Computational Complexity **Autumn, 2018**

Solution : Homework 1

Lecturer: Yuxi Fu *Homework taker: Xu Li 018033210002*

Problem 1.5.

Solution: Assume there is a k-tape TM M accepts L in time $T(n)$, we can construct a 2-tape TM M' as follows

- The i^{th} location of the j^{th} tape in M corresponds to the $(i + jk)^{th}$ location of the work tape in M'.
- For every symbol a in Γ_M, there is two symbols a and $\hat{a}$ in $\Gamma_{M'}$. The symbol with $\hat{}$ indicates the the corresponding head of M is positioned in that location.
- Each step in M corresponds to two sweeps in M': first, it sweeps the tape in the left-to-right direction and records to its register the k symbols that are marked by $\hat{}$. Then M' uses the transition function of M to determine the new state, symbols, and head movements and sweeps the tape back in the right-to-left direction to update the encoding accordingly.

The head of M' only depend on n (the input length). Since the length of each tape in M will not exceed $T(n)$, each step of M' tasks $O(T(n))$ time. Since M accepts L in time $T(n)$, M' accepts L in time $O(T(n)^2)$
■

Problem 1.6.

Solution: Given a k-tape TM M accepts L in time $T(n)$, we can construct a oblivious UTM U as follows

- Create 5 tapes with 1 input tape (same as the input tape in M), 1 work tape to simulate all work tapes in M, 1 work tape to record the description of M (transition functions), 1 work tape to record the current state of M and 1 output tape.
- Symbols in the same position of k tapes in M are encoded into one symbol in U.
- The work tape for simulation in U is split into zones, where the range of zone R_i is $[2^{i+1} - 1, 2^{i+2} - 2]$, L_i is $[-2^{i+2} + 2, -2^{i+1} + 1]$. The special symbol $\boxdot$ is used for buffer cells. A zone is empty if all of its cells are marked with $\boxdot$; half full if half of its cells are marked with $\boxdot$; full if none of its cells are marked with $\boxdot$.
- $\forall i \in \{0, ..., \log(T)\}$: L_i is full $\Leftrightarrow$ R_i is empty; L_i is half full $\Leftrightarrow$ R_i is half full; L_i is empty $\Leftrightarrow$ R_i is full. And location 0 is always contains a non-$\boxdot$ symbol.
- For each step in M, if the head is moved to the right, then move the corresponding tape to left, and the verse visa.

■ Once a shift with index i is performed, the next $2i - 1$ shifts of that parallel tape will all have index less than i. Therefore there are at most $kT/2^i$ shifts with index i. And the total number of shifts equals to

$$\sharp(shift) = O\left(k \sum_{i=1}^{\log(T)} \frac{T}{2^i} 2^i\right) = O(T(\log(T)))$$

Problem 1.9.

Solution: Assume a RAM TM R computes a Boolean function f in time $T(n)$, there are at most $T(n)$ time read and write operations. We construct a TM M as follows

- We use work tape t_1 to simulate array A, t_2 and t_2 to recored the relationship between address and the position of that address in t_1.
- The i^{th} address appears in R corresponding to the i^{th} position in t_1.
- For read or write step in R, M first find the corresponding position using t_2 and then move the head of t_1 to the right position and do the same thing as R.

Since there are at most $T(n)$ addresses, the length of t_1 is at most $T(n)$. Therefore it tasks $O(T(n))$ time to simulate read/write operation. So M computes Boolean function f in time $T(n)^2$, i.e., if a Boolean function f is computable within time $T(n)$ (for some time constructible T) by a RAM TM, then it is in DTIME($T(n)^2$). ■

Problem 1.13.

Solution:

(a) $\mathrm{BIT}(n,i) = \forall_{(i+C)^3<x<(i+C+1)^3} : \mathrm{PRIME}(x) \wedge (\forall_{(i+C)^3<y<(i+C+1)^3} : \mathrm{PRIME}(y) \wedge x \leq y) \wedge \mathrm{DIVIDE}(x,n)$

(b) $\mathrm{COMPARE}(n,m,i,j) = (\mathrm{BIT}(m,i) \wedge \mathrm{BIT}(n,j)) \vee (\neg\mathrm{BIT}(m,i) \wedge \neg\mathrm{BIT}(n,j)$

(c) The configuration can be encoded using form

$$\text{input} \ddagger \text{head} \ddagger \text{state}\ddagger$$

where $\ddagger \notin \Gamma$. To add the new symbols, we map $0, 1, \ddagger$ as follows

$$0 \longmapsto 00, 1 \longmapsto 01, \ddagger \longmapsto 11$$

We use 0 to denote the initial state q_{start} and use 1 to denote the halt state q_{halt}. We let $head = 1^n$ if head is on th n^{th} position and $state = 1^n$ is $q = n$.

