BUSY BEAVER PROBLEM

**The busy beaver problem** is a fun theoretical computer science problem.

The problem is to find the smallest program that outputs as many data as possible and eventually halts.

More formally it goes like this –

given an n-state Turing Machine  with a two symbol alphabet {0, 1}, what is the maximum number of 1s that the machine may print on an initially blank tape (0-filled) before halting?

It turns out that this problem can't be solved. For a small number of states it can be reasoned about, but it can't be solved in general. Theorists call such problems non-computable.

Currently people have managed to solve it for n=1,2,3,4 (for Turing Machines with 1, 2, 3 and 4 states) by reasoning about and running all the possible Turing Machines, but for n ≥ 5 this task has currently been impossible. While most likely it will be solved for n=5, theorists doubt that it shall ever be computed for n=6.

Let's denote the number of 1s that the busy beaver puts on a tape after halting as S(n) and call it **the busy beaver function** (this is the solution to the busy beaver problem).

The busy beaver function is also interesting -- it grows faster than any computable function. It grows like this:

* S(1) = 1
* S(2) = 4
* S(3) = 6
* S(4) = 13
* S(5) ≥ 4098 (the exact result has not yet been found)
* S(6) ≥ 4.6 · 101439 (the exact result shall never be known)

If we were to use one atom for each 1 that the busy beaver puts on the tape, at n=6 we would have filled the whole universe! That's how fast the busy beaver function is growing.

|  |
| --- |
|  |
| Busy-Beaver-Turing-Machine-with-2-states  a0 -> b1r a1 -> b1l |
| b0 -> a1l b1 -> h1r |
|  |
| The initial tape is filled with 0's. The starting state is 'a' and the halting |
| state is 'h'. The notation 'a0 -> b1r' means "if we are in the state 'a' and |
| the current symbol on the tape is '0', then put a '1' in the current cell, |
| switch to state 'b' and move to the right 'r'. This process repeats until the |
| machine ends up in the halting state 'h'. |
|  |
| When run, it produces 4 ones on the tape and halts. |
|  |
| Here are all the tape changes. The tape is infinite and initially blank |
| (filled with 0's). |
|  |
| . starting state |
| | |
| v state change |
| ------------ |
| ...|0|0|0|0|0|0|0|0|0|0|... a0 -> b1r |
| ...|0|0|0|0|1|0|0|0|0|0|... b0 -> a1l |
| ...|0|0|0|0|1|1|0|0|0|0|... a1 -> b1l |
| ...|0|0|0|0|1|1|0|0|0|0|... b0 -> a1l |
| ...|0|0|0|1|1|1|0|0|0|0|... a0 -> b1r |
| ...|0|0|1|1|1|1|0|0|0|0|... b1 -> h1r HALT |
| ...|0|0|1|1|1|1|0|0|0|0|... |
|  |
| The busy beaver stopped after 6 steps and the tape got filled with 4 ones. |

BUSY BEAVER

|  |
| --- |
|  |
| Turing Machine for 1-state Busy Beaver: |
|  |
| a0 -> h1r |
|  |
| The tape gets filled with 1 one and it terminates after 1 step. |
|  |
| Turing Machine for 2-state Busy Beaver: |
|  |
| a0 -> b1r a1 -> b1l |
| b0 -> a1l b1 -> h1r |
|  |
| The tape gets filled with 4 ones and it terminates after 6 steps. |
|  |
| Turing Machine for 3-state Busy Beaver: |
|  |
| a0 -> b1r a1 -> h1r |
| b0 -> c0r b1 -> b1r |
| c0 -> c1l c1 -> a1l |
|  |
| The tape gets filled with 6 ones and it terminates after 14 steps. |
|  |
| Turing Machine for 4-state Busy Beaver: |
|  |
| a0 -> b1r a1 -> b1l |
| b0 -> a1l b1 -> c0l |
| c0 -> h1r c1 -> d1l |
| d0 -> d1r d1 -> a0r |
|  |
| The tape gets filled with 13 ones and it terminates after 107 steps. |
|  |
| Turing Machine for 5-state Busy Beaver: |
|  |
| a0 -> b1l a1 -> a1l |
| b0 -> c1r b1 -> b1r |
| c0 -> a1l c1 -> d1r |
| d0 -> a1l d1 -> e1r |
| e0 -> h1r e1 -> c0r |
|  |
| The tape gets filled with 4098 ones and it terminates after |
| 47176870 steps. |
|  |
| Turing Machine for 6 state Busy Beaver: |
|  |
| a0 -> b1r a1 -> e0l |
| b0 -> c1l b1 -> a0r |
| c0 -> d1l c1 -> c0r |
| d0 -> e1l d1 -> f0l |
| e0 -> a1l e1 -> c1l |
| f0 -> e1l f1 -> h1r |
|  |
| Currently best 6 state Busy Beaver outputs 4.6e1439 ones and |
| terminates after 2.8e2879 steps.  The first four Σ(n) values of 1, 4, 6, 13, for Busy Beaver, are deceptively small but although the exact value of Σ(5) is not yet known, it has been established that it is at least 4098. The best lower bound so far obtained for Σ(6) is 1018267. To hold a string of this number of ones needs *far* more character cells on the simulated tape than there are particles in the universe. This, in turn, means that a full exploration of Σ(6) may need more computational capacity than exists in the known universe and hence the values for Σ(6) -- and S(6), the exact number of state transitions that the optimal BB machine would make in printing out Σ(6) ones -- can never be known.  Much of the problem in working out Σ values even for n = 3 and n = 4 lies in determining whether some of the candidate BB-TMs, that appear to be looping, really *are* in a loop or whether, given long enough, they might actually terminate with an even larger Σ(n) value than the best result known so far. . Radó and Liu ruled out more than 40 such problem cases (`holdouts') in working out that Σ(3) = 6. Allen Brady had to work much harder (in 1983) to eliminate 27 `holdouts' when showing that Σ(4) = 23 and S(4) = 107. |
|  |
|  |

