the maximum value of the initial values, it follows that the sequence must finally end with the zero tuple.

Since this classic proof does not directly provide a ranking function that is required for proving the termination of the above program, we therefore used another proof idea. First of all, we can quickly prove that the maximum m cannot grow, and second, we determine the 4-tuples where the maximum remains equal for one, two, three, and four iterations. We prove that the tuples which maintain the maximum value for one iteration are exactly those containing a sub-tuple $(m, 0)$ or $(0, m)$ when the tuple is considered as placed on a circle.

- (1) $m0^{**} \rightarrow m^{***}$
- (2) $0m^{**} \rightarrow m^{***}$

The tuples which maintain the maximum value for two iterations are exactly the following ones which represent 16 cases that are obtained by cyclic shifts of the listed tuples:

- (1) $m00^{*} \rightarrow m0^{**} \rightarrow m^{***}$
- (2) $mm0^{*} \rightarrow 0m^{**} \rightarrow m^{***}$
- (3) $00m^{*} \rightarrow 0m^{**} \rightarrow m^{***}$
- (4) $0mm^{*} \rightarrow m0^{**} \rightarrow m^{***}$

Only a few 4-tuples maintain the maximum value for three iterations, and these are exactly the following ones (with cyclic shifts these are 8 4-tuples):

- (1) $m000 \rightarrow m00m \rightarrow m0m0 \rightarrow mmmm \rightarrow 0000$
- (2) $mmm0 \rightarrow 00mm \rightarrow 0m0m \rightarrow mmmm \rightarrow 0000$

Finally, there is no 4-tuple which can maintain the maximum value for four iterations. For this reason, the 4-tuple $(m, \text{next}(m), \text{next}(\text{next}(m)), \text{next}(\text{next}(\text{next}(m))))$ can be used as a valid ranking function to prove the termination of `DucciNat4` since we must have for any 4-tuple one of the following cases:

- (1) $\text{next}(m) < m$
- (2) $\text{next}(m) == m \ \& \ \text{next}(\text{next}(m)) < \text{next}(m)$
- (3) $\text{next}(m) == m \ \& \ \text{next}(\text{next}(m)) == \text{next}(m) \ \& \ \text{next}(\text{next}(\text{next}(m))) < \text{next}(\text{next}(m))$
- (4) $\text{next}(m) == m \ \& \ \text{next}(\text{next}(m)) == \text{next}(m) \ \& \ \text{next}(\text{next}(\text{next}(m))) == \text{next}(\text{next}(m)) \ \& \ \text{next}(\text{next}(\text{next}(\text{next}(m)))) < \text{next}(\text{next}(\text{next}(m)))$

Once the theorems asserting that the maximum value is maintained for the mentioned set of 4-tuples for one, two, and three iterations, and proving that no 4-tuple can maintain its maximum value for four iterations, the proof can be obtained with the usual proof rule using the ranking function $(m, \text{next}(m), \text{next}(\text{next}(m)), \text{next}(\text{next}(\text{next}(m))))$. This is a nice example to demonstrate the need for ranking functions based on tuples instead of single natural numbers. Moreover, a big challenge for the proof is the exponential growth of the expressions if we would expand them over four iterations. This can be avoided by first proving the following transition equations which are then instantiated for as many points of time as required for the proofs:

$$\begin{aligned} G(\ & (\text{next}(a) == (a < b ? b - a : a - b)) \ \& \\ & (\text{next}(b) == (b < c ? c - b : b - c)) \ \& \\ & (\text{next}(c) == (c < d ? d - c : c - d)) \ \& \\ & (\text{next}(d) == (d < a ? a - d : d - a)) \) \end{aligned}$$

3 DucciInt4: 4-Tuples of Signed Numbers

A reasonable variant of DucciNat4 is the following program DucciInt4 which uses signed integers instead of unsigned integers:

```
module DucciInt4(int ?aIn,?bIn,?cIn,?boundIn,event !rdy) {
  int a,b,c,d,bound;
  a = aIn;
  b = bIn;
  c = cIn;
  d = cIn-1;
  bound = boundIn;
  do {
    next(a) = a - b;
    next(b) = b - c;
    next(c) = c - d;
    next(d) = d - a;
    p:pause;
  } while(a<bound & b<bound & c<bound & d<bound);
  emit(rdy);
}
```

We will see that the 4-tuples will contain at least one entry that grows beyond any upper bound, and also at least one entry that shrinks below any lower bound. Since we cannot easily determine which of the entries are of the one or the other kind, we have an additional difficulty to prove the termination of the above program for all possible 4-tuples. We have to exclude the t-tuples where all values are the same, and this is done by assigning $d = cIn-1$.

