1. Let M be a Turing machine, let w be an arbitrary input string, and let s be an integer. We say that ***M accepts w in space s*** if, given w as input, M accesses only the first s (or fewer) cells on its tape and eventually accepts.

   *(a) Sketch a Turing machine/algorithm that correctly decides the following language:

$$\textsc{SquareSpace} = \left\{ \langle M, w \rangle \;\middle|\; M \text{ accepts } w \text{ in space } |w|^2 \right\}$$

Solution (timed simulation): We exploit two facts about Turing machines:

- Turing machines use only *local* memory; if a TM uses s cells on its tape, those are the *first s consecutive* cells on the tape.
- Each cell on the tape holds an element of a *finite* tape alphabet Γ.

Any Turing machine that uses only the first s cells of its tape must be in one of $|Q| \cdot s \cdot |\Gamma|^s$ configurations—there are $|Q|$ possible states, s possible head positions, and $|\Gamma|^s$ possible tape contents. Thus, if the Turing machine runs for more than $|Q| \cdot s \cdot |\Gamma|^s$ steps, it must either use more than s cells of its tape, or revisit a configuration is has previously seen, trapping the machine in an infinite loop.

Let U be any universal Turing machine. Let $\langle M, w \rangle$ be an input string for U, encoding some Turing machine M and some string w as input to M. Without loss of generality, we assume that U has two tapes: One storing the verbatim contents of M's single work tape, the other storing the encoding $\langle M \rangle$, the current state of M, and other bookkeeping information to drive the simulation of M.

We modify U into a new Turing machine U^* by adding a third "clock" tape and a new preprocessing phase:

- First, U^* writes a special symbol ■ onto M's tape at position $|w|^2 + 1$, and then returns the main tape's head to the left end of the tape.
- Second, U^* writes ■ onto the clock tape at position $|Q| \cdot |w|^2 \cdot |\Gamma|^{|w|^2}$, and then returns the clock tape's head to the left end of the tape.
- Both of these preprocessing steps require moving one of U^*'s heads to a particular position on its tape. In each case, we use another auxilliary tape to calculate a string of 1s of the appropriate length, using Turing machines for unary multiplication as subroutines, starting with strings $1^{|w|}$, $1^{|Q|}$, and $1^{|\Gamma|}$ as input.

After the preprocessing phase, U^* simulates the execution of M on input w, using the same instructions as U, but with two additional rejection criteria.

- If the head on M's tape ever reads the symbol ■, then U^* halts and rejects, because M has used more than $|w|^2$ space.
- After simulating each M-transition, U^* moves the clock-tape's head one step to the right. If the clock tape's head ever reads the symbol ■, then U^* halts and rejects, because M is stuck in an infinite loop.
- If neither of the first two cases apply, then M must eventually halt after using at most $|w|^2$ tape cells. As usual, if M accepts, then U^* accepts, and if M rejects, then U^* rejects.

We conclude that U^* decides $\textsc{SquareSpace}$. ■

Solution (graph analysis): We exploit two facts about Turing machines:

- Turing machines use only *local* memory; if a TM uses s cells on its tape, those are the *first* s *consecutive* cells on the tape.
- Each cell on the tape holds an element of a *finite* tape alphabet Γ.

Any Turing machine that uses only the first s cells of its tape must be in one of $|Q| \cdot s \cdot |\Gamma|^s$ configurations—there are $|Q|$ possible states, s possible head positions, and $|\Gamma|^s$ possible tape contents. Thus, if the Turing machine runs for more than $|Q| \cdot s \cdot |\Gamma|^s$ steps, it must either use more than s cells of its tape, or revisit a configuration is has previously seen, trapping the machine in an infinite loop.

Moreover, each configuration (q, i, T) of a Turing machine either is a halting configuration (either $q = \text{accept}$ or $q = \text{reject}$) or has a unique successor configuration, defined by the machine's transition function. Specifically, if $\delta(q, T_i) = (q', \delta, \alpha)$, then the successor of (q, i, T) is $(q', i + \Delta, T')$, where $T' = T[1..i-1] \cdot \alpha \cdot T[i+1..s]$.

Based on these observations, we can solve SQUARESPACE as follows. Given a machine encoding $\langle M \rangle$ and string w, we construct a directed graph $G = (V, E)$, where

- V is the set of all legal configurations $x = (q, i, T)$ where $q \in Q$, $1 \le i \le |w|^2$, and $T \in \Gamma^{|w|^2}$.
- E is the set of all legal transitions from one configuration to the next.

Let $s = (\text{Start}, 1, w \bullet \square^{|w|^2 - |w|})$ denote the starting configuration for M running on input w. Then M accepts w using at most $|w|^2$ space if and only if some accepting configuration is reachable from s, which we can determine via whatever-first search. Equivalently, because every node in G has out-degree at most one, we can just follow the edges, marking each node as we visit it, reporting success if we reach an accepting vertex, and reporting failure if we reach either a rejecting vertex or a marked vertex.

