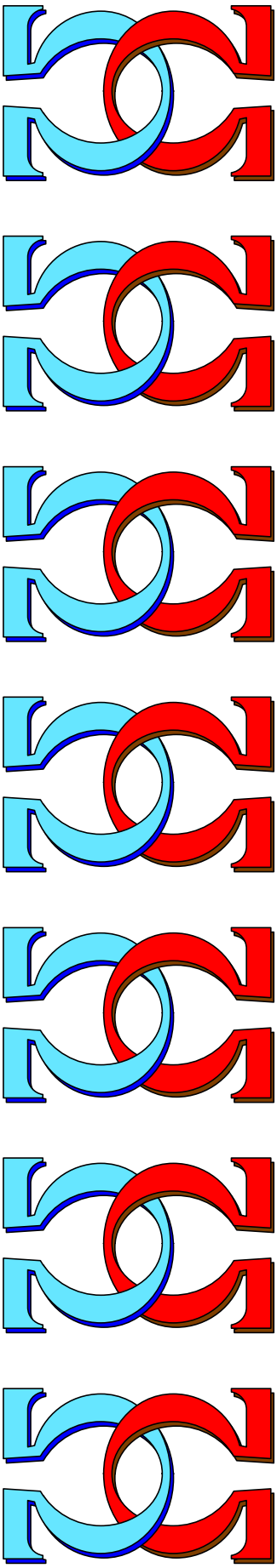

CDMTCS
Research
Report
Series

Uniform Candy Distribution

Sunny Daniels
Alumni, University of Auckland

CDMTCS-225
Oct 2003

Centre for Discrete Mathematics and
Theoretical Computer Science

Uniform Candy Distribution

Sunny Daniels,
Alumni, Department of Computer Science,
The University of Auckland,
sdaniels@eudoramail.com

October 28, 2003

1 The problem

The following puzzle was recently posted on "Alan and Danny's Puzzle Page" (http://www.cs.cmu.edu/puzzle/) by Professors Alan Frieze (Department of Mathematical Sciences: frieze@math.cmu.edu) and Daniel Sleator (School of Computer Science: sleator@cs.cmu.edu), both of Carnegie-Mellon University:

Puzzle 6: Uniform Candy Distribution

> n children are sitting around a circular table. Each child starts out with an integer number of candies. The following step is repeated:
>
> Every child who has an odd number of candies is given another piece of candy by the teacher. Each child now has an even number. Now every child passes half of his / her candy to the child on his / her left.
>
> *Prove that eventually all the children will have the same amount of candy.*

It seems reasonably clear from the explanatory note accompanying the puzzle that a solution to this puzzle might have, or might not have, been published in the literature:

> Every few weeks or so we'll put up a new puzzle. Some of them will be old classics, some of them will be new. Most will involve constructing an algorithm or a proof, while some will involve writing a computer program to solve them. After a few weeks, we'll post a solution, along with proper references to the problem.

I believe that I have solved this problem. My solution and a subsequent minor correction have been submitted to Professor Sleator on 14/10/2003 and 17/10/2003 respectively. I present my solution in this technical report, which also contains two experimental and discussion sections 3 and 7. They illustrate further how I arrived at the result. Note that, due to the nature of the challenge, I am unaware at the time of writing as to whether this problem has been solved and, if it has, whether a published solution exists. Related problems may have been discussed in the literature. Hence, this report describes my own solution only and makes no reference to other solutions that may have preceeded this one.

2 Re-grouping of solution steps

The 'algorithm' is formulated as an infinite sequence of steps, each consisting of two sub-steps:

1. Teacher gives an additional candy to every child with an odd number of candies. We'll call this a "top-up step".
2. Children redistribute candy (in way described in problem). We'll call this a "redistribute" step.

This is depicted in figure 1. However, we will see shortly that, for our purposes, it is better for us to treat the first top-up step differently from the rest; we regard the algorithm as a single top-up step, followed by an infinite sequence of "redistribute then top-up" steps. This is depicted in figure 2.

2.1 Amounts of candy are even between steps

Obviously each top-up step results in all children having an even number of candies. Hence, the children's numbers of candies are even both before and after each "redistribute then top-up" step.

