# Proof of Non-Halting for the 6-State Turing Machine 1RB1LC_1RC---_0LD1RE_0LA1LD_0RC1RF_0LF0RB

**via Run-Length Shift Registers and Binary Parity Countdowns**

**Abstract**

We prove that the 6-state, 2-symbol Turing machine 1RB1LC_1RC---_0LD1RE_0LA1LD_0RC1RF_0LF0RB, started on an initially blank (all-0) tape, never halts. By recording the lengths of the consecutive blocks of 1s on the tape from left to right each time the head reaches the right boundary in state C, the machine's execution reduces to a deterministic rewriting system on a list of positive integers $A = [a_1, a_2, \ldots, a_k]$ paired with a boolean flag indicating whether the previous bounce added a new term to the list. We show that the tail $[a_2, \ldots, a_k]$ consists exclusively of 1s and 2s and acts as a one-element shift register whose 2-positions evolve as an exact binary counter. Using a parity-lock invariant, we prove that the leftmost block length $a_1$ can reach 4 only on steps where the list length $k$ is odd and every odd-indexed entry from $a_3$ upward equals 1. Consequently, the very next bounce is forced to decrement $a_1$ from 4 back to 3, preventing the machine from ever reaching its halting condition.

## 1 Introduction and Overview of the Non-Halting Proof

Consider the 6-state, 2-symbol Turing machine with transition table 1RB1LC_1RC---_0LD1RE_0LA1LD_0RC1RF_0LF0RB, initialized in state A on an all-0 tape:

Table 1: Transition table for 1RB1LC_1RC---_0LD1RE_0LA1LD_0RC1RF_0LF0RB.

| Read Symbol | State A | State B | State C | State D | State E | State F |
|---|---|---|---|---|---|---|
| 0 | 1RB | 1RC | 0LD | 0LA | 0RC | 0LF |
| 1 | 1LC | --- (HALT) | 1RE | 1LD | 1RF | 0RB |

The only halting transition in the table is B1 -> HALT (reading a 1 while in state B). Rather than tracking individual head movements step by step, we observe the tape only at **right bounces**—the moments when the head is in state C at the far right of all 1s on the tape, reading a 0 and about to reverse direction to the left.

At every right bounce, the non-zero portion of the tape consists of blocks (stretches) of consecutive 1s separated from one another by single 0s. Recording the length of each stretch of 1s from left to right produces a sequence of lists:

$$[2] \to [3,1] \to [2,2] \to [2,1,1] \to [3,2,1] \to [2,2,2] \to [2,2,1,1] \to [1,2,2,1] \to \cdots$$

The core intuition of the proof is straightforward:

1. **Only the first number $a_1$ can ever exceed 2:** Every term after the first term $(a_2, a_3, \ldots, a_k)$ is always either 1 or 2. Meanwhile, the leftmost term $a_1$ fluctuates between $1, 2, 3$, and 4, and the machine halts if and only if the leftward sweep reaches the left boundary of the list at a moment when $a_1 \geq 4$.

2. **Why $a_1 = 4$ always resets on the very next bounce:** The leftmost term $a_1$ can only increase when a leftward sweep checks all the even-numbered positions $(a_2, a_4, a_6, \dots)$ and finds that they are all equal to 1. When this happens while $a_1 = 3$, the machine increases $a_1$ to 4 *and simultaneously shifts every element of the list one slot to the right.* Because all the even-numbered positions $(a_2, a_4, a_6, \dots)$ were 1 before the shift, shifting them one slot to the right makes all the **odd-numbered positions** $(a_3, a_5, a_7, \dots)$ equal to 1 after the shift.

3. **The Parity Trap:** We prove that whenever $a_1$ is promoted from 3 to 4, the list has an **odd** number of terms $k$ and did not gain a new term on that step. On the very next step, an odd-length list that did not just gain a term checks the **odd-numbered positions** $(a_k, a_{k-2}, \dots, a_3, a_1)$. Since $a_k, \dots, a_3$ were just filled with 1s by the rightward shift, the sweep skips straight past all of them and hits $a_1 = 4$ directly, which immediately reduces $a_1$ from 4 back down to 3 instead of halting.

## 2 From Tape Bounces to the Run-Length List $A = [a_1, a_2, \dots, a_k]$

To make the proof easy to follow, we define each variable alongside its concrete meaning on the tape.

**Definition 2.1** (Run-Length List and Variables at a Right Bounce)**.** At the $b$-th right bounce ($b = 1, 2, 3, \dots$), the tape contains $k$ stretches of `1`s separated by single `0`s:

$$\dots 000 \underbrace{11\dots1}_{a_1 \text{ ones}} 0 \underbrace{11\dots1}_{a_2 \text{ ones}} 0 \cdots 0 \underbrace{11\dots1}_{a_k \text{ ones}} 000 \dots$$

We describe the state of the machine at this bounce using the following variables:

- $A = [a_1, a_2, \dots, a_k]$ (**the run-length list**): The list of positive integers giving the length of each stretch of `1`s from left to right. Here $a_1$ is the length of the leftmost stretch of `1`s, $a_2$ is the length of the second stretch from the left, and $a_k$ is the length of the rightmost stretch of `1`s.
- $k$ (**the list length**): The total number of stretches of `1`s currently on the tape (i.e., how many numbers are in the list $A$).
- off $\in \{1, 2\}$ (**the right-bounce head offset**): How many cells to the right of the final `1` on the tape the head is located when it bounces in state `C`:
  - off $= 1$ means the head is on the *first* `0` immediately to the right of the last `1`, so the tape ends in $\dots 1^{a_k}$`[C0]`. For $b \geq 2$, this happens **if and only if** the previous step split a 2 at the end of the list into $1, 1$ (so the list just gained a new term).
  - off $= 2$ means the head is on the *second* `0` to the right of the last `1`, so the tape ends in $\dots 1^{a_k}$`0[C0]`. For $b \geq 2$, this happens **if and only if** the previous step did **not** add a new term to the list.

Let us trace the first four bounces from the blank tape to see how $A$, $k$, and off arise:

- **Bounce 1 (step 3):** Starting from `[A0]`, the machine executes `A0` $\to$ `1RB`, `B0` $\to$ `1RC`, reaching `11[C0]`. Here $A = [2]$, the list length is $k = 1$, and the head is 1 cell right of the last `1`, so off $= 1$.
- **Bounce 2 (step 14):** Starting from `11[C0]`, the head sweeps left past the two `1`s, appends two `1`s on the left, and sweeps right to reach `11101[C0]`. Now $A = [3, 1]$, $k = 2$, and off $= 1$ (the list gained a term).
- **Bounce 3 (step 25):** Starting from `11101[C0]`, the head sweeps left, turns around inside the block of three `1`s, and shifts one `1` into the second block to reach `110110[C0]`. Now $A = [2, 2]$, $k = 2$, and off $= 2$ (the list did not gain a term).
- **Bounce 4 (step 33):** Starting from `110110[C0]`, the head turns around inside the rightmost block of two `1`s and splits it into two separate `1`s, reaching `110101[C0]`. Now $A = [2, 1, 1]$, $k = 3$, and off $= 1$ (the list gained a term).

