

Design Turing machines $M = (Q, \Sigma, \Gamma, \delta, \text{start}, \text{accept}, \text{reject})$ for each of the following tasks, either by listing the states Q , the tape alphabet Γ , and the transition function δ (in a table), or by drawing the corresponding labeled graph.

Each of these machines uses the input alphabet $\Sigma = \{1, \#\}$; the tape alphabet Γ can be any superset of $\{1, \#, \square, \triangleright\}$ where \square is the blank symbol and \triangleright is a special symbol marking the left end of the tape. Each machine should **reject** any input not in the form specified below.

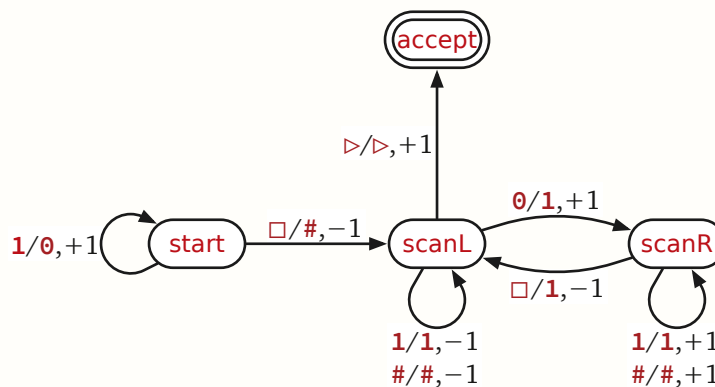
The solutions below describe single-tape, single-head Turing machines. There are arguably simpler Turing machines that multiple tapes and/or multiple heads.

1. On input 1^n , for any non-negative integer n , write $1^n \# 1^n$ on the tape and **accept**.

Solution: Our Turing machine M_1 uses the tape alphabet $\Gamma = \{0, 1, \#, \square, \triangleright\}$ and the following states, in addition to **accept** and **reject**:

- **start** — Initialize the tape by replacing every **1** with **0**. When we find a blank, write **#** and start scanning left.
- **scanL** — Scan left for the rightmost **0**. If we find it, replace it with **1** and start scanning right. If we find \triangleright instead, we're done; halt and accept.
- **scanR** — Scan right for the leftmost blank. When we find it, write **1** and start scanning left again.

Here is the transition graph of the machine. To simplify the drawing, we omit all transitions into the hidden **reject** state.



Here is the transition function; again, all unspecified transitions lead to the **reject**

state.

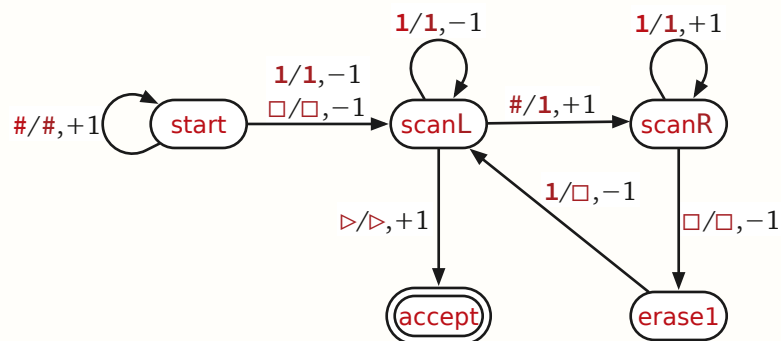
$\delta(p, a) = (q, b, \Delta)$	explanation
$\delta(\text{start}, 1) = (\text{start}, 0, +1)$	init phase: replace 1 s with 0 s
$\delta(\text{start}, \square) = (\text{scanL}, \#, -1)$	finished init phase; write # and start scanning left
$\delta(\text{scanL}, 1) = (\text{scanL}, 1, -1)$	scan left to rightmost 0
$\delta(\text{scanL}, \#) = (\text{scanL}, \#, -1)$	
$\delta(\text{scanL}, 0) = (\text{scanR}, 1, +1)$	found it; write 1 and start scanning right
$\delta(\text{scanL}, \triangleright) = (\text{accept}, \triangleright, +1)$	found start of tape instead; we're done!
$\delta(\text{scanR}, 1) = (\text{scanR}, 1, +1)$	main loop: scan right to leftmost \square
$\delta(\text{scanR}, \#) = (\text{scanR}, \#, +1)$	
$\delta(\text{scanR}, \square) = (\text{scanL}, 1, -1)$	found it; write 1 and start scanning left ■