Before discussing a ranking function, we have to prove some loop invariants. A first loop invariant is the sum $a + b + c + d$:

$$\text{next}(a + b + c + d) == (a - b) + (b - c) + (c - d) + (d - a) == 0$$

This yields a first fundamental loop invariant:

$$\text{Invariant1: } G \text{ (!start} \rightarrow a+b+c+d == 0)$$

For all initial values (a, b, c, d) provided by the environment, the tuple collapses into the zero-sum hyperplane after one iteration and remains there forever. Because of this invariant, we also get $(a + b + c + d)^2 = 0$ after the initialization, and therefore

$$\begin{aligned} 0 &= (a + b + c + d)^2 \\ &= ((a + c) + (b + d))^2 \\ &= (a + c)^2 + (b + d)^2 + 2 \cdot (a + c) \cdot (b + d) \\ &\Leftrightarrow \\ &-2 \cdot (a + c) \cdot (b + d) = (a + c)^2 + (b + d)^2 \end{aligned}$$

As part of a ranking function, we consider now the sum of squares $a^2 + b^2 + c^2 + d^2$ and get the following with the above equation (note that $(a + c)^2 + (b + d)^2 \geq 0$ holds):

$$\begin{aligned} &\text{next}(a^2 + b^2 + c^2 + d^2) \\ &= \text{next}(a)^2 + \text{next}(b)^2 + \text{next}(c)^2 + \text{next}(d)^2 \\ &= (a - b)^2 + (b - c)^2 + (c - d)^2 + (d - a)^2 \\ &= 2 \cdot (a^2 + b^2 + c^2 + d^2) - 2 \cdot (a \cdot b + b \cdot c + c \cdot d + a \cdot d) \\ &= 2 \cdot (a^2 + b^2 + c^2 + d^2) - 2 \cdot (a + c) \cdot (b + d) \\ &= 2 \cdot (a^2 + b^2 + c^2 + d^2) + (a + c)^2 + (b + d)^2 \\ &\geq 2 \cdot (a^2 + b^2 + c^2 + d^2) \end{aligned}$$

We therefore have the following invariants:

$$I_2 : \mathbf{G}(\neg \text{start} \wedge p \wedge \neg \text{rdy} \rightarrow \text{next}(a^2 + b^2 + c^2 + d^2) = 2 \cdot (a^2 + b^2 + c^2 + d^2) + (a+c)^2 + (b+d)^2) \quad (1)$$

$$I_3 : \mathbf{G}(\neg \text{start} \wedge p \wedge \neg \text{rdy} \rightarrow \text{next}(a^2 + b^2 + c^2 + d^2) \geq 2 \cdot (a^2 + b^2 + c^2 + d^2)) \quad (2)$$

Due to invariant I_3 , the sum of squares $a^2 + b^2 + c^2 + d^2$ doubles in each iteration, so that it will exponentially grow provided that this sum is initially not zero. Due to the initial assignments, and the invariant I_3 , we can easily prove the following invariant by induction:

$$I_4 : \mathbf{G}(a^2 + b^2 + c^2 + d^2 \geq 1) \quad (3)$$

Having invariants I_3 and I_4 , we can now prove that the sum of squares is strictly growing in each loop iteration:

$$I_5 : \mathbf{G}(\neg \text{start} \wedge p \wedge \neg \text{rdy} \rightarrow \text{next}(a^2 + b^2 + c^2 + d^2) > (a^2 + b^2 + c^2 + d^2)) \quad (4)$$

We therefore see that the sum of squares is strictly growing in each loop iteration, and has even an exponential growth due to I_3 .

However, these invariants do not yet allow us to conclude that the program terminates for all inputs. To derive a ranking function which is strictly decreasing, we consider a function $\varrho := M - (a^2 + b^2 + c^2 + d^2)$ with a constant M that is large enough so that this difference will never become negative (since the subtraction is defined on natural numbers and requires this condition to be well-defined). We therefore have to find an upper bound M for the sum of squares.

Assuming that rdy holds, we can derive that $a < \text{bound}$, $b < \text{bound}$, $c < \text{bound}$, and $d < \text{bound}$ holds. Due to I_4 , it is impossible that all four variables are zero, and due to I_1 , it is impossible that all four variables have the same sign.