### 2.1 Why the Machine Only Halts at the Left Boundary When $a_1 \geq 4$

As the head sweeps left from a right bounce, it uses states `C`, `D`, and `A` to step across blocks of `1`s. Specifically, suppose state `D` arrives at the single `0` immediately to the right of some block $a_j$ ($1 \leq j \leq k$):

1. Executing `D0 -> 0LA` moves the head one cell left onto the rightmost `1` of block $a_j$ in state `A`.
2. Executing `A1 -> 1LC` leaves that `1` unchanged and moves the head one more cell left—onto the **second-from-right position** of block $a_j$—in state `C`.

Now two possibilities occur depending on the size of $a_j$:

- **If $a_j = 1$:** Because block $a_j$ has only a single `1`, the cell to the left of its rightmost `1` is actually the `0` to the left of block $a_j$. Thus state `C` reads `0` (`C0 -> 0LD`), moving the head in state `D` onto the rightmost `1` of the next block $a_{j-1}$. Since `D1 -> 1LD` moves left across all `1`s without changing them, state `D` skips completely over block $a_{j-1}$ and lands on the `0` to the right of block $a_{j-2}$.
- **If $a_j \geq 2$:** The second-from-right cell of block $a_j$ is a `1`. Thus state `C` reads `1` (`C1 -> 1RE`) and immediately reverses direction to the right!

Notice why off $\in \{1, 2\}$ determines which blocks get tested during this leftward sweep:

- When off $= 1$ (tape ends in $\ldots 1^{a_k}$`[C0]`), the initial step `C0 -> 0LD` places state `D` directly onto the rightmost `1` of $a_k$. State `D` therefore skips $a_k$ completely and lands on the `0` to the right of $a_{k-1}$. Thus the leftward sweep tests $a_{k-1}, a_{k-3}, a_{k-5}, \ldots$ (every second block starting from $k-1$).
- When off $= 2$ (tape ends in $\ldots 1^{a_k}0$`[C0]`), the initial step `C0 -> 0LD` places state `D` onto the `0` immediately to the right of $a_k$. Thus the leftward sweep tests $a_k, a_{k-2}, a_{k-4}, \ldots$ (every second block starting from $k$).

This motivates two convenient index variables:

- $s$ (**the starting test index**): The index of the first block tested by the leftward sweep, given by:

$$s = \begin{cases} k-1 & \text{if off} = 1 \text{ (the previous step added a term)}, \\ k & \text{if off} = 2 \text{ (the previous step did not add a term)}. \end{cases}$$

- $m$ (**the turnaround index**): Moving from right to left through the tested indices $j = s, s-2, s-4, \ldots, m$ is the **first (largest) tested index where** $a_m \geq 2$. If every tested block has length 1 and $s$ is even (meaning the last tested block is $a_2 = 1$), then after testing $a_2 = 1$, state `C` lands on the `0` immediately to the right of $a_1$; we denote this left-boundary hit by $m = 0$.

What happens when $m = 0$, so that state `C` reads the `0` immediately to the right of the leftmost block $a_1$ ($\ldots 000\,1^{a_1}$`[C0]`$1^{a_2}\ldots$)?

1. `C0 -> 0LD` moves onto the rightmost `1` of $a_1$, and `D1 -> 1LD` walks all the way across $a_1$ until it hits the first blank `0` to the left of $a_1$.
2. At that left boundary, `D0 -> 0LA`, `A0 -> 1RB`, and `B0 -> 1RC` write two new `1`s onto the blank tape to the left of $a_1$, enlarging the leftmost block to $a_1 + 2$ ones, and place the head in state `C` on the leftmost `1` of the old block $a_1$:

$$\ldots 0011\texttt{[C1]}\ \underbrace{11\ldots1}_{a_1-1 \text{ ones}}\ 01^{a_2}\ldots$$

3. From here, `C1 -> 1RE` moves one cell right, `E1 -> 1RF` moves a second cell right (if $a_1 \geq 2$), and `F1 -> 0RB` writes a `0` and moves a third cell right in state `B` (if $a_1 \geq 3$).

4. If $a_1 \geq 4$, there is still a fourth `1` of $a_1$ waiting in that cell! State `B` reads that `1` and executes `B1 -> HALT`. Conversely, if $a_1 \leq 3$, state `B` (or `F` or `E`) lands on the `0` separating $a_1$ and $a_2$ instead, and never halts.

**Exact Halting Criterion:** The Turing machine halts on a given bounce if and only if $m = 0$ and $a_1 \geq 4$. Because $m = 0$ happens precisely when the starting test index $s$ is even and $a_2 = a_4 = \cdots = a_s = 1$, this is completely equivalent to:

- off $= 2$ (did not gain a term), $k$ is even, $a_1 \geq 4$, and every even term $a_2, a_4, \ldots, a_k$ is 1; **or**
- off $= 1$ (just gained a term), $k$ is odd, $a_1 \geq 4$, and every even term $a_2, a_4, \ldots, a_{k-1}$ is 1.

# 3 The Three Exact List Transformation Rules (With Worked Examples)

When the leftward sweep does not halt, how does the rightward return sweep change the list $A = [a_1, a_2, \ldots, a_k]$? Suppose the leftward sweep turns around at block $a_m \geq 2$ (for some $1 \leq m \leq k$). At the turnaround moment, state `C` is on the second-from-right `1` of block $a_m$:

$$\ldots 1^{a_m - 2}\,\texttt{[C1]}\,1\,0\,1^{a_{m+1}}\ldots$$

Executing the five transitions `C1 -> 1RE`, `E1 -> 1RF`, `F0 -> 0LF`, `F1 -> 0RB`, and `B0 -> 1RC` transforms this local region into:

$$\ldots 1^{a_m - 1}\,0\,1\,\texttt{[C1]}\,1^{a_{m+1}-1}\ldots$$

Look at what those five steps just accomplished:

- Block $a_m$ lost its rightmost `1`, so its length decreased by 1 ($a_m \mapsto a_m - 1$).
- The `0` to the right of old $a_m$ became a `1`.

What happens next depends on whether $m = k$ (the turnaround happened at the very last block) or $m < k$ (there are more blocks to the right of $a_m$).