3 An example

I wrote a simple C program, *candy*, to simulate this algorithm. I am happy to supply a copy of *candy* on request. Here is an example of the output from *candy*. In this example run, we assume that there are four children, and that they start with sixteen, two, eleven and nine candies, respectively:

```
sdaniels@wylie:~/candy$ ./candy 16 2 11 9
Initially                    :   16    2   11    9
After top-up 1               :   16    2   12   10
After redistribution 1       :    9    7   11   13
After top-up 2               :   10    8   12   14
After redistribution 2       :    9   10   13   12
After top-up 3               :   10   10   14   12
After redistribution 3       :   10   12   13   11
After top-up 4               :   10   12   14   12
After redistribution 4       :   11   13   13   11
After top-up 5               :   12   14   14   12
After redistribution 5       :   13   14   13   12
After top-up 6               :   14   14   14   12
After redistribution 6       :   14   14   13   13
After top-up 7               :   14   14   14   14
After redistribution 7       :   14   14   14   14
Candy now evenly distributed.
sdaniels@wylie:~/candy$
```

4 Maximum amount of candy held by any one child never decreases

Here we show that the "redistribute then top-up" step has the important property of never increasing the maximum number of candies held by a child.

4.1 Notation

To do this, we start by defining some notation. We use the residue classes modulo n as names for the children, in the obvious way:

- We arbitrarily assign $[0]_n$ to one child. $[0]_n$ refers to this same child throughout the execution of the algorithm.
- For any child $[k]_n$, $[k-1]_n$ is the child to $[k]_n$'s left.
- For any child $[k]_n$, $[k+1]_n$ is the child to $[k]_n$'s right.

For an *arbitrary* "redistribute then top-up" step S:

- For any child $[k]_n$, we let $B_{[k]_n}$ denote the number of candies held by $[k]_n$ *before* S. (We observed in section 2.1 that this number $B_{[k]_n}$ is always even).

- For any child $[k]_n$, we let $A_{[k]_n}$ denote the number of candies held by $[k]_n$ *after* S. (We observed in section 2.1 that this number $A_{[k]_n}$ is always even).

(We assume that it is always clear from the context as to which "redistribute then top-up" step we are talking about).

4.2 Proof

Take an arbitrary child $[k]_n$, and an arbitrary "redistribute then top-up" step. In the "redistribute" sub-step, $[k]_n$ loses half of its candy, but gains half of $[k+1]_n$'s candy. Hence, $[k]_n$'s amount of candy after the "redistribute" step is:

$$\frac{B_{[k]_n}}{2} + \frac{B_{[k+1]_n}}{2}$$

i.e.

$$\frac{B_{[k]_2} + B_{[k+1]_n}}{2}$$

So, if we use $\left\lceil \cdot \right\rceil$ to denote the action of rounding an integer up to the next even integer, we have:

$$A_{[k]_n} = \left\lceil \frac{B_{[k]_n} + B_{[k+1]_n}}{2} \right\rceil$$

Now, there are two cases:

1. $B_{[k]_n} = B_{[k+1]_n}$: Then:

   $$A_{[k]_n} = \left\lceil \frac{B_{[k]_n} + B_{[k]_n}}{2} \right\rceil = \left\lceil B_{[k]_n} \right\rceil$$

   which equals $B_{[k]_n}$ since $B_{[k]_n}$ is even. Hence $A_{[k]_n} = B_{[k]_n} = B_{[k+1]_n}$, so $A_{[k]_n} \leq B_{[k]_n}$ and $A_{[k]_n} \leq B_{[k+1]_n}$.

2. $B_{[k]_n} \neq B_{[k+1]_n}$: Assume WLOG that $B_{[k]_n} < B_{[k+1]_n}$. Since both $B_{[k]_n}$ and $B_{[k+1]_n}$ are even, we must have $B_{[k]_n} \leq B_{[k+1]_n} - 2$. Hence:

   $$\begin{aligned} \frac{B_{[k]_n} + B_{[k+1]_n}}{2} &\leq \frac{(B_{[k+1]_n} - 2) + B_{[k+1]_n}}{2} \\ &= \frac{2B_{[k+1]_n} - 2}{2} \\ &= B_{[k+1]_n} - 1 \end{aligned}$$

   Now, the top-up sub-step, at worst, adds one to $\frac{B_{[k]_n} + B_{[k+1]_n}}{2}$. We have just shown that $\frac{B_{[k]_n} + B_{[k+1]_n}}{2} \leq B_{[k+1]_n} - 1$, so the result $A_{[k]_n}$ of this top-up sub-step is, at worst, $B_{[k+1]_n}$.