Assume that one variable is negative and the other three are positive, let's say $d = -(a+b+c)$ is negative while $a, b, c > 0$ holds. Then, we get

$$a^2 + b^2 + c^2 + d^2 < \text{bound}^2 + \text{bound}^2 + \text{bound}^2 + (3 \cdot \text{bound})^2 = 12 \cdot \text{bound}^2$$

If two variables are negative and the other two are positive, let's say $c, d < 0 < a, b$, and $a + b = -(c + d)$, we get $a^2 + 2ab + b^2 = c^2 + 2cd + d^2$, and therefore

$$\begin{aligned} a^2 + b^2 + c^2 + d^2 &= a^2 + b^2 + a^2 + 2ab + b^2 + 2cd \\ &= 2 \cdot (a^2 + b^2 + ab + cd) \\ &< 4 \cdot \text{bound}^2 \end{aligned}$$

We can therefore prove the following invariant:

$$I_6 : \mathbf{G}(\neg \text{start} \wedge p \wedge \neg \text{rdy} \rightarrow (a^2 + b^2 + c^2 + d^2 < 12 \cdot \text{bound}^2)) \quad (5)$$

As a ranking function, we need however $\varrho := 24 \cdot \text{bound}^2 - a^2 + b^2 + c^2 + d^2$ since we have $a^2 + b^2 + c^2 + d^2 < 12 \cdot \text{bound}^2$ right before the termination point of time, so that the loop will iterate once more and may double again the sum of squares according to invariant I_3 . Using the above mentioned invariants and the mentioned ranking function, we can easily prove the termination of the program.

Using the insufficient ranking function $12 \cdot \text{bound}^2 - a^2 + b^2 + c^2 + d^2$ yields the counterexample $(a, b, c, d) = (44, -87, 43, 0)$ with $\text{bound} = 45$ which satisfies the invariants, but if another iteration takes place, we get $\text{next}(a, b, c, d) = (131, -130, 43, -44)$ whose sum of squares is larger than $12 \cdot \text{bound}^2$, but less than $24 \cdot \text{bound}^2$.

Google Gemini pointed out that there is a nice way to derive the nature of the program since each iteration can be described as a matrix multiplication with the following matrix:

$$\begin{pmatrix} \text{next}(a) \\ \text{next}(b) \\ \text{next}(c) \\ \text{next}(d) \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \quad (6)$$

According to the power iteration principle (e.g., see [6, page 491]), the tuple (a, b, c, d) converges to the eigenvector (v_a, v_b, v_c, v_d) of the eigenvalue with greatest absolute value. We therefore determine the eigenvalues of the above matrix M with its characteristic equation:

$$\det(M - \lambda I) = 0 \implies (\lambda - 1)^4 - 1 = 0 \quad (7)$$

Taking the fourth roots of unity yields:

$$\lambda - 1 \in \{1, -1, i, -i\} \quad (8)$$

Thus, the four eigenvalues are:

$$\begin{aligned} \lambda_0 &= 2 \\ \lambda_1 &= 0 \\ \lambda_2 &= 1 + i \\ \lambda_3 &= 1 - i \end{aligned}$$

The eigenvalue with greatest absolute value is $\lambda_0 = 2$, and the corresponding eigenvector is $(1, -1, 1, -1)$. Therefore, the tuple (a, b, c, d) of variables of the program will converge to a multiple of $(1, -1, 1, -1)$ with a factor 2^k in iteration k , so that we will quickly have the following:

$$a \approx -b \approx c \approx -d \quad (9)$$

As the growth rate is determined by $\lambda_0 = 2$ the sum of squares doubles in each iteration. The only initial values that fail to diverge exponentially are $a = b = c = d$ which are excluded by the initial assignment of d .

4 DucciNat3: 3-Tuples of Signed Numbers

Another reasonable variant of DucciNat4 is the following program DucciNat3 which uses 3-tuples of unsigned integers:

```
module DucciNat3(nat ?aIn,?bIn,?cIn,event !rdy) {
  nat a,b,c,m1,m;
  a = aIn;
  b = bIn;
  c = cIn;
  while(a!=0 | b!=c) {
    m1 = (a<b ? b : a);
    m = (m1<c ? c : m1);
    next(a) = (a<b ? b-a : a-b);
    next(b) = (b<c ? c-b : b-c);
    next(c) = (c<a ? a-c : c-a);
    p:pause;
  }
  m1 = (a<b ? b : a);
  m = (m1<c ? c : m1);
  emit(rdy);
}
```

We know that any Ducci sequence using unsigned integers must become eventually periodic. The period of `DucciNat4` is just one and the final value obtained is the zero tuple. For `DucciNat3`, we obtain a period of length 3 so that another loop condition is required to guarantee the termination of the program.