## 3.1 Rule 1: Splitting the Final Block ($m = k$)

If off $= 2$ and the final block has $a_k = 2$, then $s = k$ and the leftward sweep immediately turns around at $m = k$. Since there is no block $a_{k+1}$ to the right of $a_k$, decreasing $a_k = 2$ by 1 and turning the `0` to its right into a `1` simply splits the final 2 into $1, 1$, with the head stopping in state `C` on the very first `0` to the right of the new 1:

$$[a_1, \ldots, a_{k-1}, 2] \quad (\text{off} = 2) \quad \longrightarrow \quad [a_1, \ldots, a_{k-1}, 1, 1] \quad (\text{off} = 1,\ \text{length becomes } k+1).$$

**Example 3.1** (Worked Examples of Rule 1)**.**

- **Bounce 3 → Bounce 4:** At Bounce 3, $A = [2, 2]$ with off $= 2$ ($k = 2$). The first tested index is $s = 2$, where $a_2 = 2 \geq 2$, so $m = 2 = k$. The final 2 splits into $1, 1$, producing $A = [2, 1, 1]$ with off $= 1$ ($k = 3$).
- **Bounce 12 → Bounce 13:** At Bounce 12, $A = [2, 2, 1, 1, 2]$ with off $= 2$ ($k = 5$). Here $s = 5$ and $a_5 = 2$, so $m = 5 = k$. The final 2 splits into $1, 1$, producing $A = [2, 2, 1, 1, 1, 1]$ with off $= 1$ ($k = 6$).

### 3.2 Rule 2: Interior Turnaround and Rightward Shift ($1 \le m < k$)

When $1 \le m < k$, the `1` written to the right of $a_m$ merges onto the left end of the next block $a_{m+1}$, acting as an incoming **carry** $c = 1$ into block $a_{m+1}$. As the head continues moving right across each remaining block $a_j$ ($j = m+1, m+2, \ldots, k$), remember that every block $a_j$ for $j \ge 2$ has length $a_j \in \{1, 2\}$. Checking the four combinations of incoming carry $c \in \{0, 1\}$ and block size $a_j \in \{1, 2\}$ reveals a remarkable simplification:

- If $c = 1$ and $a_j = 1$: the block gains the carried `1` on its left and becomes $a'_j = 2$, while no `1` is pushed out its right end (outgoing carry $c' = 0$).
- If $c = 1$ and $a_j = 2$: the block gains a `1` on its left and pushes its rightmost `1` across the next `0`, so it stays $a'_j = 2$ and sends outgoing carry $c' = 1$.
- If $c = 0$ and $a_j = 1$: the head steps cleanly over the single `1` via `C1 -> 1RE -> E0 -> 0RC`, leaving $a'_j = 1$ and outgoing carry $c' = 0$.
- If $c = 0$ and $a_j = 2$: the head enters the leftmost `1` of 11, which is also its second-from-right `1`! Thus it pushes the rightmost `1` across the next `0`, reducing the block to $a'_j = 1$ and sending outgoing carry $c' = 1$.

In all four cases, the new block length $a'_j$ and the new outgoing carry $c'$ satisfy:

$$a'_j = 1 + c, \qquad c' = a_j - 1.$$

Look at what this means for the next block $j+1$: the new value of block $j+1$ will be $a'_{j+1} = 1 + c' = 1 + (a_j - 1) = a_j$. **Every block from $m+1$ to $k-1$ simply shifts its value one position to the right!** And because $m < k$ guarantees that the final block before the bounce was $a_k = 1$ (if off $= 1$, $a_k = 1$ was just created; if off $= 2$, $a_k$ was tested first and had to be 1 since $m < k$), the old $a_k = 1$ simply drops off the right end of the list with outgoing carry $c_{\text{out}} = a_k - 1 = 0$, leaving off $= 2$.

**Rule 2 Summary (Interior Turnaround $1 \le m < k$):** Decrement $a_m$ by 1, insert a 2 immediately to its right (at position $m+1$), shift the sublist $[a_{m+1}, \ldots, a_{k-1}]$ one slot to the right (into positions $m+2, \ldots, k$), drop the old final term $a_k = 1$, and set off $= 2$:

$$[a_1, \ldots, a_{m-1},\ \mathbf{a_m},\ \underline{a_{m+1}, \ldots, a_{k-1}}, 1] \longrightarrow [a_1, \ldots, a_{m-1},\ \mathbf{a_m - 1},\ \mathbf{2},\ \underline{a_{m+1}, \ldots, a_{k-1}}] \quad (\text{off} = 2).$$

**Example 3.2** (Worked Examples of Rule 2)**.**

- **Bounce 11 → Bounce 12 ($m = 1$):** At Bounce 11, $A = [3, 1, 1, 2, 1]$ with off $= 2$ ($k = 5$, so $s = 5$). The leftward sweep tests the odd positions: $a_5 = 1$, $a_3 = 1$, and stops at $a_1 = 3 \ge 2$ ($m = 1$). Applying Rule 2 at $m = 1$:
  1. $a_1 = 3$ decreases by 1 to become 2;
  2. a new 2 is inserted at position 2;
  3. the old elements $[a_2, a_3, a_4] = [1, 1, 2]$ shift one slot right into positions $3, 4, 5$, while the old $a_5 = 1$ drops off the end.

  This produces $A = [2, 2, 1, 1, 2]$ with off $= 2$.
- **Bounce 26 → Bounce 27 → Bounce 28 ($m = 6$, then $m = 4$):**

– At Bounce 26, $A = [3, 1, 1, 2, 1, 2, 1, 1]$ with off $= 2$ ($k = 8$, so $s = 8$). Testing the even positions from the right: $a_8 = 1$, then $a_6 = 2 \geq 2$, so $m = 6$. Decrementing $a_6$ to 1, inserting 2 at position 7, and shifting $[a_7] = [1]$ into position 8 yields Bounce 27:

$$A = [3, 1, 1, 2, 1, \mathbf{1}, \mathbf{2}, 1] \quad (\text{off} = 2).$$

– At Bounce 27, off $= 2$ and $k = 8$, so the sweep again tests the even positions: $a_8 = 1$, *skips* $a_7 = 2$ (since 7 is odd), sees $a_6 = 1$, skips $a_5 = 1$, and stops at $a_4 = 2$ ($m = 4$). Applying Rule 2 at $m = 4$ decrements $a_4$ to 1, inserts 2 at position 5, and shifts the old $[a_5, a_6, a_7] = [1, 1, 2]$ one slot right into positions $6, 7, 8$:

$$A = [3, 1, 1, \mathbf{1}, \mathbf{2}, \underline{1, 1, 2}] \quad (\text{off} = 2).$$

### 3.3 Rule 3: Left-Boundary Bounce ($m = 0$)

When the starting test index $s$ is even and every even-positioned term is 1 ($a_2 = a_4 = \cdots = a_s = 1$), the leftward sweep reaches the left boundary ($m = 0$). For $k \geq 2$, if $a_1 \leq 3$, the left-end bounce updates $a_1$, injects a carry $c = 0$ (if $a_1 = 1$) or $c = 1$ (if $a_1 \in \{2, 3\}$) into position 2, and runs the exact same rightward shift register across positions $2, \ldots, k$:

- **If $a_1 = 1$:** $a_1$ becomes 3, a 1 is inserted at position 2, $[a_2, \ldots, a_{k-1}]$ shifts one slot right into positions $3, \ldots, k$, the old $a_k = 1$ drops off, and off $= 2$.
- **If $a_1 = 2$:** $a_1$ becomes 3, a 2 is inserted at position 2, $[a_2, \ldots, a_{k-1}]$ shifts one slot right into positions $3, \ldots, k$, the old $a_k = 1$ drops off, and off $= 2$.
- **If $a_1 = 3$:** $a_1$ becomes 4, a 2 is inserted at position 2, $[a_2, \ldots, a_{k-1}]$ shifts one slot right into positions $3, \ldots, k$, the old $a_k = 1$ drops off, and off $= 2$.
- **If $a_1 \geq 4$:** The machine **halts**.

