CPSC 511/611 — Assignment #1

Solutions for Problem #1

Note: This might have been unclear from the lectures (when "time-constructible" functions were defined) or from the assignment: It could be assumed, here that $T(n) \geq n$ for every integer $n \geq 0$. The "initialization" phase described here is too expensive, otherwise, and students will not be penalized if they have made this assumption without realizing it.

Solution: Consider a multi-tape deterministic Turing machine $\widehat{M}$ that simulates a deterministic Turing machine M with k two-dimensional tapes. Each tape of M will be represented using three tapes of $\widehat{M}$. In particular, for each integer i such that $1 \leq i \leq k$,

- Tape $3i-2$ of $\widehat{M}$ will be used to store an unpadded decimal representation of the (positive integer) index of the ***row*** where the tape head for M's i^{th} tape head is presently located.
- Tape $3i-1$ of $\widehat{M}$ will be used to store an unpadded decimal representation of the (positive integer) index of the ***column*** where the tape head for M's i^{th} tape head is presently located.
- Tape $3i$ of $\widehat{M}$ will be used to store the ***contents*** of M's i^{th} tape. In particular, this tape is used to store encodings of 3-tuples

  $$(r, c, \sigma)$$

  where r is a (positive integer) *row*, c is a positive integer *column*, and σ is a nonblank element of the tape alphabet Γ of M. A 3-tuple of this form should be included for every non-blank cell on this tape.

  It can be assumed that these 3-tuples are sorted — in non-descending order of the row r and, for 3-tuples for the same row, by ascending order of column.

$\widehat{M}$ should also have an input tape (which might be called "tape 0") and at least one "scratch tape" that is used for intermediate computations during the the machine's simulation of M.

M should have tape alphabet

$$\Gamma \cup \{\texttt{(}, \texttt{)}, \texttt{,}, \texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}\}$$

so that the above 3-tuples can be encoded in a straightforward way: If a row r and column c are encoded by unpadded decimal representations then the string encoding the 3-tuple (r, c, σ) will have length in $O(\log r + \log c)$.

Following an ***initialization*** phase, that sets up the contents of the tapes described above, each move M will be ***simulated***, one step at a time.

Initialization: Suppose that the input for M (and $\widehat{M}$) is a string

$$\omega = \sigma_1 \sigma_2 \dots \sigma_n$$

with length $n \geq 0$ in $\Sigma^\star$.

- For $1 \leq i \leq k$, "1" should be written onto tapes $3i-2$ and $3i-1$, in order to record the fact that each of M's tape heads is initially positioned at row 1 and column 1.
- For $2 \leq i \leq k$, tape $3i$ should remain empty (that is, filled with blanks) in order to record the fact that the i^{th} tape of M is initially empty as well.
- If $n = 0$ then tape 3 should remain empty as well. Otherwise the encoding of the sequence of 3-tuples

  $$(1, 1, \sigma_1), (1, 2, \sigma_2), \dots, (1, i, \sigma_i), \dots, (1, n, \sigma_n)$$

  should be written onto this tape.

Suppose that the scratch tape is used to maintain a decimal representation of the integer i for $i = 1, 2, \dots, n$ when the encoding of the 3-tuple $(1, i, \sigma_i)$ is being written to the tape. The value of a binary counter can be increased by one (even using a one-tape Turing machine) using time that is linear in the length of the counter — and this can be used to argue that tape 3 can be initialized using a number of steps that is linear in the length of the string that is written onto it. Since each integer i such that $1 \leq i \leq n$ has a decimal representation that is $O(\log n)$, and (at least) one-half of them have decimal representations with length $\Omega(\log n)$ as well, the length of the above string is in $\Theta(n \log n)$.

Thus the cost of this initialization process is in $\Theta(n \log n)$.

Simulation of a Step: Comparing with the information on tapes $3i-2$ and $3i-1$, as needed, the contents of tape $3i$ should be consulted, for $1 \leq i \leq k$. Since comparing decimal representations of numbers is a linear-time process (if done properly) the time needed to discover the symbol that is visible on each tape will be linear in the representation of the corresponding tape of M, that is, linear in the length of the non-blank portion of $\widehat{M}$'s tape $3i$ for the appropriate integer i.

Since k is a constant the information that must be remembered about this (as i increases) can be remembered in $\widehat{M}$'s finite control. The time needed to complete this step of the simulation

— by changing state and updating the contents of all of $\widehat{M}$'s tapes — is also linear in the sum, for $1 \le i \le k$, of the length of $\widehat{M}$'s tape $3i$, for $1 \le i \le k$.