Following the proof ideas of `DucciNat4`, we first prove that the maximum value cannot grow during the loop iterations. Then, we determine the set of 3-tuples where the maximum value is maintained for one, two, and three iterations by proving the following theorems:

```

MaxEqual1:
  G((next(m) == m)
    <-> !(start|p) |
      a==m & (b==0 | c==0) |
      b==m & (c==0 | a==0) |
      c==m & (a==0 | b==0) ))

MaxEqual2
  G((next(next(m)) == m)
    <-> !(start|p) |
      a==m & b==0 & c==0 |
      b==m & c==0 & a==0 |
      c==m & a==0 & b==0 |
      a==0 & b==m & c==m |
      b==0 & c==m & a==m |
      c==0 & a==m & b==m ))

MaxEqual3
  G((next(next(next(m))) == m) <-> (next(next(m)) == m) )

MaxEqual
  G((next(next(m)) == m) -> G(next(next(m))==m) )

```

The sequence must therefore end in its period $(m, m, 0) \rightarrow (0, m, m) \rightarrow (m, 0, m) \rightarrow (m, m, 0) \rightarrow \dots$. The remaining three 3-tuples which can maintain the maximum value forever have transitions to one of the three states of the period, i.e. $(m, 0, 0) \rightarrow (m, 0, m)$, $(0, m, 0) \rightarrow (m, m, 0)$, and $(0, 0, m) \rightarrow (0, m, m)$. However, it is unclear which one of the three states of the period is entered first, and we may even enter one of the other three states $(m, 0, 0)$, $(0, m, 0)$, $(0, 0, m)$ which enter the period only in the next transition.

Adapting the ranking function used for program `DucciNat4` to $(m, \text{next}(m), \text{next}(\text{next}(m)))$ will not work since the maximum value may not decrease in every transition. However, we know by the above theorems that the maximum value either decreases within the next two iterations, or will be maintained forever. In the latter case, the period of the sequence must be entered, and this may require no more than four steps to reach the state $(0, m, m)$ that causes the termination of the loop. A suitable ranking function is therefore a tuple $(m, \text{next}(m), \text{dist})$ where `dist` is the expression below that measures the distance from a state which maintains the maximum forever to the termination state $(0, m, m)$:

```

(a==m & b==0 & c==0 ? 3 :
(a==0 & b==m & c==0 ? 2 :
(a==0 & b==0 & c==m ? 1 :
(a==m & b==m & c==0 ? 1 :
(a==m & b==0 & c==m ? 2 :
(a==0 & b==m & c==m ? 0 : 4))))))

```

5 General DucciNatX Programs

The appendix shows reduction sequences of n -tuples containing a maximum value m and the number 0 for n in $\{3, 4, 5, 6, 7, 8\}$ to determine the periodic behavior of the Ducci sequences using tuples of size n . Note that the maximum value may also be maintained if we have patterns

$*m0*$ and $*0m*$, but we can represent the periods nevertheless with zeros and the maximum value. We can see that for $n = 2^k$, we obtain binary trees which collapse into the zero tuple, while for other numbers n , we obtain true periods, and sometimes even different periods of different lengths.

For instance, for `DucciNat5`, all paths must eventually enter either the cycle of length 15 of the self-loop on the zero tuple. Moreover, all states maintaining their maximum value outside these cycles must enter one of these cycles in one step. Hence, for the termination proof, we would have to prove that if the maximum is maintained for one iteration, it is maintained forever since we already must have entered a cycle with the next transition. The ranking function would then have the form (m, dist) with a distance function that would map the states to the distance to those states which will satisfy the termination condition (if so).

For `DucciNat6`, there are four cycles, the self-loop on the zero tuple, two cycles of length six, and one cycle of length three. The longest paths maintaining the maximum value without the cyclic states have length two so that the ranking function can be defined as a tuple $(m, \text{next}(m), \text{dist})$ with a suitable distance function, and in a similar way, we can construct ranking functions for any program `DucciNatX` provided the termination condition would satisfy at least one state of every cycle of its state transition diagram.