(At the very first bounce $b = 1$, where $A = [2]$ has length $k = 1$ and off $= 1$, $m = 0$ turns $[2]$ into $[3, 1]$ with off $= 1$.)

**Example 3.3** (Worked Examples of Rule 3)**.**

- **Bounce 10 → Bounce 11 ($m = 0$ with $a_1 = 1$):** At Bounce 10, $A = [1, \underline{1, 2, 1}, 1]$ with off $= 1$ ($k = 5$, so $s = 4$). The sweep checks the even positions $a_4 = 1$ and $a_2 = 1$, reaching $m = 0$. Since $a_1 = 1$, $a_1$ becomes 3, a 1 is inserted at position 2, and the old $[a_2, a_3, a_4] = [1, 2, 1]$ shifts one slot right into positions $3, 4, 5$, giving:

$$A = [\mathbf{3}, \mathbf{1}, \underline{1, 2, 1}] \quad (\text{off} = 2).$$

- **Bounce 29 → Bounce 30 ($m = 0$ with $a_1 = 3$):** At Bounce 29, $A = [3, \underline{1, 1, 1, 2, 1, 1, 1}, 1]$ with off $= 1$ ($k = 9$, so $s = 8$). Every even position $a_8, a_6, a_4, a_2$ is 1, so $m = 0$. Since $a_1 = 3$, $a_1$ becomes 4, a 2 is inserted at position 2, and the old $[a_2, \ldots, a_8] = [1, 1, 1, 2, 1, 1, 1]$ shifts one slot right into positions $3, \ldots, 9$, giving:

$$A = [\mathbf{4}, \mathbf{2}, \underline{1, 1, 1, 2, 1, 1, 1}] \quad (\text{off} = 2).$$

# 4 Tracking the Positions of the 2s and Why They Form a Binary Countdown

Look closely at Rules 1, 2, and 3: after the first term $a_1$, the only numbers ever inserted into positions $2, 3, \ldots, k$ are 1 and 2, and existing numbers in positions $2, \ldots, k$ are only ever shifted right or decremented from 2 to 1. Therefore, **every term $a_j$ for $j \geq 2$ is always equal to 1 or 2.**

Because every term in the tail $[a_2, a_3, \ldots, a_k]$ is either 1 or 2, writing out long strings of 1s is unnecessary. We can represent the tail completely by recording the list length $k$ and the 1-based positions where the 2s are located!

**Definition 4.1** (The Position List $P$ of the 2s in the Tail)**.** Given a run-length list $A = [a_1, a_2, \ldots, a_k]$, we define:

- $P = [p_1, p_2, \ldots, p_r]$ (**the 2-position list**): The strictly increasing list of 1-based indices $2 \leq p_1 < p_2 < \cdots < p_r \leq k$ where $a_{p_i} = 2$.
- $r$ (**the number of 2s in the tail**): The number of elements in $P$ (i.e., how many 2s appear in $[a_2, \ldots, a_k]$).

**Example 4.2** (How to Translate Between $A$ and $P$)**.** Consider Bounce 35, where $k = 10$ and

$$A = [\underbrace{2}_{a_1}, \underbrace{\mathbf{2}}_{a_2}, \underbrace{\mathbf{2}}_{a_3}, \underbrace{1}_{a_4}, \underbrace{\mathbf{2}}_{a_5}, \underbrace{1}_{a_6}, \underbrace{1}_{a_7}, \underbrace{1}_{a_8}, \underbrace{1}_{a_9}, \underbrace{1}_{a_{10}}].$$

Looking at positions 2 through 10, the 2s sit at positions $2, 3$, and $5$. Therefore:

$$a_1 = 2, \qquad k = 10, \qquad P = [2, 3, 5] \quad (r = 3).$$

Now let us see what happens to the position list $P = [p_1, p_2, \ldots, p_r]$ when Rule 2 ($2 \leq m < k$) is applied at $m = p_i$:

1. The 2 at position $p_i$ is decremented to 1, and a new 2 is inserted immediately to its right at position $p_i + 1$. So $p_i$ moves one slot to the right: $p_i \mapsto p_i + 1$.
2. Every element to the right of $p_i$ is shifted one slot to the right, so every 2 that was at a position $p_j > p_i$ *also* moves one slot to the right: $p_j \mapsto p_j + 1$ for all $j > i$.
3. Every element to the left of $p_i$ is untouched, so $p_j$ stays unchanged for all $j < i$.

**How $P$ Changes on an Interior Step ($m = p_i < k$):** Find the rightmost position $p_i \in P$ whose parity (even or odd) matches the starting test index $s$. Then **add $+1$ to $p_i$ and to every $p_j$ to the right of $p_i$:**

$$[p_1, \ldots, p_{i-1}, \ \mathbf{p_i}, \mathbf{p_{i+1}}, \ldots, \mathbf{p_r}] \longrightarrow [p_1, \ldots, p_{i-1}, \ \mathbf{p_i + 1}, \mathbf{p_{i+1} + 1}, \ldots, \mathbf{p_r + 1}].$$

Why is this rule so powerful? Because adding $+1$ to an integer **flips its parity** (even becomes odd, and odd becomes even)! Watch what happens to the parities of $[p_1, \ldots, p_r]$:

- Suppose $s$ is **even**, so the leftward sweep looks for the rightmost **even** position in $P$. To track which positions in $P$ are even, assign each $p_i$ an **even-parity bit** $\bar{b}_i$:

$$\bar{b}_i = \begin{cases} 1 & \text{if } p_i \text{ is even,} \\ 0 & \text{if } p_i \text{ is odd.} \end{cases}$$

Since the sweep picks the rightmost even $p_i$, it picks the rightmost bit with $\bar{b}_i = 1$ (so all bits to its right have $\bar{b}_{i+1} = \cdots = \bar{b}_r = 0$, meaning $p_{i+1}, \ldots, p_r$ are odd). Adding $+1$ to $p_i, p_{i+1}, \ldots, p_r$ makes $p_i$ odd ($\bar{b}_i : 1 \to 0$) and makes $p_{i+1}, \ldots, p_r$ even ($\bar{b}_{i+1}, \ldots, \bar{b}_r : 0 \to 1$):

$$(\ldots, 1, 0, 0, \ldots, 0)_2 \longrightarrow (\ldots, 0, 1, 1, \ldots, 1)_2.$$

In binary arithmetic, changing $(\ldots 100 \ldots 0)_2$ to $(\ldots 011 \ldots 1)_2$ is literally **subtracting** 1 **in binary**!