2. On input $\#^n 1^m$, for any non-negative integers m and n , write 1^m on the tape and **accept**. In other words, delete all the $\#$ s, thereby shifting the 1 s to the start of the tape.

Solution: Our machine M_2 repeatedly scans for the last $\#$ and replaces it with 1 , then scans for the rightmost 1 and replaces it with a blank, until the search for the last $\#$ fails. We use the minimal tape alphabet $\Gamma = \{1, \#, \square, \triangleright\}$ and the following states, in addition to **accept** and **reject**:

- **start** — Scan right past all $\#$ s
- **scanL** — Scan left to the rightmost $\#$ or \triangleright . If we find $\#$, replace it with 1 ; if we find \triangleright , we're done!
- **scanR** — Scan right to the leftmost \square (just after the rightmost 1 , if any).
- **erase1** — Replace the rightmost 1 with \square .

Here is the transition graph of the machine. To simplify the drawing, we omit all transitions into the hidden **reject** state.



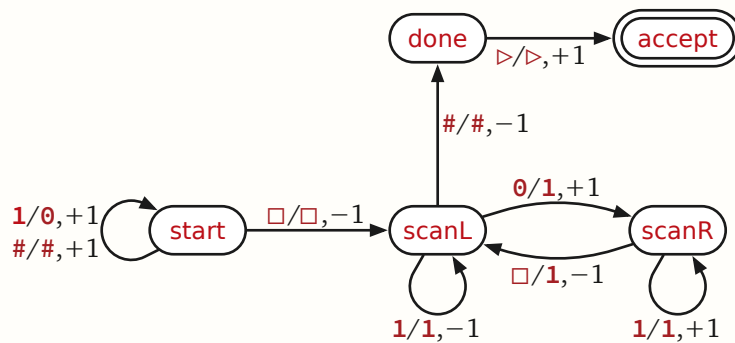
■

3. On input $\#1^n$, for any non-negative integer n , write $\#1^{2n}$ on the tape and **accept**. [Hint: Modify the Turing machine from problem 1.]

Solution: Our machine M_3 mirrors M_1 with a few minor changes. First, we won't both writing a second $\#$ between the first and second copies of the input string; second, we treat the initial $\#$ as the de-facto beginning of the tape. Here are the states:

- **start** — Scan right for first blank, replacing **1**s with **0**s
- **scanL** — Scan left for rightmost **0**, replace with **1**
- **scanR** — Scan right for leftmost blank, replace with **1**
- **done** — Found the initial $\#$; reset the head to the start position and accept

And here is the transition graph, as usual omitting transitions to **reject**.



4. On input 1^n , for any non-negative integer n , write 1^{2^n} on the tape and **accept**. [Hint: Use the three previous Turing machines as subroutines.]

Solution: Our machine M_4 works in several phases:

- Write **#1** at the end of the input string
- Repeatedly transform $1^a \# 1^b$ into $1^{a-1} \#^{b+1} 1^{2c}$ using a small modification of M_3 (which uses M_1 as a subroutine).
- When the initial string of **1**s is empty, remove all **#**s using M_2 .

So here are the states:

- **start**: Scan right for a blank, and write **#**
- **write1**: Write **1** after **#** and start main loop
- three states from M_3 to double the number **1**s to the right of **#**s
- **scanL1**: scan left for rightmost **1** left of **#**s, replace with **#** and repeat main loop
- four states from M_2 to delete the **#**s