Therefore, we have just shown that:

$$A_{[k]_n} \leq B_{[k+1]_n}$$

Now considering also the case in which $B_{[k]_n} > B_{[k+1]_n}$ (finishing with our WLOG assumption!), we see that $A_{[k]_n}$ is guaranteed to be at most:

$$\max\left\{B_{[k]_n}, B_{[k+1]_n}\right\}$$

Hence (the maximum of a subset will always be $\leq$ to the maximum of the whole set):

$$A_{[k]_n} \leq \max\left\{B_{[1]_n}, B_{[2]_n}, \ldots, B_{[n]_n}\right\}$$

Since k is arbitrary, this implies:

$$A_{[1]_n}, A_{[2]_n}, \ldots, A_{[n]_n} \leq \max\left\{B_{[1]_n}, B_{[2]_n}, \ldots, B_{[n]_n}\right\}$$

And therefore:

$$\max\left\{A_{[1]_n}, A_{[2]_n}, \ldots, A_{[n]_n}\right\} \leq \max\left\{B_{[1]_n}, B_{[2]_n}, \ldots, B_{[n]_n}\right\}$$

as required.

5 Teacher stops giving out candy at some point

The significance of the result that we have just proved (in section 4) is that it implies that the teacher cannot go on giving out candy indefinitely: the teacher must, after finitely many "redistribute then top-up" steps, permanently stop giving out candy.

To show this, we start with the obvious observation that candies are never destroyed; hence the total number (which we'll call T) of candies held by the children never decreases.

But clearly, if we let M be the maximum number of candies held by a child (a function of time), T can never exceed nM. But the result of section 4 says that M never decreases. Hence, T is bounded above by a quantity (namely nM) that is itself bounded above by a constant. (That constant is the value of nM just before the first "redistribute then top-up" step). Therefore T must converge to some limit. Since T is always an integer, this means that T must remain constant after some finite number of "redistribute then top-up" steps. Therefore, the teacher must permanently stop giving out candy after some finite number of steps.

6 The rest is just linear algebra

Obviously, after the teacher finishes giving out candy, the "redistribute then top-up" steps are just "redistribute" steps: If the teacher is no longer giving out candy, then the top-up sub-steps must be identity transformations: all of the children must have even numbers of candies on entry to the top-up sub-steps. Hence, the subsequent "redistribute then top-up" steps must be just linear transformations:

$$\begin{aligned} A_{[1]_n} &= \frac{B_{[1]_n} + B_{[2]_n}}{2} \\ A_{[2]_n} &= \frac{B_{[2]_n} + B_{[3]_n}}{2} \\ A_{[3]_n} &= \frac{B_{[3]_n} + B_{[4]_n}}{2} \\ &\cdots \\ A_{[n]_n} &= \frac{B_{[n]_n} + B_{[n+1]_n}}{2} \end{aligned}$$

Using matrix notation:

$$\begin{pmatrix} A_{[1]_n} \\ A_{[2]_n} \\ A_{[3]_n} \\ \cdots \\ A_{[n]_n} \end{pmatrix} = \begin{pmatrix} 1/2 & 1/2 & 0 & \cdots & 0 \\ 0 & 1/2 & 1/2 & \cdots & 0 \\ 0 & 0 & 1/2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1/2 & 0 & 0 & \cdots & 1/2 \end{pmatrix} \begin{pmatrix} B_{[1]_n} \\ B_{[2]_n} \\ B_{[3]_n} \\ \cdots \\ B_{[n]_n} \end{pmatrix}$$

We'll call the matrix in this equation ♠.

