Student Information

Name : Emre Geçit
ID : 2521581

Question 1

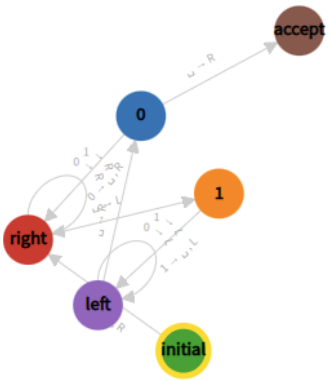
Design a Turing machine which recognizes the language $L = \{0^N 1^N | N \geq 1\}$.
$\Sigma = \{0, 1, \sqcup\}$, means that you cannot write any other symbol than these symbols.

States:

- initial (Initial State): This state's purpose is rejecting empty strings. If the tape head is on a 0, it will go to state "right", otherwise the machine will reject the string and halt.
- right: This state will drive the tape-head to the right.
- 1: This state will check if the righmost symbol of the string is 1. If it is, changes it with a blank symbol, and changes machine state to "left".
- left: This state will drive the tape-head to the left.
- 0: This state will check if the leftmost symbol of the string is 0. If it is, changes it with a blank symbol, and changes machine state to "right". Otherwise, the machine will reject the string and halt.

Note that in these descriptions, undefined transititons leads to rejection of the string.

Figure 1: Turing machine which recognizes the language $L = \{0^N 1^N | N \geq 1\}$



Sample inputs:

Figure 2: Input = 000111

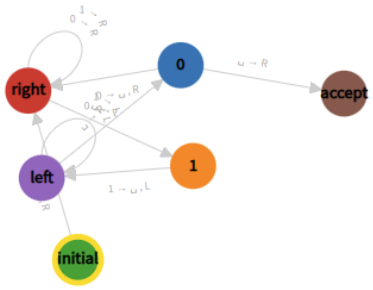


Figure 3: 000111 accepted

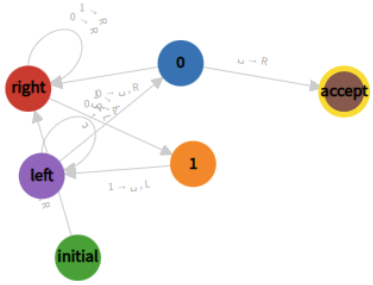


Figure 4: Input = 0000111

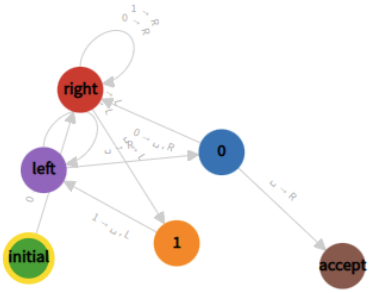


Figure 5: 0000111 rejected

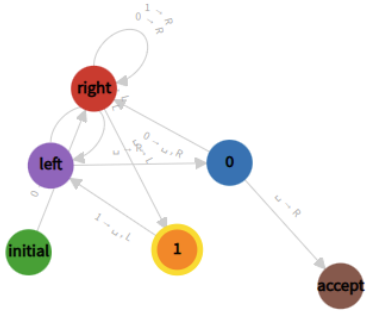


Figure 6: Input = 000011111

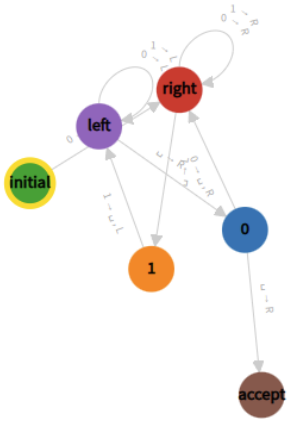
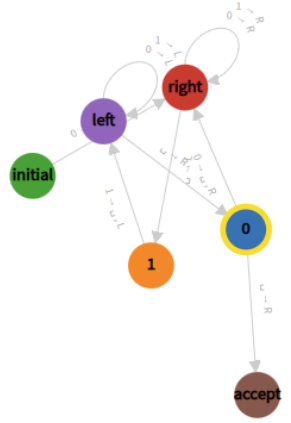