(d) We let D1(p,n)/D2(p,n)/D3(p,n) to be true if the p^{th} and the $p+1^{th}$ symbol of the string encoded by number n is the first/second/third †.

$$\mathrm{D1}(p,n) = \mathrm{BIT}(p,n) \wedge \mathrm{BIT}(p+1,n) \wedge (\forall i < p/2 : \neg(\mathrm{BIT}(2i-1,n) \wedge \mathrm{BIT}(2i,n)))$$

$\mathrm{D2}(p,n) = \mathrm{BIT}(p,n) \wedge \mathrm{BIT}(p+1,n) \wedge (\exists d1, \forall d1/2 < i < p/2 : \mathrm{D1}(d1,n) \wedge \neg(\mathrm{BIT}(2i-1,n) \wedge \mathrm{BIT}(2i,n)))$

$\mathrm{D3}(p,n) = \mathrm{BIT}(p,n) \wedge \mathrm{BIT}(p+1,n) \wedge (\exists d2, \forall d2/2 < i < p/2 : \mathrm{D2}(d1,n) \wedge \neg(\mathrm{BIT}(2i-1,n) \wedge \mathrm{BIT}(2i,n)))$

We let HEAD(h,n) to be true if h is the head position of the configuration encoded by n:

$\mathrm{HEAD}(h,n) = \exists d1, d2 : \mathrm{D1}(d1,n) \wedge \mathrm{D2}(d2,n) \wedge (\forall d1/2+1 < i < d2/2 : \neg\mathrm{BIT}(2i-1,n) \wedge \mathrm{BIT}(2i,n)) \wedge (h = d2/2 - d1/2 - 1)$

We let STATE(s,n) to be true if s is the state of the configuration encoded by n:

$\mathrm{STATE}(h,n) = \exists d2, d3 : \mathrm{D2}(d2,n) \wedge \mathrm{D3}(d3,n) \wedge (\forall d2/2+1 < i < d3/2 : \neg\mathrm{BIT}(2i-1,n) \wedge \mathrm{BIT}(2i,n)) \wedge (h = d3/2 - d2/2 - 1)$ $\mathrm{INIT}_{M,x}(n) = \forall i \leq |x| : \neg\mathrm{BIT}(2i-1,n) \wedge (\mathrm{BIT}(2i,n) \wedge x[i] \vee \neg\mathrm{BIT}(2i,n) \wedge \neg x[i]) \wedge \mathrm{HEAD}(0,n) \wedge \mathrm{STATE}(0,n)$

(e) $\mathrm{HALT}_M(n) = \mathrm{STATE}(1,n)$

(f) NEXT$(n,m) = \exists d1, h1, h2, q1, q2 : (\forall i < d1 : \text{COMPARE}(n,m,i,i) \vee i = h1) \wedge \text{HEAD}(h1,n) \wedge \text{HEAD}(h2,m) \wedge \text{STATE}(s1,n) \wedge \text{STATE}(s2,m) \wedge < q1, BIT(2*h1,n) > \rightarrow < q2, BIT(2*h1,m), h2-h1 > \in \delta$

(g) $\text{VALID}_M(m,t) = \forall i < t-1 : \text{NEXT}(x_i, x_{i+1})$

(h) $\text{HALT}_{M,x}(t) = \exists m : \text{HALT}_M(x_t) \wedge \text{INIT}_{M,x}(x_1) \wedge \text{VALID}_M(m,t)$

(i) The halting problem can be defined by

$$\exists t : \text{HALT}_{M,x}(t)$$

. If TRUE-EXP is computable, then halting problem is also computable.

■

Problem 1.15.

Solution:

(a) Given two arbitrary number $b, b' > 1$, assume a TM M accept L_S^b in time $T(n^c)$. We can create a new TM M'. Given an input $L_S^{b'}$, M' first transform $L_S^{b'}$ to L_S^b and then do the same thing as TM M. Since the transformation takes $O(n)$ time, M' works in $O(T(n^c))$ and thus $L_S^{b'} \in$P.

(b) The input size is $n + l + k$. It tasks $O(j)$ time to judge whether j is prime and it tasks $O(n)$ time to judge where j dividing n. Therefore it takes $O((k-l)(j+n) < O((n+l+k)^2)$ to accept language UNARYFACTORING, i.e., UNARYFACTORING $\in$P.

■