6.1 Numerical experimentation with ♠

To reassure myself that I was on the right track, I tried evaluating some large powers of ♠ (for $n = 5$) with *Octave*[1]:

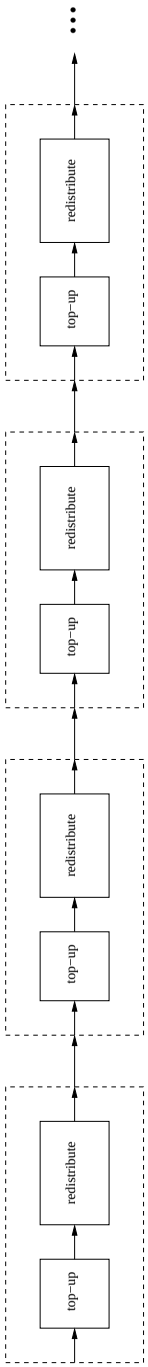

Figure 1: Conceptual view of algorithm as described

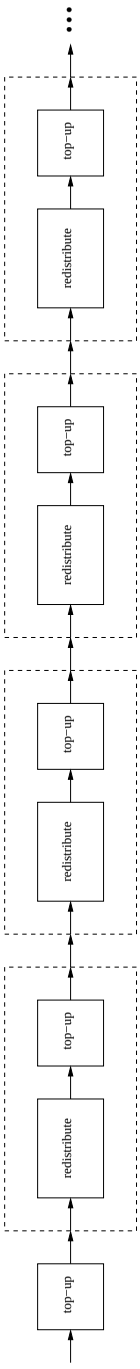

Figure 2: Our conceptual view of algorithm

```
sdaniels@wylie:~$ octave
GNU Octave, version 2.0.16 (i386-pc-linux-gnu).
Copyright (C) 1996, 1997, 1998, 1999, 2000 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1> spadetemp = [1, 1, 0, 0, 0; 0, 1, 1, 0, 0; 0, 0, 1, 1, 0; 0, 0, 0, 1, 1; 1, 0, 0, 0, 1]
spadetemp =

  1  1  0  0  0
  0  1  1  0  0
  0  0  1  1  0
  0  0  0  1  1
  1  0  0  0  1

octave:2> spade5 = spadetemp / 2
spade5 =

  0.50000  0.50000  0.00000  0.00000  0.00000
  0.00000  0.50000  0.50000  0.00000  0.00000
  0.00000  0.00000  0.50000  0.50000  0.00000
  0.00000  0.00000  0.00000  0.50000  0.50000
  0.50000  0.00000  0.00000  0.00000  0.50000

octave:3> spade5^10
ans =

  0.24805  0.21484  0.16113  0.16113  0.21484
  0.21484  0.24805  0.21484  0.16113  0.16113
  0.16113  0.21484  0.24805  0.21484  0.16113
  0.16113  0.16113  0.21484  0.24805  0.21484
  0.21484  0.16113  0.16113  0.21484  0.24805
```

```
octave:4> spade5^20
ans =

  0.20577  0.20178  0.19533  0.19533  0.20178
  0.20178  0.20577  0.20178  0.19533  0.19533
  0.19533  0.20178  0.20577  0.20178  0.19533
  0.19533  0.19533  0.20178  0.20577  0.20178
  0.20178  0.19533  0.19533  0.20178  0.20577

octave:5> spade5^30
ans =

  0.20069  0.20021  0.19944  0.19944  0.20021
  0.20021  0.20069  0.20021  0.19944  0.19944
  0.19944  0.20021  0.20069  0.20021  0.19944
  0.19944  0.19944  0.20021  0.20069  0.20021
  0.20021  0.19944  0.19944  0.20021  0.20069

octave:6> spade5^40
ans =

  0.20008  0.20003  0.19993  0.19993  0.20003
  0.20003  0.20008  0.20003  0.19993  0.19993
  0.19993  0.20003  0.20008  0.20003  0.19993
  0.19993  0.19993  0.20003  0.20008  0.20003
  0.20003  0.19993  0.19993  0.20003  0.20008

octave:7> spade5^50
ans =

  0.20001  0.20000  0.19999  0.19999  0.20000
  0.20000  0.20001  0.20000  0.19999  0.19999
  0.19999  0.20000  0.20001  0.20000  0.19999
```

```
  0.19999  0.19999  0.20000  0.20001  0.20000
  0.20000  0.19999  0.19999  0.20000  0.20001

octave:8> spade5^60
ans =

  0.20000  0.20000  0.20000  0.20000  0.20000
  0.20000  0.20000  0.20000  0.20000  0.20000
  0.20000  0.20000  0.20000  0.20000  0.20000
  0.20000  0.20000  0.20000  0.20000  0.20000
  0.20000  0.20000  0.20000  0.20000  0.20000

octave:9>
```