Figure 7: 000011111 rejected



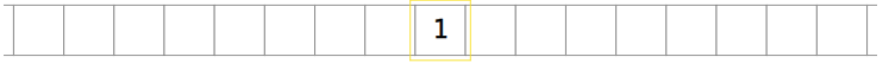


Figure 8: Input = 0001110

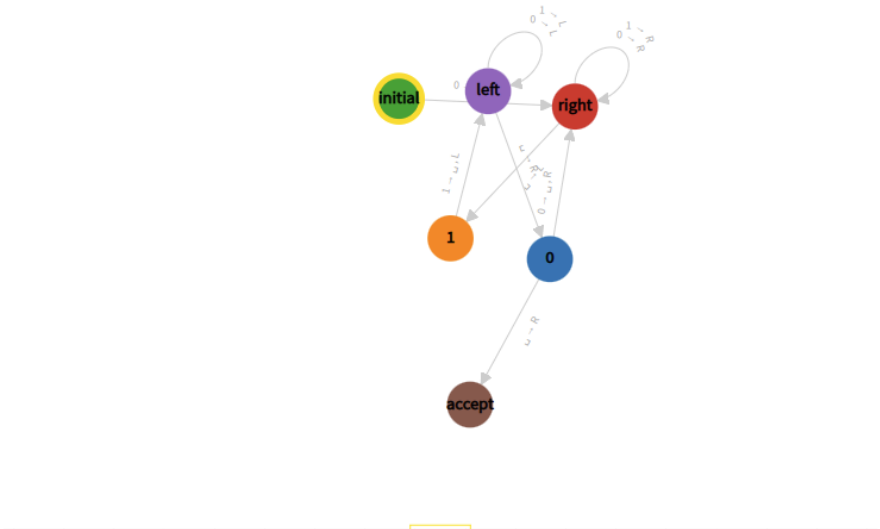


Figure 9: 0001110 rejected

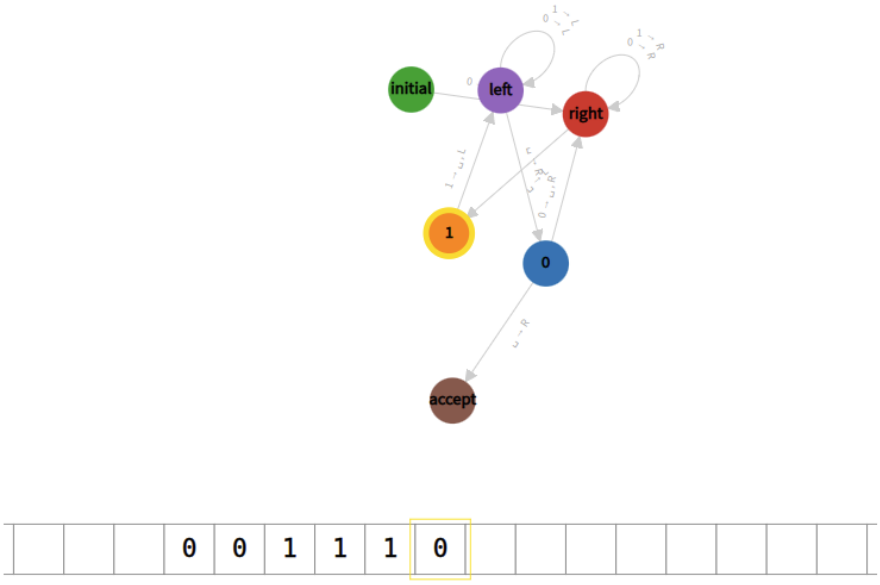


Figure 10: Input = 100011

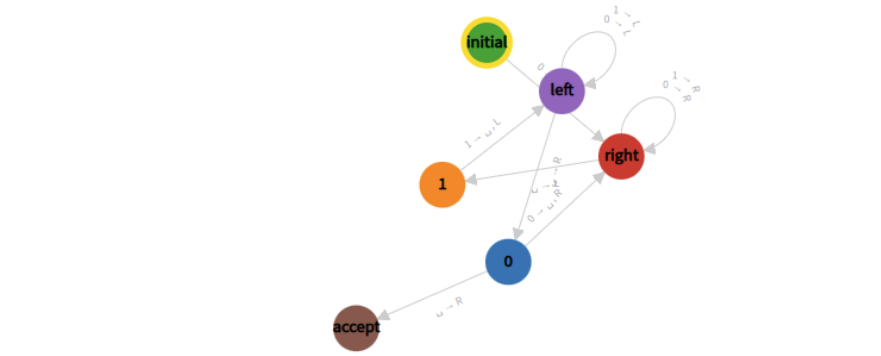
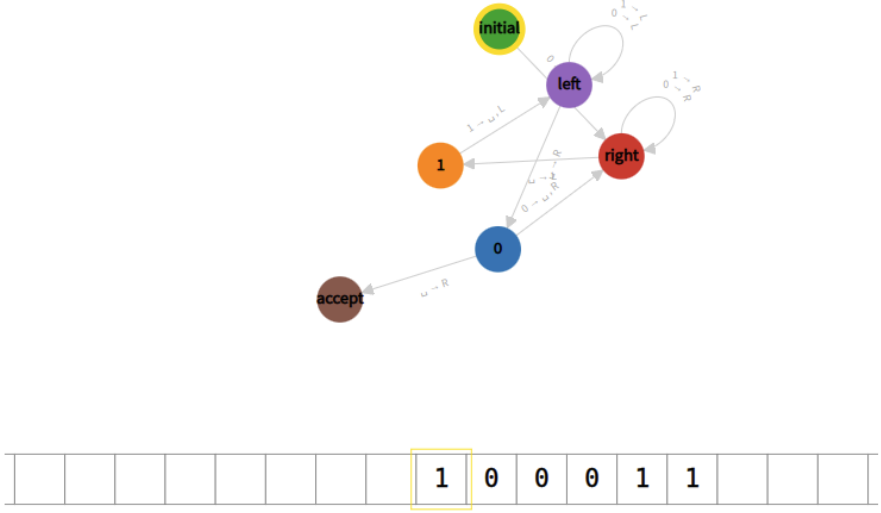


Figure 11: 100011 rejected



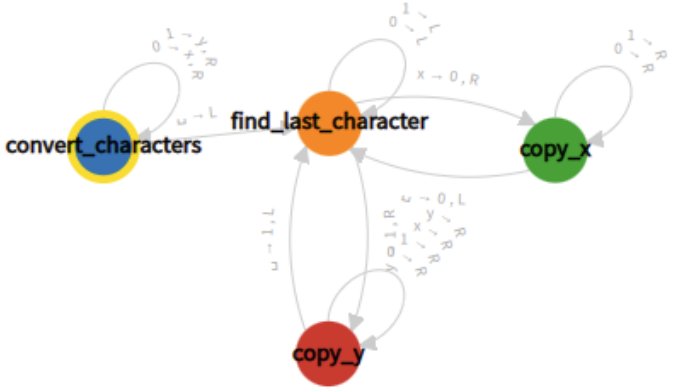
Question 2

Design a Turing machine that computes the function $f(w) = ww^R$, $\Sigma(w) = \{0, 1\}$.

States:

- convert_symbols (Initial state): This states changes all zeros with x's, and all ones with y's. Then, it moves the tape head to the right.
- find_last_character: This state finds the last symbol that has not been copied and passes to copy_y if the tape head is on a y, or to copy_x if the tape head is on a x.
- copy_y: This state goes to the end of the tape, and writes a 1.
- copy_x: This state goes to the end of the tape, and writes a 0.