- Similarly, if $s$ is **odd**, the sweep looks for the rightmost **odd** position in $P$. Defining the **odd-parity bit** $b_i = p_i \bmod 2$ (1 if $p_i$ is odd, 0 if $p_i$ is even), each bounce picks the rightmost $b_i = 1$ and flips $b_i : 1 \to 0$ and $b_{i+1}, \ldots, b_r : 0 \to 1$, which again **subtracts** 1 **in binary**!

**Example 4.3** (Worked Example of the Binary Countdown (Bounces 35 to 39))**.** At Bounce 35, $k = 10$ (even) and off $= 2$, so $s = 10$ (even): the machine looks for the rightmost **even** position in $P$. Table 2 shows how the full list $A$, the 2-positions $P$, and the 3-bit binary counter $\bar{b} = (\bar{b}_1\bar{b}_2\bar{b}_3)_2$ evolve step by step from Bounce 35 to Bounce 39.

Table 2: Step-by-step evolution from Bounce 35 to Bounce 39 ($k = 10, \text{off} = 2$), showing how $P$ counts down in binary from $4 = (100)_2$ to $0 = (000)_2$.

| **Bounce** | **Full Run-Length List** $A$ | **2-Positions** $P$ | **Rightmost Even** $p_i$ | **Even Bits** $(\bar{b}_1\bar{b}_2\bar{b}_3)_2$ | **Base-10 Value** |
|---|---|---|---|---|---|
| 35 | $[2, \mathbf{2}, \mathbf{2}, 1, \mathbf{2}, 1, 1, 1, 1, 1]$ | $[\mathbf{2}, 3, 5]$ | $m = p_1 = 2$ (increment $p_1, p_2, p_3$) | $(100)_2$ | 4 |
| 36 | $[2, 1, \mathbf{2}, \mathbf{2}, 1, \mathbf{2}, 1, 1, 1, 1]$ | $[3, 4, \mathbf{6}]$ | $m = p_3 = 6$ (increment $p_3$ only) | $(011)_2$ | 3 |
| 37 | $[2, 1, \mathbf{2}, \mathbf{2}, 1, 1, \mathbf{2}, 1, 1, 1]$ | $[3, \mathbf{4}, 7]$ | $m = p_2 = 4$ (increment $p_2, p_3$) | $(010)_2$ | 2 |
| 38 | $[2, 1, \mathbf{2}, 1, \mathbf{2}, 1, 1, \mathbf{2}, 1, 1]$ | $[3, 5, \mathbf{8}]$ | $m = p_3 = 8$ (increment $p_3$ only) | $(001)_2$ | 1 |
| 39 | $[2, 1, \mathbf{2}, 1, \mathbf{2}, 1, 1, 1, \mathbf{2}, 1]$ | $[3, 5, 9]$ | None! All $p_i$ are odd ($m = 0$) | $(000)_2$ | 0 |

Notice two key features in Table 2:

1. At every single step of the countdown, the rightmost 2 ($p_3$) moves right by $+1$ ($5 \to 6 \to 7 \to 8 \to 9$), because $p_3$ is always at or to the right of whichever $p_i$ is chosen.
2. When the binary counter reaches $(000)_2 = 0$ at Bounce 39, all 2s sit at odd positions ($P = [3, 5, 9]$), which means every even position in $A$ is 1, triggering the left-boundary bounce $m = 0$!

We summarize this binary countdown property in the following lemma.

**Lemma 4.4** (The Binary Countdown Lemma)**.** *While the list length $k$ and the sweep parity $s \bmod 2$ remain constant and the rightmost 2-position satisfies $p_r < k$:*

1. *If $s$ is **even**, each bounce decrements the $r$-bit binary integer $B_{even}(P) = (\bar{b}_1\bar{b}_2 \ldots \bar{b}_r)_2$ (where $\bar{b}_i = 1$ if $p_i$ is even and 0 if $p_i$ is odd) by 1, and increases the rightmost position $p_r$ by $+1$.*
2. *If $s$ is **odd**, each bounce decrements the $r$-bit binary integer $B_{odd}(P) = (b_1 b_2 \ldots b_r)_2$ (where $b_i = 1$ if $p_i$ is odd and 0 if $p_i$ is even) by 1, and increases the rightmost position $p_r$ by $+1$.*
3. *Moreover, for any $r$-tuple $P$, the bitwise complement identity $\bar{b}_i + b_i = 1$ implies that at all times:*

$$B_{even}(P) + B_{odd}(P) = (11 \ldots 1)_2 = 2^r - 1.$$

*Thus, each even-parity step that decrements $B_{even}(P)$ by 1 simultaneously **increments** $B_{odd}(P)$ by 1.*

# 5 Proof That the Halting Condition Is Never Reached

We now have everything needed to prove that the machine never halts. Recall how the first term $a_1$ changes across bounces:

- When $m \geq 2$, $a_1$ does not change.
- When $m = 1$, $a_1$ decreases by 1 ($a_1 \mapsto a_1 - 1$).
- When $m = 0$, $a_1 \in \{1, 2\}$ becomes 3, $a_1 = 3$ becomes 4, and $a_1 \geq 4$ halts.

Therefore, to ever halt, the machine must first hit an $m = 0$ bounce while $a_1 = 3$ (which raises $a_1$ to 4) and then reach another $m = 0$ bounce **before** hitting any $m = 1$ bounce (since any $m = 1$ bounce immediately reduces $a_1$ from 4 back to 3).

Let us first see what happens immediately after *any* $m = 0$ bounce:

- Before an $m = 0$ bounce, every even position in $A$ is 1, so all 2s in the tail sit at **odd** positions ($P \subseteq \{3, 5, 7, \dots\}$).
- During the $m = 0$ bounce (Rule 3), every existing element of the tail shifts one slot to the right, moving all those 2s from odd positions to **even** positions! In addition, if $a_1 \in \{2, 3\}$, a new 2 is inserted at position 2, which is **also even**.
- Therefore, on the step immediately following *any* $m = 0$ bounce, off $= 2$ and **every 2 in the tail sits at an even position** (which means every odd position $a_3, a_5, a_7, \dots$ is equal to 1).

This leads directly to our key parity lemma.

**Lemma 5.1** (Immediate Reset After Any Odd-Length $m = 0$ Bounce)**.** *Suppose an $m = 0$ bounce occurs at step $b$ when the list length $k$ is **odd** (so* off $= 1$*). Then on the very next bounce $b + 1$, the machine is **guaranteed** to hit $m = 1$, which immediately decrements $a_1$ by 1.*

*Proof.* At bounce $b$, the list length $k$ is odd and $m = 0$. Rule 3 updates $a_1$ to $a_1' \in \{3, 4\}$, shifts all 2s in the tail to even positions, leaves the list length unchanged at $k$ (still odd), and sets off $= 2$.