Hence, it appears as if:

$$\lim_{p\to\infty,\ p\in\mathbb{N}} \spadesuit^p = \begin{pmatrix} 1/n & 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & 1/n & \cdots & 1/n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/n & 1/n & \cdots & 1/n \end{pmatrix}$$

We'll abuse our notation by calling this limit (assuming that it exists, and that the above equation holds) $\spadesuit^\infty$. If we can prove the above equality, then we have solved the problem: the (infinite) sequence of redistribution steps after the teacher stops giving out candy is just a multiplication of the vector of children's candy inventories by $\spadesuit^\infty$. Clearly, if the above equation holds, then the effect of this multiplication by $\spadesuit^\infty$ is to redistribute the children's candy so that all children have equal numbers of candies. Moreover, since the numbers of candies held by the children are always integers, the definition of a limit (i.e. the difference between $\spadesuit^\infty$ and the RHS above gets arbitrarily close to zero) will then imply that the children all have the same number of candies after a *finite* number of steps.

Hence, it remains only to prove the correctness of the above equality.

6.2 Evaluation of $\spadesuit^\infty$

When I was first confronted with the problem of evaluating $\spadesuit^\infty$, my first reaction was to try the standard linear algebra technique of diagonalization by eigenvalues. I tried diagonalizing $\spadesuit$ for $n = 3$ with Octave's built-in *eig* function. I discovered that, indeed, $\spadesuit$ for $n = 3$ can be diagonalized by eigenvalues (my knowledge of linear algebra is a bit rusty; if I remember rightly, not all matrices can be diagonalized by eigenvalues). I then looked carefully at the output, to see if I could guess the general form of the eigenvalues / eigenvectors.

I rapidly realised that the matrix of eigenvectors is just the matrix representation of a (n-point) discrete Fourier transform; I realised that the linear transformation represented by $\spadesuit$ has a simple intuitive interpretation in the frequency domain.

The interpretation is this: Suppose that we want to multiply $\spadesuit$ by a 1 by n input vector X. If we look back at the definition of $\spadesuit$ in section 6, we see that all we have to do is:

1. Let Y be the result of rotating the entries of X *down* by one position.
2. Add X onto Y.
3. Halve the result.

Hence, if we let $\downdownarrows$ denote the operation of rotating the entries of a column vector down by one position, we have the identity:

$$\spadesuit X = \frac{X + \downdownarrows(X)}{2}$$

Now, we apply a Discrete Fourier transform to both sides (noting that the DFT operation is linear), in preparation for application of the DFT shift theorem:

$$\mathrm{DFT}(\spadesuit X) = \frac{\mathrm{DFT}(X) + \mathrm{DFT}(\downdownarrows(X))}{2}$$

Now, the well-known *DFT shift theorem* (*Numerical Recipes*[3] chapter 12) says that $\mathrm{DFT}(\downdownarrows(X))$ equals $H\mathrm{DFT}(X)$, where H is the diagonal matrix defined by:

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & e^{\frac{-2\pi j}{n}} & 0 & \cdots & 0 \\ 0 & 0 & e^{\frac{-4\pi j}{n}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & e^{\frac{-2(n-1)\pi j}{n}} \end{bmatrix}$$

I call the diagonal of H a *phase helix*, although the more usual term is a *linear phase term*. The elements of this diagonal are clearly discrete samples of a helix in three-dimensional

(element number, real component, imaginary component) space. Clearly the length of the diagonal (n samples) corresponds to one complete turn of the helix. Clearly also the helix is of unit radius. Now:

$$\begin{aligned}\mathrm{DFT}(\spadesuit X) &= \frac{\mathrm{DFT}(X) + H\mathrm{DFT}(X)}{2} \\ &= \frac{I\mathrm{DFT}(X) + H\mathrm{DFT}(X)}{2} \\ &= \frac{(I+H)\mathrm{DFT}(X)}{2} \\ &= \frac{(I+H)}{2}\mathrm{DFT}(X)\end{aligned}$$