6 General DucciIntX Programs

For any `DucciIntX` program, we first deduce that the sum of the entries of the tuples are zero after the first iteration, and tuples consisting of equal entries collapse into the zero tuple. Second, we determine the corresponding transition matrix which is a circulant matrix, so that its eigenvalues and eigenvectors correspond with the Fourier matrix of size n . In particular, its characteristic polynomial is $(1 - \lambda)^n - 1 = 0$ so that the n eigenvalues satisfy $\lambda_i := 1 - \sqrt[n]{1}$, i.e., we therefore obtain the following eigenvalues and their absolute values for $k = 0, \dots, n-1$:

$$\lambda_k := 1 - \cos\left(\frac{2\pi k}{n}\right) - i \cdot \sin\left(\frac{2\pi k}{n}\right) \text{ with } |\lambda_k| = \sqrt{2 - 2 \cdot \cos\left(\frac{2\pi k}{n}\right)}$$

Hence, for even $n = 2m$, the greatest absolute value is $|\lambda_m| = 2$, and its eigenvector is $(1, -1, 1, -1, \dots)$. For odd $n = 2m+1$, the greatest absolute value is $|\lambda_m| = \sqrt{2 - 2 \cdot \cos\left(\frac{2m}{2m+1}\pi\right)}$ which is $\sqrt{3}$ for $n = 3$, and grows with n towards 2. For example, for the `DucciInt3` program, we obtain the following spectrum of eigenvalues:

$$\lambda_0 = 0 \quad \lambda_1 = 1 - \cos\left(\frac{2\pi}{3}\right) - i \cdot \sin\left(\frac{2\pi}{3}\right) = \frac{3}{2} - \frac{\sqrt{3}}{2}i \quad \lambda_2 = 1 - \cos\left(\frac{4\pi}{3}\right) - i \cdot \sin\left(\frac{4\pi}{3}\right) = \frac{3}{2} + \frac{\sqrt{3}}{2}i$$

Computing the null spaces for each eigenvalue yields the corresponding linearly independent eigenvectors v_0, v_1, v_2 :

$$v_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad v_1 = \begin{bmatrix} -\frac{1}{2} + \frac{\sqrt{3}}{2}i \\ -\frac{1}{2} - \frac{\sqrt{3}}{2}i \\ 1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} -\frac{1}{2} - \frac{\sqrt{3}}{2}i \\ -\frac{1}{2} + \frac{\sqrt{3}}{2}i \\ 1 \end{bmatrix} \quad (10)$$

The geometric interpretation explains the execution of the program in general:

- *The Invariant Projection* ($\lambda_0 = 0$): The corresponding eigenvector $v_0 = [1, \dots, 1]^T$ represents the tuples with equal entries. Because its corresponding eigenvalue λ_0 is 0, any initial state tuple of this state set collapses into the zero tuple after the first transition. This also explains why the zero-sum invariant holds for all $t \geq 1$.

- *Exponential Growth and Spiral Divergence:* The remaining eigenvalues $\lambda_{1,2}$ determine the exponential growth of the values: Since we have $\sqrt{3} \leq |\lambda_i| \leq 2$, any non-zero state tuple will experience an exponential growth with the factor $\max\{|\lambda_i| \mid i = 1, \dots, n-1\}$. Moreover, the corresponding eigenvectors enforce a spiral behavior on the 2D plane orthogonal to the eigenvector per macrostep.

In summary, we can say that the sum of squares (measuring the distance to the origin) grows with a factor of $|\lambda_m|$ in every step, and since the sum of the tuple's entries is zero, there are entries which grow over any positive bound and others which shrink below any negative bound. Hence, as long as the initial inputs are not all equal (which would directly be mapped to the zero tuple), the geometric divergence ensures that at least one entry of the tuple will eventually grow over any bound, forcing the loop to terminate.

7 General DucciBoolX Programs

Consider a program `DucciBoolN` operating over the finite field $\mathbb{Z}_2$ (Boolean algebra where addition corresponds to the XOR operator $\oplus$). For instance, `DucciBool9` is as follows:

```
module DucciBool9(bool ?x0,x1,x2,x3,x4,x5,x6,x7, event !rdy) {
  bool a0,a1,a2,a3,a4,a5,a6,a7,a8;
  a0 = x0;
  a1 = x1;
  a2 = x2;
  a3 = x3;
  a4 = x4;
  a5 = x5;
  a6 = x6;
  a7 = x7;
  a8 = !x7;
  while(a0|a1|a2|a3|a4|a5|a6|a7|a8) {
    next(a0) = a0 xor a1;
    next(a1) = a1 xor a2;
    next(a2) = a2 xor a3;
    next(a3) = a3 xor a4;
    next(a4) = a4 xor a5;
    next(a5) = a5 xor a6;
    next(a6) = a6 xor a7;
    next(a7) = a7 xor a8;
    next(a8) = a8 xor a0;
    p:pause;
  }
  emit(rdy);
}
```