Now consider bounce $b + 1$:

1. Because off $= 2$ and the list length $k$ is odd, the starting test index at bounce $b + 1$ is $s = k$, which is **odd**. Thus the leftward sweep at bounce $b + 1$ tests the odd indices $k, k - 2, \dots, 5, 3, 1$.
2. However, all 2s in the tail at bounce $b + 1$ sit at **even** indices! Every odd index from 3 to $k$ is equal to 1:
$$a_3' = a_5' = \cdots = a_k' = 1.$$
3. Consequently, the leftward sweep at bounce $b + 1$ skips past $a_k', a_{k-2}', \dots, a_3'$ and is forced to stop at $m = 1$ (since $a_1' \geq 3 \geq 2$). Applying Rule 2 at $m = 1$ immediately decrements $a_1'$ by 1.

□

**Example 5.2** (Concrete Illustration of Lemma 5.1 at Bounce 29 → 30 → 31)**.** At Bounce 29, the list has odd length $k = 9$ and off $= 1$:

$$\text{Bounce 29:} \quad A = [3, 1, 1, 1, \mathbf{2}, 1, 1, 1, 1] \quad (k = 9 \text{ odd}, \ \text{off} = 1, \ P = [5]).$$

Here $s = 9 - 1 = 8$ (even), and there are no even numbers in $P = [5]$, so Bounce 29 hits $m = 0$ with $a_1 = 3$. Rule 3 raises $a_1$ from 3 to 4, inserts a 2 at index 2, and shifts the 2 at index 5 to index 6, yielding Bounce 30:

$$\text{Bounce 30:} \quad A = [\mathbf{4}, \mathbf{2}, 1, 1, 1, \mathbf{2}, 1, 1, 1] \quad (k = 9 \text{ odd}, \text{ off} = 2, \ P = [2, 6]).$$

Now at Bounce 30, $k = 9$ is odd and off $= 2$, so $s = 9$ (odd): the sweep tests the odd positions $a_9, a_7, a_5, a_3, a_1$. Because $P = [2, 6]$ contains only even positions, $a_9 = a_7 = a_5 = a_3 = 1$. Thus Bounce 30 immediately hits $m = 1$ and decrements $a_1$ from 4 back down to 3 at Bounce 31:

$$\text{Bounce 31:} \quad A = [\mathbf{3}, \mathbf{2}, \mathbf{2}, 1, 1, 1, \mathbf{2}, 1, 1] \quad (k = 9 \text{ odd}, \text{ off} = 2, \ P = [2, 3, 7]).$$

Lemma 5.1 tells us that whenever $m = 0$ happens at an **odd** list length $k$, the very next bounce is always $m = 1$, which immediately reduces $a_1$. Therefore, the *only* way an $m = 0$ bounce can be followed by another $m = 0$ bounce without an intervening $m = 1$ bounce is if the first $m = 0$ bounce occurs at an **even** list length $k$.

Our next lemma analyzes what happens after any even-length $m = 0$ bounce and shows that even when it leads to a second $m = 0$ bounce, that second $m = 0$ bounce **must occur at the odd length** $k + 1$, where Lemma 5.1 immediately forces $m = 1$ on the following step!

**Lemma 5.3** (Evolution After Any Even-Length $m = 0$ Bounce)**.** *Suppose an $m = 0$ bounce occurs at step $b$ when the list length $k$ is **even** (so off $= 2$) and $a_1 \in \{1, 2\}$. At step $b + 1$, $a_1 = 3$, off $= 2$, and all 2s in the tail sit at even positions:*

$$Q = [q_1 < q_2 < \cdots < q_s] \subseteq \{2, 4, \ldots, k\} \quad (\textit{all } q_i \textit{ even}).$$

*Let $D = k - q_s$ be the distance from the rightmost 2 (at $q_s$) to the end of the list $k$, and assume $D \leq 2^s - 2$. Then:*

1. ***Case 1** ($D < 2^s - 2$)**:** The next bounce with $m \leq 1$ is an $m = 1$ **bounce** at odd list length $k + 1$, which decrements $a_1$ from 3 back to 2.*
2. ***Case 2** ($D = 2^s - 2$, **tight equality**)**:** The next bounce with $m \leq 1$ is an $m = 0$ **bounce at odd list length** $k + 1$ (with off $= 1$ and $a_1 = 3$). That bounce raises $a_1$ from 3 to 4 at odd length $k + 1$, after which Lemma 5.1 guarantees that the **very next bounce hits** $m = 1$ (reducing $a_1 : 4 \to 3$), followed by a second $m = 1$ bounce (reducing $a_1 : 3 \to 2$) before any further $m = 0$ bounce can occur.*

*Proof.* At step $b + 1$, $k$ is even, off $= 2$ (so $s = k$ is even), and all $s$ positions in $Q = [q_1, \ldots, q_s]$ are even, so $B_{\text{even}}(Q) = (11 \ldots 1)_2 = 2^s - 1$.

By Lemma 4.4(1), each step decrements $B_{\text{even}}(Q)$ by 1 and moves the rightmost position $q_s$ right by $+1$. Since both $k$ and $q_s$ are even, $D = k - q_s$ is an even integer. After exactly $D$ bounces:

- The rightmost element $q_s$ reaches $k$. Because $D$ is even, $q_s = k$ is still even.
- Because $q_s$ is the least-significant bit of the $s$-bit counter $Q$, the prefix of the first $s - 1$ elements $Q_{\text{prefix}} = [q_1, \ldots, q_{s-1}]$ only stepped once every 2 bounces—that is, $D/2$ times. By Lemma 4.4(3), those $D/2$ even-parity steps increased $B_{\text{odd}}(Q_{\text{prefix}})$ from 0 to $D/2$, and moved $q_{s-1}$ to $q_{s-1} + D/2$.

On the next bounce (step $b + 1 + D$), the sweep sees $q_s = k$ (even) and applies Rule 1 ($m = k$): the 2 at position $k$ splits into $1, 1$, removing $q_s$ from the 2-position list, **increasing the list length from $k$ (even) to $k + 1$ (odd)**, and setting off $= 1$.

