2020 MAT – Q2 (4 pages; 30/10/21)

(i) $100 = 64 + 32 + 4 = 2^6 + 2^5 + 2^2$

So binary expansion of 100 is 1100100

(ii) [At this point, it is a very good idea to read through to the end of the question, to get some ideas. We note that:

- powers of 2 are bound to be relevant

- some integers can't be expressed as combinations of f & g

- for each integer that works, there will be only one way of obtaining a suitable combination of f & g

- there is a recurrence relation governing the total number of integers that work

To gain an insight into the problem, we could examine how 36 and 67 work:

$gfg(1) = gf[4(1)] = g[2(4)(1) + 1] = 4(2)(4)(1) + 4$

$= 2^5 + 2^2 = 36$

$ffgg(1) = ffg[4(1)] = ff[4(4)(1)] = f[2(4)(4)(1) + 1]$

$= 2(2)(4)(4)(1) + 2(1) + 1 = 2^6 + 2^1 + 2^0 = 67$

We notice that a final f is needed in order to produce the odd number 67, and this is the key to the question.

An alternative way of experimenting is to set up a tree diagram, to consider all combinations of f & g, until 100 is reached. Although time-consuming, this has the advantage of guaranteeing an answer to (ii), and possibly seeing why only certain integers work, and why the recurrence relation works. But unfortunately the tree diagram doesn't help with the last two issues.]

In order to produce 100, the last operation has to be g (f would produce an odd number). This takes us back to 25. The preceding operation then has to be f (g would produce a multiple of 4). This takes us back to 12. The preceding operation then has to be g, to take us back to 3. And the first operation then has to be f (taking us back to 1).

So $100 = gfgf(1)$

(iii) If 200 is to work, the final operation would have to be g (f would produce an odd number)

[just in case the marker hasn't read the sol'n to (ii)]

This takes us back to 50. But now neither $f\ nor\ g$ works as the precding operation, as 50 is neither an odd number nor a multiple of 4.

So 200 is not in S.

(iv) For a given integer n in S, we can establish the final operation, which must be either f, if n is odd, or g, if n is a multiple of 4. This takes us back to the integer m, say (where either $n = 2m + 1$ or $n = 4m$), and the process can be continued until 1 is reached. Thus there will be only one way of getting back to 1 from n, and therefore only one combination of fs & gs that will produce n, starting from 1.

(v) For an element of S that lies in the range $[2^{k+2}, 2^{k+3})$, either the last operation applied was g, in which case this arose from one of the u_k elements of S in the range $[2^k, 2^{k+1})$ (there being a 1-1 match between elements in the two ranges);

or the last operation applied was f.

In this case, consider an element r, where $2^{k+1} \leq r < 2^{k+2}$

Then, applying f to r, we get

$2^{k+2} + 1 \leq 2r + 1 < 2^{k+3} + 1$

And, as $2r + 1$ is odd, it cannot equal 2^{k+3},

and so $2r + 1$ is in the range $[2^{k+2} + 1, 2^{k+3})$.

Also, 2^{k+2} is not odd, and so cannot have f as its last operation.

Thus, there is a 1-1 match between elements in the range $[2^{k+1}, 2^{k+2})$ and elements in the range $[2^{k+2}, 2^{k+3})$ having f as their last operation.

So the u_{k+2} elements in the range $[2^{k+2}, 2^{k+3})$ are accounted for by the u_k elements in the range $[2^k, 2^{k+1})$, together with the u_{k+1} elements in the range $[2^{k+1}, 2^{k+2})$;

ie $u_{k+2} = u_{k+1} + u_k$ for $k \geq 0$

(vi) From (v), $\sum_{r=0}^{k} u_{r+2} = \sum_{r=0}^{k} u_{r+1} + \sum_{r=0}^{k} u_r$ (*)

$\sum_{r=0}^{k} u_{r+2}$ is the number of elements of S in the range $[2^2, 2^{k+3})$,

which equals $s_{k+2} - 2$, as the elements 1 & 3 are missing from $[2^2, 2^{k+3})$

$\sum_{r=0}^{k} u_{r+1}$ is the number of elements of S in the range $[2^1, 2^{k+2})$,

which equals $s_{k+1} - 1$, as the element 1 is missing from $[2^1, 2^{k+2})$

$\sum_{r=0}^{k} u_r$ is the number of elements of S in the range $[2^0, 2^{k+1})$,

which equals s_k

Hence, $s_{k+2} - 2 = (s_{k+1} - 1) + s_k$,

and so $s_{k+2} = s_{k+1} + s_k + 1$, as required.