Let the state vector at loop iteration k be denoted by:

$$v^{(k)} = \begin{pmatrix} a_0^{(k)} & a_1^{(k)} & \dots & a_{n-1}^{(k)} \end{pmatrix}^T \in \mathbb{Z}_2^n \quad (11)$$

The loop condition specifies that termination occurs if and only if $v^{(k)} = \mathbf{0}$, where $\mathbf{0}$ is the zero vector. The state transition within the loop is defined by:

$$a_i^{(k+1)} = a_i^{(k)} \oplus a_{i+1}^{(k)} \pmod{n} \quad \forall i \in \{0, 1, \dots, n-1\} \quad (12)$$

This system can be represented as a linear transformation:

$$v^{(k+1)} = Tv^{(k)} = (I + P) \cdot v^{(k)} \pmod{2} \quad (13)$$

where I is the $n \times n$ identity matrix and P is the cyclic permutation (left-shift) matrix satisfying $P^n = I$. To analyze the program, we exploit the following characteristic 2 property of the field $\mathbb{Z}_2$.

Theorem 1 (Freshman's Dream in $\mathbb{Z}_2$). *For any linear operators A and B over $\mathbb{Z}_2$ that commute, and for any power of two $m = 2^j$, $(A + B)^{2^j} = A^{2^j} + B^{2^j}$.*

Applying the above fact to the transition matrix T :

$$T^{2^j} = (I + P)^{2^j} = I^{2^j} + P^{2^j} = I + P^{2^j} \quad (14)$$

Hence, we seek a condition under which the transformation becomes periodic and satisfies $T^{2^j} = T$. This requires $I + P^{2^j} = I + P$, which simplifies to:

$$P^{2^j} = P^1 \implies 2^j \equiv 1 \pmod{n} \quad (15)$$

Theorem 2 (Condition for Immediate Periodicity). *A power $j \geq 1$ satisfying $2^j \equiv 1 \pmod{n}$ exists if and only if n is an odd integer.*

Proof. If n is odd, $\gcd(2, n) = 1$. By Euler's Totient Theorem, $2^{\phi(n)} \equiv 1 \pmod{n}$, ensuring a valid multiplicative order j exists. Conversely, if n is even, $2^j \pmod{n}$ can never equal 1 since 2^j is always even or zero modulo an even number. $\square$

When n is odd, let $m = 2^j$. Then $T^m = T$. For any state after the first step ($k \geq 1$), we have:

$$T^k v^{(0)} = T^{m-1} (T^k v^{(0)}) \quad (16)$$

This reveals that if any state reaches the all-zeros vector $\mathbf{0}$ at step $k \geq 1$, it must be that $T(v^{(0)}) = \mathbf{0}$. Therefore, no states can latent-collapse into $\mathbf{0}$ after multiple distinct steps unless they were already mapping to $\mathbf{0}$ in exactly one step. For this reason, the program terminates if and only if it terminates after the first iteration. To discover which states can trigger termination, we solve the kernel equation $Tv = \mathbf{0}$:

$$a_i \oplus a_{i+1} \pmod{n} = 0 \implies a_0 = a_1 = a_2 = \dots = a_{n-1} \quad (17)$$

Thus, the only preimages capable of transitioning into the terminating state are the following vectors which are however excluded by the initialization of the variables:

1. **The All-Zeros Vector:** $\mathbf{0} = (0, 0, \dots, 0)^T$
2. **The All-Ones Vector:** $\mathbf{1} = (1, 1, \dots, 1)^T$

Therefore, we have the following for an arbitrary vector size n :

1. If n is even, the system does not satisfy $2^j \equiv 1 \pmod{n}$, and termination can occur.
2. If n is odd, the system forms a closed periodic orbit excluding $\mathbf{0}$ and $\mathbf{1}$ unless initialized inside them.
3. The structural initialization $a_{n-1} = \neg a_{n-2}$ guarantees the state starts outside these preimages.

Therefore, the program `DucciBoolN` is guaranteed to ****never terminate**** if and only if n is an odd number.

