CMU 15-855* Fall 2017
Graduate Computational Complexity Theory
Homework 2

Zhikun Wang*

February 11, 2021

For problem statements and course notes, please see this 120MB pdf file published by Ryan O'Donnell.

This homework is not scored or reviewed by a professor or a TA. If you believe you've found a mistake then please never hesitate to email me or comment on my blog! Thanks!

I discussed with Yuzhou Gu about problem 1.4 as it is allowed by the homework instructions.

1 Almost-Everywhere Time Hierarchy Theorems.

1.1

We prove this by contradiction. Say that if the statement does not stand, then for any language $L \in \mathsf{TIME}(T(n))$ there is M with running time $O(t(n))$, which only differs from L on finitely many inputs. In this case we can construct M' as follows:

1. Test if the input is in a hardcoded finite set $S = \{x | M(x) \neq L(x)\}$. If so, output hardcoded $L(x)$.
2. Otherwise, simulate M on x.

This TM runs in time $O(t(n))$. Thus, $L \in \mathsf{TIME}(t(n))$ which contradicts with the time hierarchy theorem.

1.2

We prove this by contradiction. Say that if the statement does not stand, then for any language $L \in \mathsf{TIME}(T(n))$ there is M running for less than $Ct(n)$ steps except on finitely many inputs. In this case we can construct M' as follows:

1. Test if the input is in a hardcoded finite set $S = \{x | M(x) \text{ takes more than } Ct(|x|)\text{steps}\}$. If so, output hardcoded $L(x)$.
2. Otherwise, simulate M on x.

This TM runs in time $O(t(n))$. Thus, $L \in \mathsf{TIME}(t(n))$ which contradicts with the time hierarchy theorem.

1.3

Consider two TMs: $M_1(x) = 0$ and $M_2(x) = 1$. Then at least one of them differs infinitely many from any language L.

*nocrizwang@gmail.com

1.4

We use a language L defined by the following TM M:

1. Check if x is a valid TM representation. If x is not a valid TM representation, halt and reject.
2. Simulate the TM represented by x on input x for $2^|x|$ steps. If x does not halt, halt and reject.
3. If x accepts, reject.
4. If x rejects, accept.

Then for any polynomial-time Turing Machine M', M' will differ with M for all inputs sufficiently long as M' runs in sub-exponential time and that M' has a representation for all lengths sufficiently long (say that we allow "comments" in representations of Turing Machines).

2 Superiority.

2.1

The machine M_1 runs as follows on input x:

1. Check if x is a valid TM representation. If x is not a valid TM representation, halt and reject.
2. Simulate the TM represented by x on input x for $|x|^1.5$ steps. If x does not halt, halt and reject.
3. If x accepts, reject.
4. If x rejects, accept.

Then for every machine M_2 running in $O(n)$ time and every large enough n we have a length n representation of M_2 where the output of M_2 differs from the output of M_1.

2.2

The proof of Nondeterministic Hierarchy Theorem does not prove $\mathsf{NTIME}(n^{1.1})$ superior to $\mathsf{NTIME}(n)$ because the proof uses lazy diagonalization which uses exponential simulation to diagonalize. Thus there may not be a input that the output of M_1 and M_2 differs with length $[n, n^2]$ as the diagonalization happens at point $f(i)$ where $f(i) = 2^{f(i-1)^{1.2}}$.

3 Awesome circuit lower bounds from depth-3 circuit lower bounds.

3.1

From the "depth reduction lemma" proved in the first homework, it is possible to remove at most $(r/k)m$ edges to reduce the depth to 2^{-r} of the original depth. Thus we can remove at most $(100c_1/\log(c_1 \log n)))c_2 n = O(n/\log\log n)$ edges and make the depth of each subcircuit at most $0.01 \log n$. As each gate of the circuit can take in at most 2 inputs, each subcircuit depends on at most $2^{0.01 \log n} = O(n^{0.01})$ inputs.

3.2

First we add an NOT gate to every circuit input. After adding this gate we can will be able to access both the original value and the negated value of the input.

In our construction for simplicity we consider the final output of the original circuit as a cut wire.

On the first layer we construct OR gates. For every subcircuit, we enumerate over all $2^{O(n^{0.01})}$ assignments of inputs that is not a cut wire. For an assignment h, we add an OR gate consisting of all negated values in h. Thus the gate will evaluate 0 only if the inputs has the same value as h, otherwise the gate will evaluate 1.

On the second layer we use AND gates. For every subcircuit and every assignment g of the inputs which is a cut wire we construct two AND gates. The first AND gate's inputs are all OR gates of the same subcircuit which corresponds to the assignment h such that when combined with g, the subcircuit evaluates to 1. The second AND gate has all other OR gates of the same subcircuit as input. Then, the first AND gate will evaluate 1 if and only if the subcircuit with given inputs evaluates to 0 (i.e. not evaluated to 1), and the second and gate will evaluate 1 if and only if and only if the subcircuit with given inputs evaluates to 1.

On the third layer we use AND gates as well. For all assignment of cut wires such that the cut wire for the circuit output is 1, we construct an AND gate with the AND gates on the second layer which assume the same value of cut wires and has the same output of cut wire as the input.

On the fourth layer we use an OR gate with all gates from the third layer as input.

Finally, as the third layer and the second layer are both AND gates, it is possible to collapse the AND gates to one layer. Thus the depth of our circuit will be 3.