(Following the edges is also known as "simulating M".) ■

Rubric: 5 points:

- For simulation:
  - + 1 for modifying universal Turing machine (or other TM simulation)
  - + 1 for correctly limiting simulation space
  - + 2 for correctly limiting simulation *time*
  - + 1 for other details
- For configuration graph: standard graph reduction rubric (scaled)

These are not the only correct solutions. These solutions are more detailed than necessary for full credit.

(b) Prove that the following language is undecidable:

$$\textsc{SomeSquareSpace} = \left\{ \langle M \rangle \;\middle|\; M \text{ accepts at least one string } w \text{ in space } |w|^2 \right\}$$

Solution (reduction from HALT): For the sake of argument, suppose there is an algorithm DECIDELOWSPACE that correctly decides the stated language. Then we can solve the halting problem as follows:

```
DECIDEHALT(⟨M, w⟩):
  Encode the following Turing machine M′:
    M′(x):
      run M on input w
      return TRUE
  return DECIDELOWSPACE(⟨M′⟩)
```

(We seem to use this reduction a lot, don't we?) We prove this reduction correct as follows. Without loss of generality, assume that the input alphabet contains the symbol 1.

$\Longrightarrow$ Suppose M halts on input w.
Let s be the number of cells that M accesses on its tape given input w.
Then M' accepts *every* input string x ***using space*** s.
In particular, M' accepts the string 1^s using space $s \le |1^s|^2$.
So DECIDELOWSPACE must accept the encoding $\langle M' \rangle$.
We conclude that DECIDEHALT correctly accepts the encoding $\langle M, w \rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.
Then M' diverges on *every* input string x.
In particular, M' does not accept any string w (in space $|w|^2$ or otherwise).
So DECIDELOWSPACE must reject the encoding $\langle M' \rangle$.
We conclude that DECIDEHALT correctly rejects the encoding $\langle M, w \rangle$.

In both cases, DECIDEHALT is correct. But that's impossible, because HALT is undecidable. We conclude that the algorithm DECIDELOWSPACE does not exist. ■

Rubric: 5 points: standard undecidability reduction rubric (scaled). This is not the only correct solution. Notice that Rice's Theorem cannot be used here.

2. Consider the following language:

$$\textsc{Picky} = \left\{ \langle M \rangle \;\middle|\; \begin{array}{c} M \text{ accepts at least one input string} \\ \text{and } M \text{ rejects at least one input string} \end{array} \right\}$$

(a) Prove that PICKY is undecidable.

Solution (reduction from HALT): We can reduce the standard halting problem to PICKY as follows:

```
DecideHalt(⟨M⟩, w):
  Encode the following Turing machine M′:
    M′(x):
      if x = w
          run M on input w
          return True
      else
          return False
  return DecidePicky(⟨M′⟩)
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w. Then M' accepts w but rejects every other input string. So $\langle M' \rangle \in$ PICKY. So DECIDEPICKY accepts $\langle M' \rangle$. So DECIDEHALT correctly accepts $\langle M \rangle, w$.

$\Longleftarrow$ Suppose M does not halt on input w. Then M' diverges on w but rejects every other input string. So $\langle M' \rangle \notin$ PICKY. So DECIDEPICKY rejects $\langle M' \rangle$. So DECIDEHALT correctly rejects $\langle M \rangle, w$.

In both cases, DECIDEHALT is correct. But that's impossible, because HALT is undecidable. We conclude that the algorithm DECIDEPICKY does not exist. ■

Rubric: 5 points: standard undecidability rubric (scaled). These are not the only correct solutions. Notice that Rice's Theorem cannot be used here.

(b) Sketch a Turing machine/algorithm that *accepts* PICKY.

Solution: The following algorithm uses a universal Turing machine with a timer, similar to problem 1(a), to simulate the encoded Turing machine M.

```
ACCEPTPICKY(⟨M⟩):
  accepted ← FALSE
  rejected ← FALSE
  for L ← 1 to ∞
      for all strings w ∈ Σ* with |w| ≤ L
          simulate M on input w for L steps
          if M(w) accepts before step L
              accepted ← TRUE
          if M(w) rejects before step L
              rejected ← TRUE
          if accepted ∧ rejected
              return TRUE
```

Each iteration of the outer loop executes in finite time, because there are only finitely many strings of length at most L and we simulate M on each of those input strings for a finite number of steps. Suppose M accepts input string w after T steps, and rejects some string w' after T' steps. Then ACCEPTPICKY will halt and return TRUE after at most $\max\{T, T'\}$ iterations of the outer loop. ■

Rubric: 5 points = 1 for using universal TM (or other TM simulation) + 1 for timer + 2 for dovetailing details + 1 for correctness argument. This is not the only correct solution.