References

- [1] BROCKMAN, G. Asymptotic behaviour of certain Ducci sequences. *Fibonacci Quarterly* 45, 2 (2007), 155–163.
- [2] BU, L. Ducci’s four-number game: Making sense of a classic problem using mobile simulation. *Journal of Humanistic Mathematics* 9, 2 (2019), 281–300.
- [3] CHAMBERLAND, M. Unbounded Ducci sequences. *Journal of Difference Equations and Applications* 9, 10 (2003), 887–895.
- [4] CHAMBERLAND, M., AND THOMAS, D. The N-number Ducci game. *Journal of Difference Equations and Applications* 10, 3 (March 2004), 339–342.
- [5] ENGEL, A. *Problem-Solving Strategies*. Problem Books in Mathematics. Springer, 1991.
- [6] EPPERSON, J. *An Introduction to Numerical Methods and Analysis*, 2 ed. Wiley, 2013.
- [7] FREEDMAN, B. The four number game. *Scripta Mathematica* 14 (1948), 35–47.

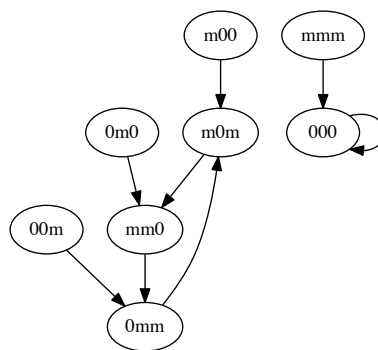


Figure 1: tuples Maintaining the Maximum or Collapsing for $n = 3$

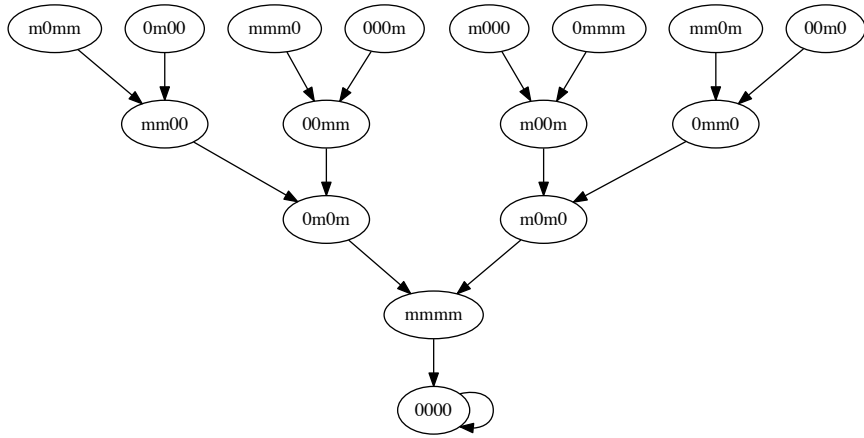


Figure 2: tuples Maintaining the Maximum or Collapsing for $n = 4$

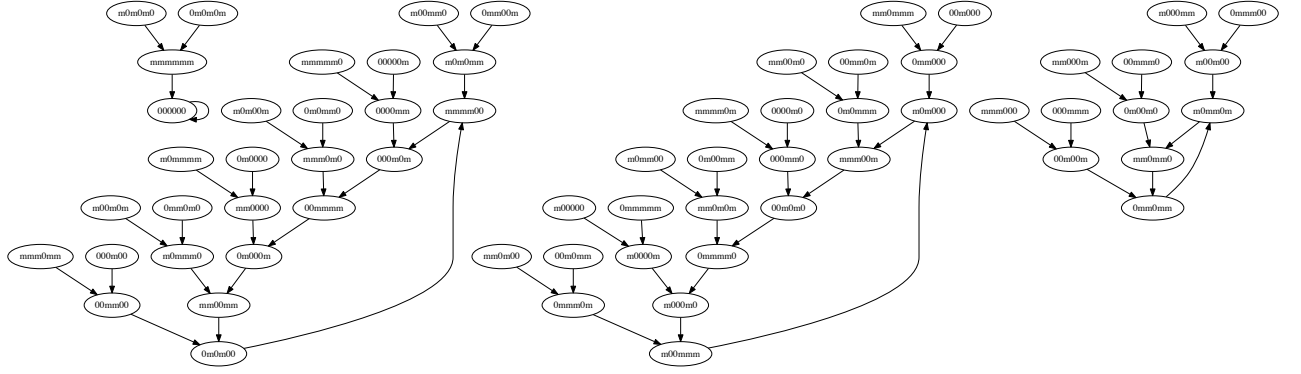


Figure 4: tuples Maintaining the Maximum or Collapsing for $n = 6$

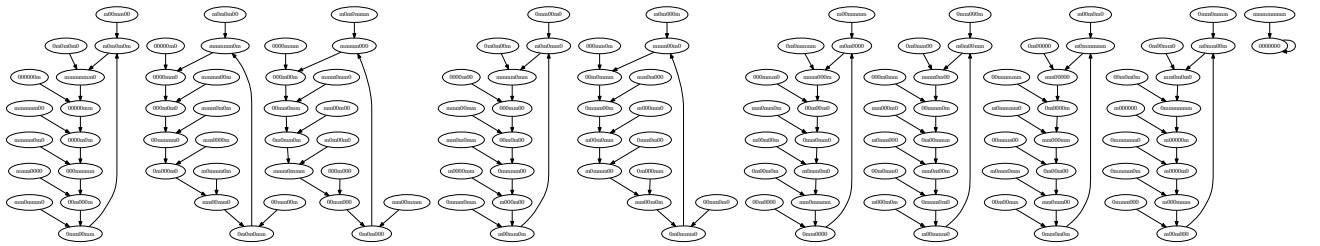


Figure 5: tuples Maintaining the Maximum or Collapsing for $n = 7$

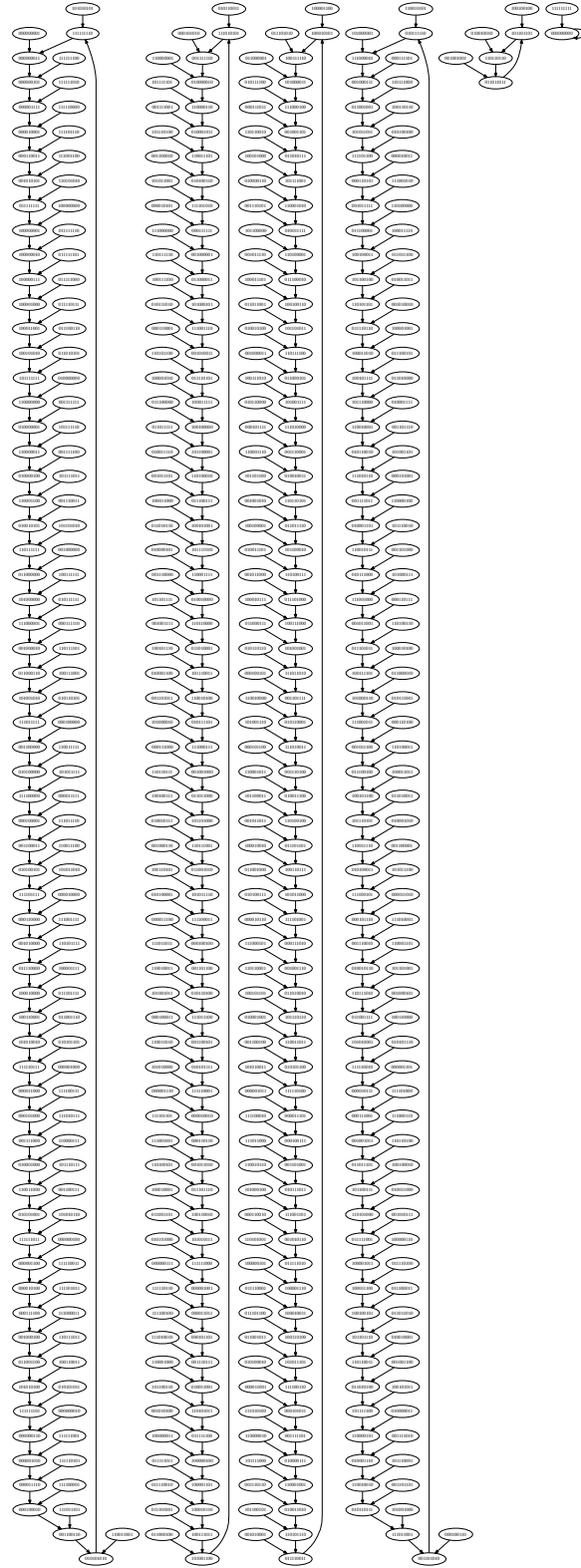


Figure 6: State transitions for DucciBool9