Then taking the inverse DFT of both sides:

$$\spadesuit X = \mathrm{IDFT}\left(\frac{(I+H)}{2}\mathrm{DFT}(X)\right)$$

So we have now diagonalized $\spadesuit$. Now, if we evaluate the (diagonal) matrix $\frac{(I+H)}{2}$, we see that:

1. The element in the top left-hand corner is $\frac{1+1}{2} = 1$, of which any power will clearly also be 1.
2. Since the entire length of the diagonal corresponds to a single turn of the helix, any other element is of the form $\frac{1+z}{2}$, where z is a complex number of unit magnitude but argument *not equal* to zero. Hence, each such element $\frac{1+z}{2}$ is of magnitude strictly less than one. Therefore, powers of these elements will tend to zero as the exponent tends to infinity.

Hence (abusing our notation again):

$$\left(\frac{I+H}{2}\right)^{\infty} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

(Element [1,1] is 1; all other elements are zero). And, of course,

$$\spadesuit^{\infty} X = \mathrm{IDFT}\left(\left(\frac{I+H}{2}\right)^{\infty}\mathrm{DFT}(X)\right)$$

(where X, as always, is an arbitrary input vector). Now, if we use ω as a shorthand for $e^{\frac{-2\pi j}{n}}$ (following Corman, Leiserson, Rivest and Stein[2]), the matrix representations of the forward and inverse discrete Fourier transforms are:

$$[\mathrm{DFT}] = \frac{1}{\sqrt{n}}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}$$

$$[\mathrm{IDFT}] = \frac{1}{\sqrt{n}}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \overline{\omega} & \overline{\omega^2} & \cdots & \overline{\omega^{n-1}} \\ 1 & \overline{\omega^2} & \overline{\omega^4} & \cdots & \overline{\omega^{2(n-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \overline{\omega^{n-1}} & \overline{\omega^{2(n-1)}} & \cdots & \overline{\omega^{(n-1)(n-1)}} \end{pmatrix}$$

Hence:

$$\left(\frac{I+H}{2}\right)^{\infty}[\text{DFT}] = \begin{pmatrix} 1/\sqrt{n} & 1/\sqrt{n} & 1/\sqrt{n} & \cdots & 1/\sqrt{n} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

So:

$$\begin{aligned} \spadesuit^{\infty} &= [\text{IDFT}]\left(\frac{I+H}{2}\right)^{\infty}[\text{DFT}] \\ &= \frac{1}{\sqrt{n}}\begin{pmatrix} 1 & 1 & \frac{1}{\omega^2} & \cdots & \frac{1}{\omega^{n-1}} \\ 1 & \overline{\omega} & \overline{\omega^2} & \cdots & \overline{\omega^{2(n-1)}} \\ 1 & \overline{\omega^2} & \overline{\omega^4} & \cdots & \overline{\omega^{2(n-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \overline{\omega^{n-1}} & \overline{\omega^{2(n-1)}} & \cdots & \overline{\omega^{(n-1)(n-1)}} \end{pmatrix}\left(\frac{I+H}{2}\right)^{\infty}[\text{DFT}] \\ &= \begin{pmatrix} 1/n & 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & 1/n & \cdots & 1/n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/n & 1/n & \cdots & 1/n \end{pmatrix} \end{aligned}$$

as required. This completes the proof.

7 A remark about multiplication by ♠

It might worry you that the powers of ♠ that we computed above appear to, in general, have non-integer entries. This occurs in spite of the fact that it is obvious, from the problem description, that the numbers of candies held by the children are always integers. It seems a little difficult for the numbers of candies to always remain integers, and also always satisfy a matrix equation involving real numbers.