Figure 12: Turing machine which computes the function $f(w) = ww^R$



Sample inputs:

Figure 13: Input = 1011

Figure 14: Output = 10111101

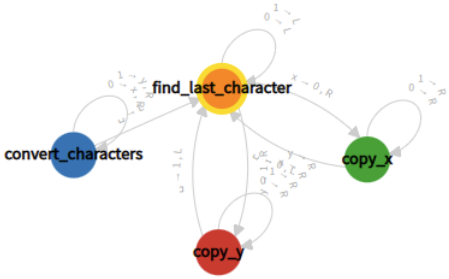


Figure 15: Input = 1110

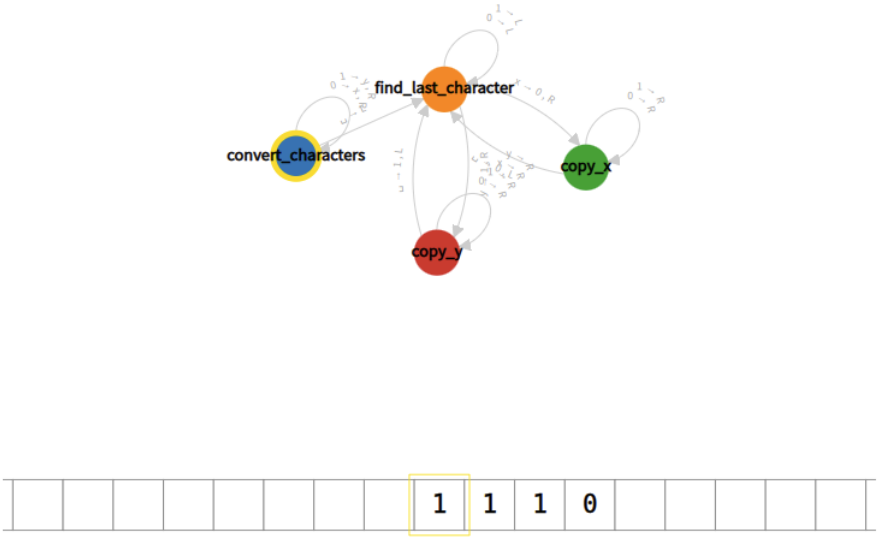


Figure 16: Output = 11100111

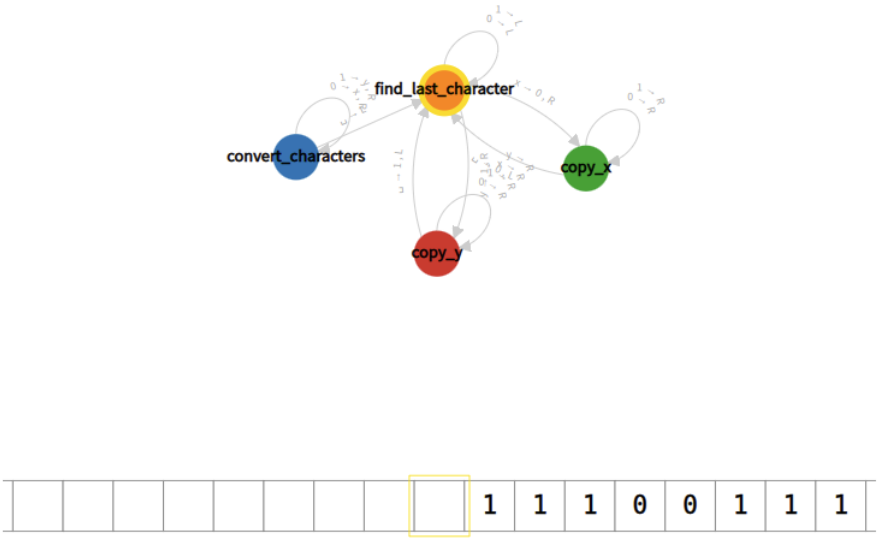


Figure 17: Input = 0101

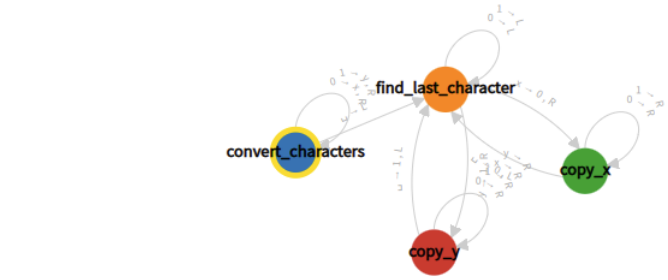


Figure 18: Output = 01011010

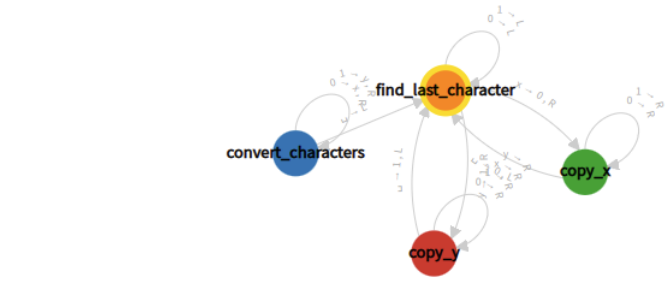


Figure 19: Input = 1010

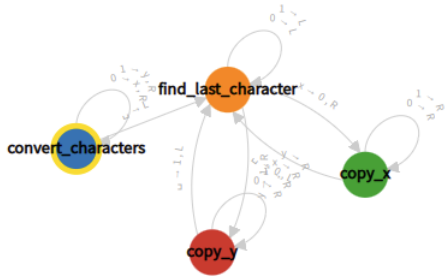


Figure 20: Output = 10100101

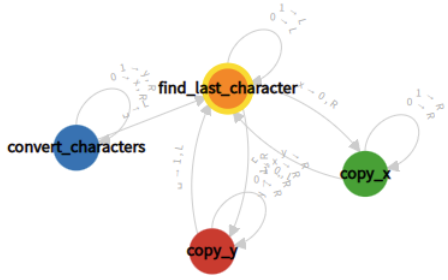


Figure 21: Input = 1010001

