1) There are 7 steps in:

$$
\begin{array}{lll}
(s, \vdash 0010\sqcup^{\infty}, 0) & \xrightarrow[M]{}{}_{\delta(s,\vdash)=(s,\vdash,R)} & (s, \vdash 0010\sqcup^{\infty}, 1) \\
& \xrightarrow[M]{}{}_{\delta(s,0)=(s,0,R)} & (s, \vdash 0010\sqcup^{\infty}, 2) \\
& \xrightarrow[M]{}{}_{\delta(s,0)=(s,0,R)} & (s, \vdash 0010\sqcup^{\infty}, 3) \\
& \xrightarrow[M]{}{}_{\delta(s,1)=(s,1,R)} & (s, \vdash 0010\sqcup^{\infty}, 4) \\
& \xrightarrow[M]{}{}_{\delta(s,0)=(s,1,R)} & (s, \vdash 0010\sqcup^{\infty}, 5) \\
& \xrightarrow[M]{}{}_{\delta(s,\sqcup)=(p,\sqcup,L)} & (p, \vdash 0010\sqcup^{\infty}, 4) \\
& \xrightarrow[M]{}{}_{\delta(p,0)=(t,1,L)} & (t, \vdash 0011\sqcup^{\infty}, 3)
\end{array}
$$

1. On input ⊢`0010`, the following TN halts after 7 steps with ⊢`0011` on the tape.

```
0 ⊢ ⊢ r 0
0 0 0 r 0
0 1 1 r 0
0 _ _ l p
p ⊢ ⊢ r halt
p 0 1 l halt
p 0 1 l halt
```

2) • The following TM computes n^2 in unary, by n times copying n. In it, `C` marks the last copied position, the position of `K` $\bmod n$ is the number of the iteration (+1 if `K` is left of `L`), and `L` marks the leftmost bit of the working copy of n at the tail. E.g. ⊢`11111C1K11L` indicates that we are in the process of producing the 3rd copy of `11111` (`K` is left of `L`, at position $2 \bmod 5$), and we have just copied its leftmost symbol `L` (as marked by the `C`).

```
0 ⊢ ⊢ r init; initialisation, detect 0,1 cases, otherwise put markers C,K,L
init _ _ * halt; 0^2 = 0
init 1 C r initC
initC _ _ l one
one C 1 r halt; 1^2 = 1
initC 1 K r appendL
shiftCright 1 C r append1; loop, shifting C to the right, stopping when K,L adjacent
shiftCright L C r appendL
shiftCright K 1 r shiftCKright
shiftCKright L 1 r append1halt; adjacent so clean up and halt
shiftCKright 1 C r append1K
left C 1 r shiftCright; searching left for marker C and then shift it to the right
left * * l left
appendL _ L l left ; appending various symbols
appendL * * r appendL
append1 _ 1 l left
append1 * * r append1
append1halt _ 1 r halt
append1halt * * r append1halt
append1K _ 1 r appendK
```

```
append1K * * r append1K
appendK _ K l left; (only called if symbol is known to be a blank)
```

- ```
  data Nat = Zer | Suc Nat deriving Show
  square :: Nat -> Nat
  square x = times x x
  times, add :: Nat -> Nat -> Nat
  times Zer k = Zer
  times (Suc n) k = add k (times n k)
  add Zer k = k
  add (Suc n) k = Suc (add n k)
  ```

3)
- Suppose there were $i \leq j$ such that $n_i \not\leq n_j$ with $j - i$ minimal. Then $j - i \geq 2$ ($i = j$ contradicts reflexivity of $\leq$, and $i + 1 = j$ local $\leq$-sortedness). Hence there is $i < k < j$ with $k - i, j - k < j - i$, so $n_i \leq n_k \leq n_j$. But then $n_i < n_j$ by transitivity. Contradiction.
- Take $R = \{(n, n), (n, n + 1), (m, n) \mid m, n \in \mathbb{N}, m > n + 1\}$ and the list $[1, 2, 3]$. The important point is that R is not transitive: $1\ R\ 2$ and $2\ R\ 3$, but not $1\ R\ 3$. (This R is even total.)

(∗) Suppose $l = [n_1, \ldots, n_k]$ is a locally R-sorted list with R a total relation, and that we have arrived at position $i + 1$ trying to insert m, starting from the head. Because we have arrived there, we know that $m \not R\ n_i$, hence by totality $n_i\ R\ m$. If $m\ R\ n_{i+1}$, we may insert m and have local R-sortedness: $n_i\ R\ m\ R\ n_{i+1}$. Otherwise, we continue with $i + 2$.

4∗) The idea is that a base n-number is divisible by $n-1$ iff its digits sum up to $n-1$. For instance, in base-10 the number 684 is divisible by 9 since $6 + 8 + 4 = 14 + 4 = 1 + 4 + 4 = 5 + 4 = 9$. The process only needs numbers < 10; in the TM only < 4, the states. The complexity is n.

```
0 ⊢ ⊢ r 0
0 _ * r halt-reject
0 0 * r 0
0 1 * r 1
0 2 * r 2
0 3 * r 3
1 _ * r halt-reject
1 0 * r 1
1 1 * r 2
1 2 * r 3
1 3 * r 1
2 _ * r halt-reject
2 0 * r 2
2 1 * r 3
2 2 * r 1
2 3 * r 2
3 _ * r halt-accept
3 0 * r 3
3 1 * r 1
3 2 * r 2
3 3 * r 3
```
```