At the following bounce $b^* = b + 2 + D$, the list has **odd length** $k + 1$ and off $= 1$, so the starting test index is $s^* = (k + 1) - 1 = k$, which is **still even**. There are now two cases for the remaining $s - 1$ elements $Q_{\text{prefix}}$:

- **If $D = 2^s - 2$ (Case 2):** Then $B_{\text{odd}}(Q_{\text{prefix}}) = D/2 = 2^{s-1} - 1 = (11\ldots1)_2$, meaning **all $s-1$ remaining elements of $Q_{\text{prefix}}$ are odd**! Because $s^* = k$ is even and no even elements remain in $Q_{\text{prefix}}$, bounce $b^*$ hits $m = 0$ **at odd list length** $k+1$ while $a_1 = 3$. This raises $a_1$ to 4 at bounce $b^*+1$, with odd list length $k+1$ and off $= 2$. By Lemma 5.1, bounce $b^*+1$ immediately hits $m = 1$, reducing $a_1$ from 4 to 3 and setting the new $(s+1)$-element position list $P' = [2, 3, q'_1 + 2, \ldots, q'_{s-1} + 2]$ to odd-parity binary state $(0, 1, 1, \ldots, 1)_2 = 2^s - 1$, which counts down to $(0, 0, \ldots, 0)_2$ (all even) and triggers a second $m = 1$ bounce reducing $a_1$ from 3 to 2.

- **If $D < 2^s - 2$ (Case 1):** Then $B_{\text{odd}}(Q_{\text{prefix}}) = D/2 < 2^{s-1} - 1$, so at least one element of $Q_{\text{prefix}}$ is still even at bounce $b^*$. Bounce $b^*$ therefore applies Rule 2 at the rightmost even element of $Q_{\text{prefix}}$, which takes 1 more even-parity step on $Q_{\text{prefix}}$ (bringing $B_{\text{odd}}(Q_{\text{prefix}})$ to $M = D/2 + 1 \le 2^{s-1} - 1$ and moving $q_{s-1}$ to $q_{s-1} + D/2 + 1$) and sets off $= 2$ at **odd length** $k+1$.

  From bounce $b^*+1$ onward, the list length is $k+1$ (odd) and off $= 2$, so the starting test index is $k+1$ (**odd**)! By Lemma 4.4(2), the system now runs the **odd-parity countdown** on $Q_{\text{prefix}}$, decrementing $B_{\text{odd}}(Q_{\text{prefix}})$ from $M = D/2 + 1$ down to 0 in $M$ steps. During those $M$ steps, $q_{s-1}$ moves right by $+1$ per step, reaching a final position of:

$$q''_{s-1} = q_{s-1} + \left(\frac{D}{2} + 1\right) + \left(\frac{D}{2} + 1\right) = q_{s-1} + D + 2 = q_{s-1} + (k - q_s) + 2 = k - (q_s - q_{s-1} - 2).$$

  Because $q_{s-1} < q_s$ were two distinct **even** integers at step $b+1$, their difference is at least 2 ($q_s - q_{s-1} \ge 2$). Therefore:

$$q''_{s-1} = k - (q_s - q_{s-1} - 2) \le k < k + 1.$$

  Thus $q_{s-1}$ **never reaches the right end** $k+1$ before the counter $B_{\text{odd}}(Q_{\text{prefix}})$ reaches 0. Once $B_{\text{odd}}(Q_{\text{prefix}}) = 0$ (all elements of $Q_{\text{prefix}}$ are even) at odd length $k+1$, the very next bounce has no odd elements in $Q_{\text{prefix}}$ and is forced to hit $m = 1$, decrementing $a_1$ from 3 back to 2.

□

**Example 5.4** (Worked Example of Case 2 ($D = 2^s - 2$) at Bounces 25 to 34)**.** Let us watch Case 2 of Lemma 5.3 happen in real time between Bounce 25 and Bounce 34:

1. **Bounce 25 ($m = 0$ at even length $k = 8$):** $A = [1, 1, \mathbf{2}, 1, \mathbf{2}, 1, 1, 1]$ with off $= 2$ and $P = [3, 5]$. Since $s = 8$ (even) and all 2s are at odd positions, Bounce 25 hits $m = 0$ with $a_1 = 1$.

2. **Bounce 26 (Step $b+1$):** Rule 3 sets $a_1 = 3$ and shifts $P = [3, 5]$ to even positions $Q = [4, 6]$ ($s = 2$ elements) at even length $k = 8$:

$$A = [3, 1, 1, \mathbf{2}, 1, \mathbf{2}, 1, 1], \qquad Q = [4, 6], \qquad D = k - q_2 = 8 - 6 = 2.$$

   Notice that for $s = 2$ elements, $2^s - 2 = 2^2 - 2 = 2$, so $D = 2 = 2^s - 2$ is in **tight equality**!

3. **Bounces 26 and 27 ($D = 2$ even-parity countdown steps):**
   - Bounce 26 picks $m = 6$, moving $Q$ from $[4, 6]$ to $[4, 7]$.
   - Bounce 27 picks $m = 4$, moving $Q$ from $[4, 7]$ to $[5, 8]$.

4. **Bounce 28 ($q_2 = 8 = k$ splits):** At Bounce 28, $Q = [5, 8]$ has $q_2 = 8 = k$, so Rule 1 splits the final 2 at index 8 into $1, 1$, increasing the length to $k + 1 = 9$ **(odd)** with off $= 1$, and leaving $Q_{\text{prefix}} = [5]$ (which is already all odd!).

5. **Bounce 29 (Second $m = 0$ bounce, now at odd length 9):** At Bounce 29, $A = [3, 1, 1, 1, \mathbf{2}, 1, 1, 1, 1]$ has odd length 9 and off $= 1$ ($s = 8$ even), and $Q_{\text{prefix}} = [5]$ has no even elements. Thus Bounce 29 hits $m = 0$ with $a_1 = 3$, raising $a_1$ to 4 at Bounce 30 ($A = [4, 2, 1, 1, 1, 2, 1, 1, 1]$).

6. **Bounce 30 and Bounce 34 (Double $m = 1$ reset):** Because $k = 9$ is odd at Bounce 30, Lemma 5.1 immediately triggers $m = 1$ at Bounce 30 (reducing $a_1 : 4 \to 3$), and after the odd countdown $[2, 3, 7] \to [2, 3, 8] \to [2, 4, 9] \to [2, 4]$ finishes, Bounce 34 triggers $m = 1$ a second time (reducing $a_1 : 3 \to 2$).

# 6 The Repeating Growth Cycles (Epochs) From Length $2^{2n} - 1$ to $2^{2n+2} - 1$

To complete the proof and show that $D \leq 2^s - 2$ holds across all time $t \to \infty$, we describe the global rhythm of the list.

**Definition 6.1** (What We Mean by a "Growth Cycle" or "Epoch")**.** As the Turing machine runs, the list length $k$ grows by $+1$ each time a 2 reaches the right end of the list and splits into $1, 1$. Rather than growing chaotically, the list repeatedly passes through a clean, self-similar **landmark configuration** whenever the list length reaches $k_n = 2^{2n} - 1$ (that is, $k = 3, 15, 63, 255, 1023, \ldots$ for $n = 1, 2, 3, 4, 5, \ldots$). At each landmark length $k_n = 2^{2n} - 1$, the machine arrives at an $m = 0$ bounce with:

- off $= 1$ (the list just grew from $2^{2n} - 2$ to $2^{2n} - 1$),
- $a_1 = 2$, and
- the 2s in the tail sitting at the exact power-of-two-minus-one indices:

$$P_n = \left[3, 7, 15, \ldots, 2^{2n-1} - 1\right] = \left[2^j - 1\right]_{j=2}^{2n-1}.$$

We call the complete stage of evolution from the landmark state at length $k_n = 2^{2n} - 1$ to the next landmark state at length $k_{n+1} = 2^{2n+2} - 1$ **Growth Cycle $n$ (or Epoch $n$)**.