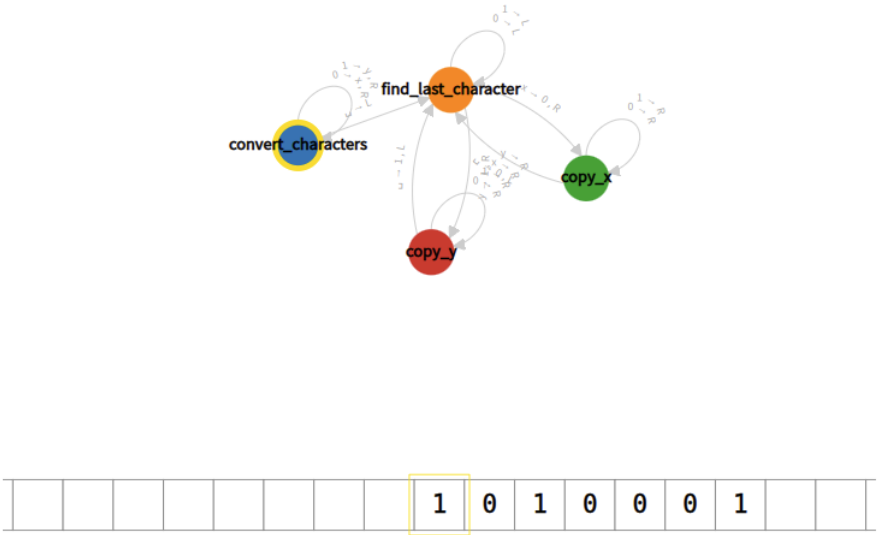


Figure 22: Output = 10100011000101

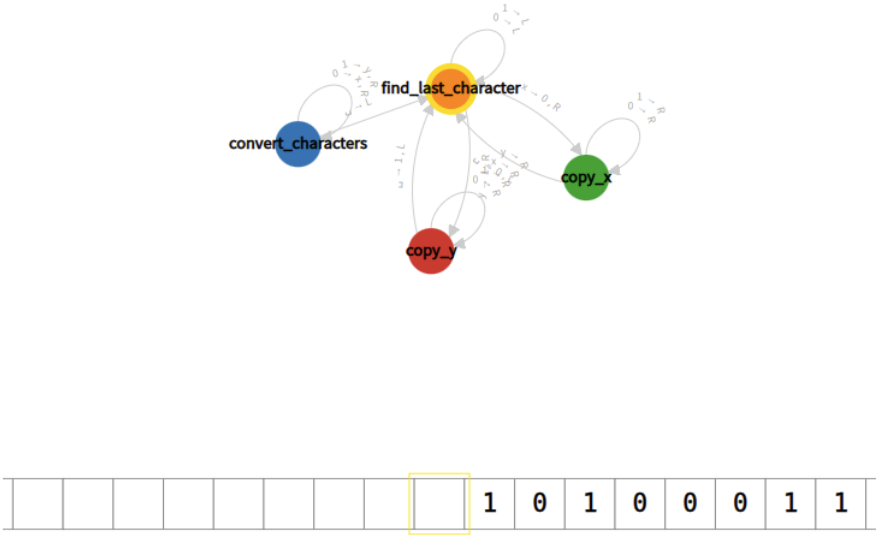


Figure 23: Input = 00111

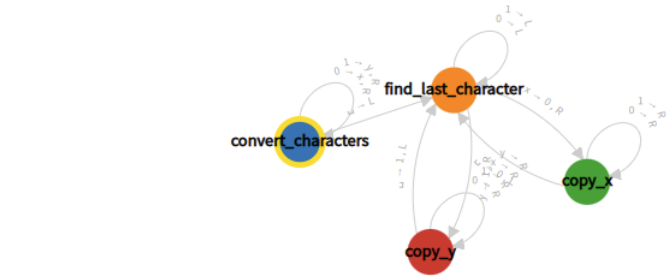
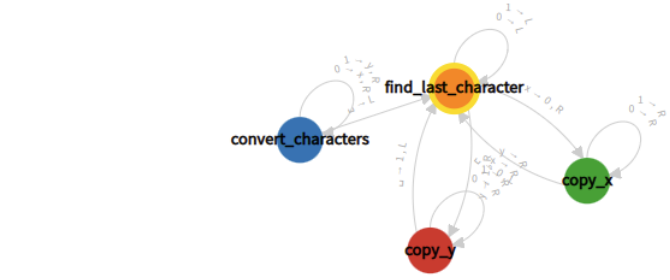


Figure 24: Output = 0011111100



Question 3

Formally define a Turing machine with a 2-dimensional tape, its configurations, and its computation. Define what it means for such a machine to decide a language L. Show that t steps of this machine, starting on an input of length n, can be simulated by a standard Turing machine in time that is polynomial in t and n.

Solution:

1) A Turing machine with a 2-dimensional tape is a pentuple $(K, \Sigma, \delta, s, H)$, where

- K is a finite set of states;
- Σ is a finite set of symbols containing the blank symbol $\sqcup$ but not containing $\uparrow$, $\rightarrow$, $\downarrow$ or $\leftarrow$;
- $s \in K$ is the initial state;
- $H \subseteq K$ is the set of halting states;
- δ, the transititon function, $\delta : (K - H) \times \Sigma \rightarrow (K - H) \times (\Sigma \cup \{\uparrow, \rightarrow, \downarrow, \leftarrow\})$ such that,

(a) for all $q \in K - H$, if $\delta(q, \triangleright) = (p, b)$, then $b = \rightarrow$

(b) for all $q \in K - H$, if $\delta(q, \bigtriangledown) = (p, b)$, then $b = \downarrow$

