TD 8: BitFlip

Exercise 2: *Busy beaver.*

Given a BitFlip program p, we define $\text{result}(p) \in \{0,1\}^*$ as follows:

$$\text{result}(p) = \begin{cases} w & \text{if when we run } p \text{ on an empty tape, it eventually stops with the word } w \text{ written on its tape} \\ \varepsilon & \text{if } p \text{ never stops} \end{cases}$$

For example, result(`>*>>`) $= 0100$, and result(`*()`) $= \varepsilon$.

The *Busy Beaver function* $BB : \mathbb{N} \to \mathbb{N}$ is defined as follows. Given an integer $n \in \mathbb{N}$, $BB(n)$ is the length of the longest possible result of a BitFlip program whose code has less than n characters. This is well-defined because there is only a finite number of programs whose code is smaller than n characters.

1. Compute the value of $BB(1)$, $BB(2)$, $BB(3)$.
2. (*) Show that $BB(50) \geq 52$.
   Hint: Write a BitFlip program whose code has less than 50 characters, and whose result is longer than 52 characters.
3. (**) Show that $BB(50) \geq 10000$.
   Hint: Use the `--busybeaver` option to make the program run until the end without displaying each step (but first make sure that it stops!).

We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *computable* when there exists a BitFlip program p such that for every $n \in \mathbb{N}$, when we run p on a tape which contains the word 1^n (and the memory pointer on the first 1), the program stops after a finite number of steps with the word $1^{f(n)}$ written on the tape, and the memory pointer on the first 1.

4. Show that if f and g are computable, then $f \circ g$ is computable.
5. (*) Let f be a computable function. Show that $\exists n_0 \in \mathbb{N}.\ \forall n \geq n_0, BB(n) > f(n)$.
6. Deduce that BB is not computable.

Solutions:

1. The only valid programs whose code is of length 1 are `>`, `<` and `*`. The first two have a result of size 2, and the other has a result of size 1. So, $BB(1) = 2$.

   Similarly, for 2 or 3 characters, there is no way to do anything better than just going three times to the right (or three times to the left). Thus we have $BB(2) = 3$ and $BB(3) = 4$.

2. With 50 characters, we can start using loops to produce larger outputs. The basic idea to achieve this is to find a way to encode a "for" loop, and at each iteration, extend the size of the tape.

   For example, the following 50-character long program writes an output of size 184, after $25,928$ steps of computation (use the `-busybeaver` option to make it compute until the end, but it also helps to watch it run to understand how it works):

```
*<<<<<<<<<<<<<<*<*>(<*(*<*)<<<<<<<<<<<<*>*(*>*)>*)
```

There are of course many other solutions! This particular program shows that $BB(50) \geq 184$: we do not know (yet) whether it is possible to produce an output of size bigger than 184, but we have shown that at least 184 is possible.

3. Actually, we can do much bigger than 184:

   ```
   *<<<<<<<*<*>(<*(*<*)*<*>(<*(*<*)<<<<*>*(*>*)>*)>*)
   ```

   After about 20 minutes of computation (using `-busybeaver` mode), we obtain:

   The program terminated after 28609526508 steps and wrote 117188 symbols on the tape.

   It is probably possible to do much more than that.

4. If f is computed by the bitflip program p, and g is computed by the bitflip program q, then the program qp computes the function $f \circ g$.
   In BitFlip, the concatenation of programs corresponds to composition of functions.

5. Let f be a computable function, and p the BitFlip program that computes f. Let $k = |p| \in \mathbb{N}$ be the size of the program p. We want to show that, given a number $n \in \mathbb{N}$ which is big enough, we can write a program whose code is smaller than n, and which produces a result bigger than $f(n)$.

   The idea would be to write the word 1^n on the tape, then apply the function f to get the word $1^{f(n)}$, and then add one more 1 to get an output of size $f(n)+1$. But if we do it naively, the code of the resulting program would be bigger than n. We need an efficient way to write 1^n on the tape with a program of length smaller than n; this is the same idea as what we did in question 2.

   Claim: the function $n \mapsto 3n$ is computable. Indeed, the following program q of size 20 computes it:

   ```
   (>)<(*>(>)*>*(<)*<)>
   ```

   So, given n, we start with the program $z =$ `*<*<*<...*<*<*` of size $2(n/3)+1$, which writes $1^{n/3}$ on the tape. Then we do the program q of size 20 and obtain 1^n on the tape. Then we do program p of size k and obtain $1^{f(n)}$. Finally, we add one more 1 by doing `<*`.

   The final program is `zqp<*`, which is of size $|z|+|q|+|p|+2 = 2n/3+23+k$. We want this size to be smaller than n: this will be the case as soon as $n \geq 3k+69$.

   Therefore, if we pick $n_0 = 3k+69$, we have proved the result.

   Remark: this means that the Busy Beaver function grows extremely fast asymptotically. It grows faster than exponentials, faster than the Ackermann function, faster than the Goodstein function, faster than Friedman's TREE(n) function, which are all computable functions.

6. If BB was computable, from the previous question we would get $BB(n) > BB(n)$ (for n big enough), which is absurd.