

## Solution: DASZ Operator

### Part A: Repeat

The key insight is that for any list  $L = (a_1, a_2, \dots, a_w)$  the application of  $D$  does not affect the weight of  $L$ :  $w(L) = \sum_i a_i$ . This is called an invariant. Since the entries  $a_i$  are non-negative there are only finitely many lists of a given weight, hence repeated application of  $D$  must ultimately result in a cycle:  $D^{t+p}(L) = D^t(L)$ .

### Part B: Fixed Points

Consider a list  $L = (a_1, a_2, \dots, a_n)$  assumed to be sorted in non-decreasing order. Suppose  $L$  is a fixed point. Since the length of  $D(L)$  is  $n - 1 + k$  where  $k$  is maximal such that  $a_k = 1$  we must have  $1 = a_1 < a_2$ . An easy induction then shows that  $a_i = i$ . It is clear that all lists  $(1, 2, 3, \dots, n - 1, n)$  are fixed points, done.

### Part C: To FPs

Since application of  $D$  does not affect weight, a fixed point of width  $m$  must have weight  $w = m(m + 1)/2 = T_m$ . Hence we only have to consider triangular numbers  $1, 3, 6, 10, 15, 21, \dots$  as starting points (the red dots at the bottom in the picture).

**Claim:** All the lists  $(T_m)$  evolve to their corresponding fixed points.

As a warm-up exercise, let's consider lists of the form  $L_k = (1, 2, \dots, k, \infty)$  where  $\infty$  stands for a sufficiently large number. Of course, we assume  $\infty - 1 = \infty$ .

**Claim 1:**  $L_k$  evolves to  $L_{k+1}$  in  $k + 1$  steps.

To see this, show by induction on  $0 \leq s \leq k$  that

$$D^s(L_k) = (1 + \delta_{1,s}, 2 + \delta_{2,s}, \dots, k + \delta_{k,s}, \infty)$$

where  $\delta_{i,s} = 0$  if  $i + s \leq k$  and 1 otherwise. Hence

$$D^k(L_k) = (2, 3, \dots, k, k + 1, \infty)$$

and in one more step we get  $L_{k+1}$ .

Note that  $\infty$  can be replaced by any number larger than all the other list elements. We write  $L_k(x)$  for the list obtained by replacing  $\infty$  by  $x$  in  $L_k$ . It is immediate from claim 1 that  $L_k(x)$  evolves to  $L_{k+1}(x - k - 1)$  in  $k + 1$  steps. A simple induction using claim 1 then shows that

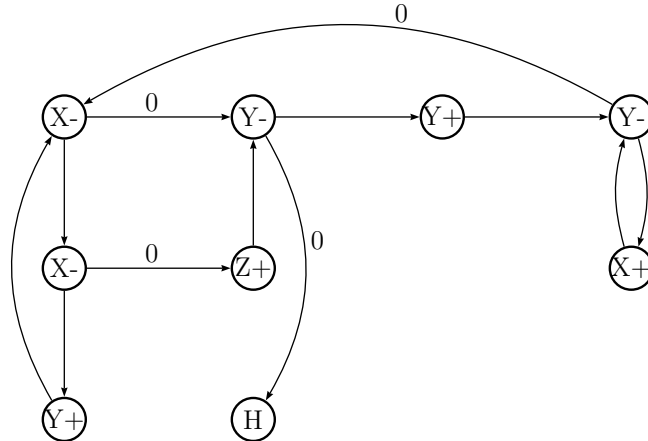
**Claim 2:**  $L_0(T_m)$  evolves to  $L_k(T_m - T_k)$  in  $T_k$  steps.

But then the main claim follows:  $L_0(T_m)$  is none other than the initial configuration  $(T_m)$ .

**Solution: RMs and Binary Digit Sums**

**Part A: Time Complexity**

Here is the flowgraph for digit sum again,  $X$  is input,  $Z$  is output and  $Y$  is an auxiliary variable.



From the graph, it is clear that the running time on input  $x$  should be roughly  $5x$ : there is one loop of length 3 and another of length 2.

But, the running time will also depend on  $\sigma(x)$  (the number of times the  $Z$  register is incremented) and  $|x|$ , the total number of binary digits of  $x$ . A fair guess is that the running time will be of the form

$$5x + a|x| + b\sigma(x) + c$$

where  $a$ ,  $b$  and  $c$  are constants. A little bit of experimentation then yields

$$5x + 4|x| - 3\sigma(x) - 2.$$

Once we have this answer it's not hard to argue that the program takes exactly this number of steps by diagram-chasing.

**Part B: Bit Sequence**

The sequence  $(t_n)$  is known as the the Prouhet-Thue-Morse sequence according to their (independent) inventors; it is exceedingly well-studied.

Clearly  $t_0 = 0 = \sigma(0) = \sigma(0) \bmod 2$ .

But  $\sigma(2x) = \sigma(x)$  and  $\sigma(2x + 1) = \sigma(x) + 1$ .

The claim follows immediately by modular induction.

**Part C: Shuffle**

Let  $S_{n-1} = a_1 a_2 \dots a_{k-1} a_k$ , so that

$$S_n = a_1 \overline{a_1} a_2 \overline{a_2} \dots a_{k-1} \overline{a_{k-1}} a_k \overline{a_k}$$

---

But then  $S_k$  satisfies the same recurrence as the Thue words  $T_{2^k}$ : just rewrite the definition in terms of shuffle. Hence the sequences must coincide.