(c) for all $q \in K - H$ and $a \in \Sigma$, if $\delta(q, a) = (p, b)$, then $b \neq \triangleright$ and $b \neq \bigtriangledown$

A configuration of a Turing machine with a 2-dimensional tape is

$$K \times N \times N \times T,$$

Where T is the set of functions from $N \times N$ to Σ, that have $t(0, y) = \triangleright$ and $t(x, 0) = \bigtriangledown$ for all $x, y \in \mathbb{N}$ and have $t(x, y) =$ for all but a finite number of (x, y) pairs. Thus, we represent a configuration by the current state, current head position, and a list of all non-blank squares on hte tape.

Given a string w, let $t_w \in T$ be the function that has $t(i + 1, 1) = w(i)$ for $0 < i \leq |w|$, $t(0, y) = \triangleright$ for $y \in \mathbb{N}$, $t(x, 0) = \bigtriangledown$ for all $x > 0$ and $t(x, y) = \sqcup$ otherwise. Then if we have a two-dimensional tape Turing machine M with two distinguished halting states y and n such that for any string w either $(s, 1, 1, t_w) \vdash_M^* (y, i, j, t')$ or $(s, 1, 1, t_w) \vdash_M^* (n, i, j, t')$ for some i, j and $t' \in T$, we let the language decided by M be the set of strings for which M thus halts in the y state.

From the definition of T, any $t \in T$ can be encoded as a finite set of ordered triples (x, y, σ). If we encode these values along a tape, we can use dovetailing to assign a unique square for each possible x, y value. We then fill in the non-blank entries of t in this tabular format, so that $t(x, y)$ is stored in the $\dfrac{(i + j)^2 + 3i + j}{2}$th square. At any time during the computation, only finitely many squares of this encoding tape will be non-blank.

Therefore, we simulate M with a three-tape Turing machine M'. M' uses one tape for calculation, one to hold the encoding of the tape of M as described above, and takes its input on the third. The calculations M needs to perform are as follows:

1. increment and decrement two registers, i and j

2. simple multiplications and additions

In each step of the computation, M' looks at the current symbol on the encoding tape and at the current state of M. If the instrucions are to write a symbol, M' writes that symbol and proceeds. If the instrucions are to move, M' increments or decrements i or j as appropriate, then calculates the value of the dovetail function. M' then moves the encoding-tape head all the way to the left, then out to the appropriate square.

At any time, if i and j represents the largest x and y coordinates that M has been in, then neither i or j is larger than t. Thus the process of simulating one move takes the time required to increment or decrement a binary register, compute some binary sums and products, and move one head by a number of spaces that is a quadratic polynomial in i and j, and thus is bounded by a quadratic polynomial in t. The increments and arithmetic are fast, and can be done in polynomial time in t. Thus the time to carry out t steps, once we have finished the initial setup, is $\mathcal{O}(t^3)$. The initial setup consists copying out the tape of M into its M' and takes 2n-step computation of alternating writes and moves, so that it can be carried out in time $\mathcal{O}(n^3)$ by the same reasoning. Thus the overall simulation proceeds in time polynomial in t and n.

Up to here, we assumed that there will be a finite number of non-blank squares. But theoretically, the tile of the 2d-tape can be filled infinitely, and there can be an infinite number of non-blank squares. In that case, the Turing machine will fail halting, and can never be simulated. Considering this possibility, I suggest continuing the copying process, as the computation proceeds, instead of first copying the tape, and than doing the simulation. That is, in each state, if the read square on the encoding tape, if the read square is blank, enter another state, copy the corresponding symbol, and continue the computation. With this process, we need to be careful when copying a blank symbol from the 2d-tape, to the encoding tape. In order not to enter in an infinite loop, when copying a blank symbol, we need to write a special symbol to the encoding tape. This new algorithm runs in a similar time as the original algorithm, but it works with tapes that have infinitely many non-blank squares.