One thing is obvious, however, from the construction of ♠: if x is a vector whose elements are all equal even integers[1], then $\spadesuit x = x$. (If all children have the same amount of candy, and each child hands half of his / her candy to the child to the left of him / her, then all children will still have the same amount of candy). Hence, one way in which the children's candy inventories can satisfy the matrix inequalities once the teacher stops giving out candy is for all of the children to have the same amount of candy *at this point.*

For all of the inputs on which I have run *candy*, this has indeed happened. For example, if we start with the vector of children's candy inventories:

$$\begin{pmatrix} 97 \\ 4 \\ 68 \\ 79 \\ 88 \\ 91 \end{pmatrix}$$

We get (note that this is a somewhat more complicated example than our earlier example in section 3 above):

```
sdaniels@wylie:~/candy$ ./candy 97 4 68 79 88 91
Initially                 :   97    4   68   79   88   91
After top-up 1            :   98    4   68   80   88   92
After redistribution 1    :   51   36   74   84   90   95
After top-up 2            :   52   36   74   84   90   96
After redistribution 2    :   44   55   79   87   93   74
```

[1] This, of course, is true even if the elements of x are not even integers

```
After top-up 3                :   44   56   80   88   94   74
After redistribution 3        :   50   68   84   91   84   59
After top-up 4                :   50   68   84   92   84   60
After redistribution 4        :   59   76   88   88   72   55
After top-up 5                :   60   76   88   88   72   56
After redistribution 5        :   68   82   88   80   64   58
After top-up 6                :   68   82   88   80   64   58
After redistribution 6        :   75   85   84   72   61   63
After top-up 7                :   76   86   84   72   62   64
After redistribution 7        :   81   85   78   67   63   70
After top-up 8                :   82   86   78   68   64   70
After redistribution 8        :   84   82   73   66   67   76
After top-up 9                :   84   82   74   66   68   76
After redistribution 9        :   83   78   70   67   72   80
After top-up 10               :   84   78   70   68   72   80
After redistribution 10       :   81   74   69   70   76   82
After top-up 11               :   82   74   70   70   76   82
After redistribution 11       :   78   72   70   73   79   82
After top-up 12               :   78   72   70   74   80   82
After redistribution 12       :   75   71   72   77   81   80
After top-up 13               :   76   72   72   78   82   80
After redistribution 13       :   74   72   75   80   81   78
After top-up 14               :   74   72   76   80   82   78
After redistribution 14       :   73   74   78   81   80   76
After top-up 15               :   74   74   78   82   80   76
After redistribution 15       :   74   76   80   81   78   75
After top-up 16               :   74   76   80   82   78   76
After redistribution 16       :   75   78   81   80   77   75
After top-up 17               :   76   78   82   80   78   76
After redistribution 17       :   77   80   81   79   77   76
After top-up 18               :   78   80   82   80   78   76
After redistribution 18       :   79   81   81   79   77   77
After top-up 19               :   80   82   82   80   78   78
After redistribution 19       :   81   82   81   79   78   79
After top-up 20               :   82   82   82   80   78   80
After redistribution 20       :   82   82   81   79   79   81
After top-up 21               :   82   82   82   80   80   82
After redistribution 21       :   82   82   81   80   81   82
After top-up 22               :   82   82   82   80   82   82
After redistribution 22       :   82   82   81   81   82   82
After top-up 23               :   82   82   82   82   82   82
After redistribution 23       :   82   82   82   82   82   82
Candy now evenly distributed.
sdaniels@wylie:~/candy$
```

So we see that the final top-up step, top-up step 23, makes the childrens' candy inventories all equal. The convergence of the algorithm after this is trivial.

However, I do not currently have any proof that the convergence of the algorithm after the teacher stops giving out candy is always trivial in this way. It might be interesting to see if this can be proved, but it is not necessary to prove this in order to solve the problem as stated. I believe that my evaluation of $\spadesuit^{\infty}$ in section 6.2 is straightforward and easy to understand, relying as it does only on basic linear algebra and some elementary properties of the discrete Fourier transform.

8 Acknowledgements

I thank Dr Ulrich Speidel, Dr Michael Dinneen and Assoc-Prof Peter Gibbons, all of the Computer Science Department, University of Auckland, for their assistance with the publication of this technical report.

References

[1] http://www.octave.org

[2] Corman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts and New York, second edition, 2001.

[3] Press, W. H., Teukolsky, S.A., Vettering, W.T., Flannery, B.P., *Numerical Recipes in C - Second Edition*, Cambridge University Press, Cambridge, England 1992.