Homework 3
March 9, 2021
TangLin

PROBLEM 1. Prove that there is a universal single-tape Turing Machine whose time complexity increases linear time for all the single-tape Turing Machine. More specifically, there is a single-tape Turing Machine U, for any other single-tape TM M there is a number c and string p: $U(p\#x) = M(x)$ and $t_U(p\#x) \leqslant c(t_M(x) + |x|)$ for any x.

Solution:

This solution is similar to the theorem which you told us about simulating single-tape TM by double-tape TM. I make a slight modification and get a single-tape universal TM. I separate the unique tape to 4 areas including Program Area, Status Area, Input Area and Output Area.

Λ	Program	#	Status	#	Input	#	Output	Λ

Figure 1: Tape Construction of S-UTM.

Now I explain how are these areas constituted.

1. Program Area.
   This Area is used to store the TM M which we want to simulate, it means that there will store the Transition Table. We have to encode the Machine M before storing. The most convenient way is encoding it by binary codes. Assume that
   $$|Q_M| = l, |\Sigma_M| = m$$
   Every transition has form
   $$pa \mapsto p'a'\Delta$$
   Where $p, p' \in Q_M, a, a' \in \Sigma_M, \Delta \in \{-1, 0, +1\}$. So every transition can be encoded to binary codes with $2(\log_2 l + 1 + \log_2 m + 1) + 1$ bits. And there are at most 2^{l+m} different transitions. So we will cost at most
   $$|P| = 2^{2(\log_2 l + \log_2 m + 2)+1} \cdot 2^{l+m} = 32l^2m^2 2^{l+m}$$
   bits to encode M.

2. Status Area.
   This area is used to store all the statuses of M and indicates which is the current status. Every status need at most $\log_2 l + 1$ bits, so all the status will cost at most
   $$l \cdot (\log_2 l + 1)$$
   bits.

3. Input and Output Area.

   There will store the input and output of M, and it will cost at most

$$(|x| + |y|) \cdot m$$

   bits. Where x is the input and y is the output.

Now I analysis the comprehensive complexity of UTM.

First, we have to "write" the Machine M and its statuses and inputs, it will cost

$$|P| + l \cdot \log_2 l + l + (|x| + |y|) \cdot m$$

steps. And then every time M calculate, the UTM have to access the transition table and modify the current status (must go and back); so it will cost $2 \cdot (|P| + l \cdot \log_2 l + l)$ steps. And for every input x, M have cost $t_M(x)$ steps until it halts. So the UTM have to cost

$$\begin{aligned} t_U(p\#x) &= t_M(x) \cdot 2 \cdot (|P| + l \cdot \log_2 l + l) + |P| + l \cdot \log_2 l + l + (|x| + |y|) \cdot m \\ &\leqslant t_M(x) \cdot 3|P| + 3|P| \cdot |x| \\ &= 3|P|(t_M(x) + |x|) \end{aligned}$$

steps. So for any specific TM M, there is a constant $c = 3|P|$, and

$$t_U(x) \leqslant c(t_M(x) + |x|).$$

Problem 2. Prove that there is a function with two values which can be computed by single-tape Turing Machine in time $O(n^{100})$, but can not be computed by single-tape Turing Machine in time $O(n^{10})$.

Solution:

My idea is from the proof of the Time Hierarchy Theorem. I will find a function by constructing a special set.

First, assume that there is an appropriate numbering for all Turing Machine, for example, according to their binary code and add 1 to the left side, and let the corresponding natural number as the number of Turing Machine. Let set A be

$$A = \{1^n | \text{Turing Machine } M_n \text{ halts and return 0 in } n^{10} \text{ steps.}\}$$

According to the time hierarchy theorem, there exists a multi-tapes Turing Machine can accept set A in $O(n^{10} \log n)$ time. Further, we can build a single-tape Turing Machine which can accept it in $O((n^{10} \log n)^2)$ time. Obviously $O(n^{20} \log^2 n) \subset O(n^{100})$, so set A can be identified in time $O(n^{100})$ but not in time $O(n^{10})$.

Now I build the function by set A

$$f(n) = \begin{cases} 1, & \text{if } 1^n \in A. \\ 0, & \text{otherwise.} \end{cases}$$

function $f(n)$ can be compute in time $O(n^{100})$ but not in time $O(n^{10})$ by single-tape Turing Machine.