Let us trace how Growth Cycle $n$ evolves from $k_n = 2^{2n} - 1$ to $k_{n+1} = 2^{2n+2} - 1$:

1. **Stage 1 (Opening Expansion from $k = 2^{2n} - 1$ to $k = 2^{2n} + 2n$):** Starting at $k_n = 2^{2n} - 1$ with $a_1 = 2$ and $P_n = [3, 7, \ldots, 2^{2n-1} - 1]$, the $m = 0$ bounce raises $a_1$ to 3 and shifts $P_n$ to $[2, 4, 8, \ldots, 2^{2n-1}]$. Because $k_n$ is odd, Lemma 5.1 immediately fires $m = 1$ on the next bounce, reducing $a_1$ to 2 and setting $P = [2, 3, 5, 9, \ldots, 2^{2n-1} + 1]$. As the odd-parity countdown runs, the rightmost 2 reaches the end of the list and splits, and this process repeats $n$ times across odd lengths $k = 2^{2n} + 1, 2^{2n} + 3, \ldots, 2^{2n} + 2n - 1$, arriving at even length $k = 2^{2n} + 2n$ with $m = 0$, $a_1 = 1$, and $2n$ twos in the tail:

$$P = \left[3, 5, 11, 19, \ldots, 2^{2n-1} + 2n - 1,\ 2^{2n} + 2n - 1\right].$$

2. **Stage 2 (Steady Even-Length Climb from $k = 2^{2n} + 2n$ to $k = 2^{2n+2} - 2n - 6$):** For every even length $k$ from $2^{2n} + 2n$ up to $2^{2n+2} - 2n - 8$, the machine hits an $m = 0$ bounce with off $= 2$, $a_1 = 1$, and $s = 2n$ twos in the tail satisfying $D = k - q_{2n} < 2^{2n} - 2$ (Case 1 of Lemma 5.3). Each such cycle raises $a_1 : 1 \to 3$ at even length $k$, splits one 2 to reach odd length $k + 1$, hits $m = 1$ **twice** at odd length $k + 1$ to reduce $a_1 : 3 \to 2 \to 1$, and splits a second 2 to arrive at the next even length $k + 2$ with $m = 0$ and $a_1 = 1$.

3. **Stage 3 (The Single Peak $a_1 = 4$ at $k_{\text{peak}}(n) = 2^{2n+2} - 2n - 5$ and Final Reset to $k_{n+1} = 2^{2n+2} - 1$):** When the even climb reaches $k = 2^{2n+2} - 2n - 6$, the rightmost 2 after the $m = 0$ shift sits at $q_{2n} = 3 \cdot 2^{2n} - 2n - 4$, which gives the exact tight distance:

$$D = k - q_{2n} = (2^{2n+2} - 2n - 6) - (3 \cdot 2^{2n} - 2n - 4) = 2^{2n} - 2.$$

By Case 2 of Lemma 5.3, after $D$ steps and one split, the machine arrives at the odd length

$$k_{\text{peak}}(n) = 2^{2n+2} - 2n - 5, \qquad \text{off} = 1, \qquad m = 0, \qquad a_1 = 3.$$

This single bounce per growth cycle raises $a_1$ from 3 to 4. Because $k_{\text{peak}}(n)$ is odd, Lemma 5.1 immediately fires $m = 1$ on the very next bounce ($a_1 : 4 \to 3$), followed by a second $m = 1$ bounce ($a_1 : 3 \to 2$). Over the final $2n + 4$ length increments, the positions of the 2s regularize one by one into powers of two, arriving at the exact landmark state of Growth Cycle $n + 1$ at $k_{n+1} = 2^{2n+2} - 1$.

Table 3: Summary of the first five Growth Cycles (Epochs $n = 1, 2, 3, 4, 5$), showing the landmark start state at $k_n = 2^{2n} - 1$ and the single peak bounce $k_{\text{peak}}(n) = 2^{2n+2} - 2n - 5$ where $a_1$ briefly touches 4 for one step.

| **Cycle** $n$ | **Start Length** $k_n$ | **Landmark 2-Positions** $P_n$ **(at** $a_1 = 2, m = 0$**)** | **Peak Length** $k_{\text{peak}}(n)$ | **Bounce Where** $a_1 = 3 \to 4 \to 3$ |
|---|---|---|---|---|
| $n = 1$ | $k_1 = 3$ (Bounce 4) | [] ($A = [2, 1, 1]$) | $k = 9$ | Bounce $29 \to 30 \to 31$ |
| $n = 2$ | $k_2 = 15$ (Bounce 69) | $[3, 7]$ | $k = 55$ | Bounce $814 \to 815 \to 816$ |
| $n = 3$ | $k_3 = 63$ (Bounce 1,058) | $[3, 7, 15, 31]$ | $k = 245$ | Bounce $15,285 \to 15,286 \to 15,287$ |
| $n = 4$ | $k_4 = 255$ (Bounce 16,549) | $[3, 7, 15, 31, 63, 127]$ | $k = 1,011$ | Bounce $256,730 \to 256,731 \to 256,732$ |
| $n = 5$ | $k_5 = 1,023$ (Bounce 262,854) | $[3, 7, 15, 31, 63, 127, 255, 511]$ | $k = 4,081$ | Bounce $4,167,679 \to 4,167,680 \to 4,167,681$ |

**Theorem 6.2** (Main Non-Halting Theorem)**.** *The Turing machine* `1RB1LC_1RC---_0LD1RE_0LA1LD_0RC1RF_0LF0RB`*, started on a blank tape, never halts.*

*Proof.* By Section 2.1, the machine halts if and only if it reaches a right bounce with $m = 0$ and $a_1 \geq 4$. Across every Growth Cycle $n \geq 1$ from $k_n = 2^{2n} - 1$ to $k_{n+1} = 2^{2n+2} - 1$, $a_1$ starts at 2 and only ever reaches 4 immediately after the single odd-length bounce at $k_{\text{peak}}(n) = 2^{2n+2} - 2n - 5$, where $m = 0$ and $a_1 = 3$. By Lemma 5.1, because $k_{\text{peak}}(n)$ is odd, the very next bounce hits $m = 1$ and immediately decrements $a_1$ from 4 back to 3. Consequently, no bounce with $m = 0$ ever occurs while $a_1 \geq 4$, and the machine runs forever. □