How to Simulate a Turing Machine?

What do we need to describe a Turing machine? For simplicity, let us consider Turing machines for acceptance and rejection, not for computations. So what do we need in this case?

- First, we need to know how many states the head will have. Let us denote this number by N.
- For simplicity of simulation, let us not worry about fancy names for the states, let us denote these states by q_0, q_1, $\ldots$, q_{N-1}.
- In the simulating program, we will simply represent each state by the corresponding index:
  - the state q_0 will be represented as 0,
  - the state q_1 will be represented as 1, etc.
- We need special states: start, accept, and reject. Let us assume:
  - that q_0 is the start state,
  - that q_{N-2} is the accept state, and
  - that q_{N-1} is the reject state.
- We also need to know what symbols are allowed. First, we need to know the number M of the symbols.
- For simplicity, let us denote the symbols simply as s_0, s_1, $\ldots$, s_{M-1}.
- In the simulating program, we will simply represent each symbol by the corresponding index:
  - the symbol s_0 will be represented as 0,
  - the symbol s_1 will be represented as 1, etc.
- We need a special symbol "blank". Let us assume that this symbol is s_0.

At each moment of time, we also need to describe which symbol is in each cell. This can be describe by an integer array `tape[n]`, so that:

- the value `tape[0]` describes the contents of the first cell 0; for example, if in this cell, we have a symbol s_2, then we take `tape[0]` $= 2$;

- the value `tape[1]` describes the contents of the second cell 1; for example, if in this cell, we have a symbol s_5, then we take `tape[1]` $= 5$; etc.

At each moment of time, we also need to know:

- in what state the head is; we will denote this state by `head`, and
- at what cell the head is; we will denote this location by a variable `location`. In the beginning, the head points to the first cell 0, so in the beginning, we have `location` $= 0$.

We also need to describe how the configuration changes. As we have mentioned, when the head in state q_n sees a symbols s_m, it can do three things:

- it can change its state;
- it can replace the original symbol with some other symbol; and also
- it can move one step to the left (L) or to the right (R) or stay in place.

This behavior can be described by three 2-D arrays:

- An integer array `state[n][m]` that describes to what state the head of the Turing machine changes if it was in the state q_n and it sees the symbol s_m. If we have reached an accept or reject state, i.e., $n = N-2$ or $n = N-1$, the Turing machine stops, so we only need to describe the values `state[n][m]` for $n < N - 2$. In other words, we can define this array as

  ```
  int[][] state = int[N - 2][M];
  ```

- An integer array `symbol[n][m]` that describes what symbol will be placed on the tape when the head in the state q_n sees the symbol s_m (it may be the same symbol as before, or it may be some other symbol written by the Turing machine).
- Finally, a character array `lr[n][m]` that describes, for each state q_n and for each symbol s_m, whether the Turing machine:
  - moves to the left (L),
  - moves to the right (R), or
  - stays in place (this will be described by a blank symbol).

Example 1. Le us assume that the Turing machine has the following rule:

$$\text{back, a} \to \text{test, b, L}$$

This rule means that when the head is in the state "back" see a symbol "a", then:

- the state of the head changes to "test";

- the symbol changes to "b", and
- the head moves one step to the left.

How will rule be described in terms of the arrays? Let us assume that:

- the state "back" is No. 3 in our ordering of states, i.e., is the state q_3,
- the state "test" is q_5,
- the symbol "a" is the symbol s_2, and
- the symbol "b" is the symbol s_6.

Then:

- the value `state[3][2]` is equal to 5, meaning that if the head is in the state q_3 (i.e., in the state "back") and it sees the symbol s_2 (i.e., the symbol "a"), then the state changes to q_5 (i.e., to the state "test");
- the value `symbol[3][2]` is equal to 6, meaning that if the head is in the state q_3 (i.e., "back") and it sees the symbol s_2 (i.e., "a"), then the corresponding symbol on the tape is changed to s_6 (i.e., to the symbol "b"); and
- the value `lr[3][2]` is equal to 'L', meaning that if the head is in the state q_3 (i.e., "back") and it sees the symbol s_2 (i.e., "a"), then the head moves one step to the left (which we describe by the symbol L).

Example 2. With the same numbering of states and symbols, if we have a slightly different rule

$$\text{back, a} \to \text{test, L}$$

in which the symbol is not changed (i.e., we keep the same symbol "a"), then:

- the values `state[3][2]` and `lr[3][2]` are the same as in Example 1, but
- the value `symbol[3][2]` changes: now this value is 2, meaning that we keep the same symbol s_2 (i.e., "a").

Example 3. If the rule is as follows:

$$\text{back, a} \to \text{L}$$

then the state of the head also does not change. So, in contrast to the two previous examples, we will now have the value `state[3][2]` equal to 3 – meaning that we keep the same state q_3 (i.e., the state "back").

How to simulate the Turing machine: idea. In the beginning, the head is near cell 0 and in state q_0. So, we must have `location` $= 0$ and `head` $= 0$:

```
  int location = 0;
  int head = 0;
```

To simulate what happens next, we need a loop. As we have mentioned earlier, there are two main types of loops:

- for-loops, when we know how many iterations we need, and
- while-loops, when we do not know beforehand how many iterations we will need.

Here, clearly, in general, we do not know how many iterations we will need, so it should be a while-loop. We stop when we reach either the accept state $N-2$ or the reject state $N-1$. If have any state $i < N-2$, we continue, so the header of the while-loop must take the following form:

```
  while(head < N - 2){
    ...}
```

What should we have inside the loop? The head is at the cell `location`, the symbol it sees is the symbol `tape[location]`; so we can write

```
    currentSymbol = tape[location];
```

Depending on the state `head` and on the symbol `currentSymbol`, we change the symbol `tape[location]` to `symbol[head][currentSymbol]`:

```
   tape[location] = symbol[head][currentSymbol];
```

We also change the state and the location:

```
   newHead = state[head][currentSymbol];
   if(lr[head][currentSymbol] == 'R') {location++;}
     else if(lr[head][currentSymbol] == 'L') {location--;}
   head = newHead;
```

At the end, if we reach the state $N-2$, we accept, else we reject.

So, we get the following pseudo-code:

```
public static Boolean check(){
  int location = 0;
  int head = 0;
  int currentSymbol;
  while(head < N - 2){
    currentSymbol = tape[location];
    tape[location] = symbol[head][currentSymbol];
    newHead = state[head][currentSymbol];
    if(lr[head][currentSymbol] == 'R') {location++;}
      else if(lr[head][currentSymbol] == 'L') {location--;}
    head = newHead;}
  if(head == N - 2){return true;}
    else{return false;}
}
```