It therefore remains only to bound the length of the sum of the lengths of $\widehat{M}$'s tape $3i$ for $1 \le i \le k$. Since k is a constant this is linear in the maximum of the lengths of any one of these tapes.

Tape 3 stores information about M's input tape, which initially has information about n non-blank symbols on it. If $2 \le i \le k$ then tape $3i$ initially stores information about zero non-blank symbols on M's i^{th} tape. If $1 \le t \le T(n)$ then, after the first t steps, the row and column of each tape head that has been visited is between 1 and $t+1$. Furthermore, at most t non-blank locations have been added to each tape that must be recalled.

It follows that, after t steps, the length of $\widehat{M}$'s tape 3 has length in $O((t+n)\log(t+n))$, while, for $2 \le i \le k$, the lengths of $\widehat{M}$'s tape $3i$ has length in $O(t \log t)$. Now, since $T(n) \ge n$, $\log(t+n) \in O(\log T(n))$ for every integer t such that $1 \le t \le T(n)$. The total cost to access tape 3 of $\widehat{M}$'s tape, during this simulation, is therefore linear in

$$\begin{aligned}
\sum_{t=1}^{T(n)} (t+n)\log T(n) &\le \log T(n) \left(\sum_{t=1}^{T(n)} t + n \sum_{t=1}^{T(n)} 1 \right) \\
&= \left(\frac{T(n)(T(n)+1)}{2} + nT(n) \right) \log T(n) \\
&\in O(T(n)^2 \log T(n)).
\end{aligned}$$

Similarly, the total cost to access tape $3i$ (for $2 \le i \le k$) is in $O(T(n)^2 \log T(n))$ as well.

Since $T(n) \ge n$, the total cost of this simulation is in $O(T(n)^2 \log T(n))$, as claimed.

Alternative Approaches:

- One can also use ***Gödel numbering*** to map the contents of a two-dimensional grid — with cells at row i and columns j for positive integers i and j — to a linear tape: The function $f : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that

  $$f(i,j) = \frac{(i+j)(i+j+1)}{2} + 1 - j$$

  for all positive integers i and j is a bijection — so that the symbol at row i and column j can the stored at location $f(i,j)$ of a one-dimensional tape.

  The ***analysis*** of a simulation using this approach is complicated by the fact that $f(i,j) \in \Theta((i+j)^2)$, so that the "nonblank portion" of the one-dimensional tape (including blanks in-between non-blank symbols) can be quadratic in $n+t$ after the first t steps.

One must notice, in order complete an analysis establishing the desired time bound, is that

$$|f(i+1,j)-f(i,j)| \in O(i+j) \quad \text{and} \quad |f(i,j+1)-f(i,j)| \in O(i+j)$$

so that the distance one must move, on the one-dimensional tape, when simulating the t^{th} step, is at most ***linear*** in t.

Indeed, a careful and clever implementation and analysis using this approach can establish that $O(T(n)^2)$ steps are sufficient to simulate the Turing machine with two-dimensional tapes. Thus an extra "log factor" can be eliminated.

- Another approach is to use a separator, #, that is not in the tape alphabet of the Turing machine that is being simulated, and to use a linear tape to store the contents of a two-dimensional tape by including

  $$\#r_1\#r_2\#\dots$$

  where r_i is the non-blank portion of row i —- beginning at column 1 and ending at the rightmost column where a non-blank symbol appears at this row.

  Suppose, though, that the Turing machine does the following for a positive integer n — possibly the length of the machine's input:

  1. Use n steps to move right (possibly using a second tape to keep track of the number of moves that have been made). writing non-blank symbols along the way; r_1 now has length n.
  2. Use n steps to move up (referring to the second tape as needed), writing non-blank symbols, once again; r_i now has length $n+1$ as well, for $1 \leq i \leq n-1$ — and the nonblank portion of the tape used for the simulation now has length in $\Theta(n^2)$.
  3. Use n steps to move back down, without erasing the non-blank symbols (and writing a nonblank symbol during the first of these moves), so that the tape head is now at row 1 and column $i+1$ — and r_n now has length $n+1$ as well.
  4. Finally, use another n steps to move right, writing non-blank symbols as you go.

  Since the part of the tape storing the contents of rows 2–n must be shifted over, during each one of the last n moves, $\Omega(n^2)$ steps will be used for each of these moves — and the number of steps used by the simulating Turing machine will be in $\Omega(n^3)$, even though the two-dimension machine being simulated used $O(n)$ steps.

  Thus this simulation is too inefficient the establish the desired result.

  It is certainly possible to vary the organization of the tape in a variety of ways. However, I am not aware of any variation for which a sub-cubic worst-case time bound can be proved.