Theory of Computation: Time Hierarchy

Efficiency of UTM

- So far, if we had to simulate a deterministic TM on an input as part of a subroutine of an algorithm, we used a Universal Turing Machine (UTM) for it.
- If we are looking at efficiency of algorithms, the running time of the UTM is also important – it adds to the total running time of the algorithm.
- Theorem: There is a UTM that for every $M\#x$, where the running time of M is denoted by function $T : \mathbb{N} \to \mathbb{N}$, writes down $M(x)$ on its tape in the end in $CT(|x|)\log(T(|x|))$ time. C is a constant that only depends on the alphabet size, number of tapes and number of states of M.

Relaxed version

To give an idea of the Proof, we give a proof for a relaxed version where the UTM $\mathcal{U}$ runs in $T(n)^2$ time if $M(x)$ is computed in $T(n)$ time:

- The input to $\mathcal{U}$ is an encoding of TM M and the input x.
- Transformation of M: Single work tape
  M only has alphabets $\{\vdash, B, 0, 1\}$ - encoding of larger alphabets using $\{0, 1\}$
  These transformations may make M run in T^2 time instead of T on a given input.
- The UTM $\mathcal{U}$ has alphabets $\{\vdash, B, 0, 1\}$ and 3 work tapes.
  One work tape is used in the same way as M (also the input and output tapes)
  One tape is used to store M's transition function
  One tape stores M's current state.

Relaxed version contd.

- One computational step: $\mathcal{U}$ scans the M's transition function and current state to find out the new state, symbols to be written and tape head movements. Then it executes this. This is done in time C - only dependent on size of the transition function.
- Total time for outputting $M(x)$ on the output tape of $\mathcal{U}$: $CT(|x|)^2$.
- For $CT(n)\log(T(n))$ running time, we need to design the UTM more carefully.

Efficiency of NUTM

- Nondeterministic UTMs can also be designed: An NDTM taking in encodings of NDTMs to be simulated as subroutines.
- Theorem: There is a NUTM that for every $M\#x$, where the running time of M is denoted by function $T : \mathbb{N} \to \mathbb{N}$, writes down $M(x)$ on its tape in the end in $CT(|x|)$ time.
  C is a constant that only depends on the alphabet size, number of tapes and number of states of M.

Time constructible functions

- Time constructible function: A function $T : \mathbb{N} \rightarrow \mathbb{N}$ such that $T(n) \geq n$ and there is a deterministic TM M that computes the function $f : \mathbb{N} \rightarrow \{0,1\}^*$ with $f(x) = \text{bin}(T(x))$.
- Examples: $n, n \log n, n^2, 2^n$.
- All functions we see in this course are time constructible. Especially when we are looking at functions that act as time bounds for Turing machines.
- $T(n) \geq n$ implies that an algorithm running in time $T(n)$ has time to read the input.

Time Hierarchy Theorem

Theorem: If f, g are time constructible functions satisfying $f(n) \log f(n) = o(g(n))$, then $DTIME(f(n)) \subsetneq DTIME(g(n))$

- Proof uses a form of diagonalization.
- We will show that $DTIME(n) \subsetneq DTIME(n^{1.5})$ and all other pairs of functions will have similar proofs.
- Diagonalization TM M: On input x, run UTM $\mathcal{U}$ for $|x|^{1.4}$ steps to simulate the execution of M_x on x.
  If $\mathcal{U}$ outputs bit $b \in \{0, 1\}$ then output $1 - b$. Else, output 0.
- M halts in $n^{1.4}$ steps and language $L = L(M)$ is in $DTIME(n^{1.5})$.

Time Hierarchy Theorem

- $L \notin DTIME(n)$: Suppose there is some TM N and constant c such that N on any input x halts within $c|x|$ steps and outputs $M(x)$.
  $N\#x$ can be simulated in $\mathcal{U}$ in time $c'c|x|\log|x|$, where c' only depends on description of N.
  There is an n_0 such that $\forall n \geq n_0$, $n^{1.4} > c'c|x|\log|x|$.
  Let x be a string representing N such that $|x| \geq n_0$ (infinitely many strings represent N)
  M will obtain output $b = N(x)$ in $|x|^{1.4}$ steps, but by definition $M(x) = 1 - b \neq N(x)$ ($\rightarrow\leftarrow$).

Nondeterministic Time Hierarchy Theorem

Theorem: if f, g are time constructible functions satisfying $f(n+1) = o(g(n))$, then $NTIME(f(n)) \subsetneq NTIME(g(n))$

- Use of NUTM here.
- In Time Hierarchy Theorem, we crucially use the fact that a DTM can compute the opposite answer: If it is running a subroutine M, then on computing $M(x)$ it can flip the answer.
- In case of an NTM, that is not clear. Because these machines verify, they do not compute.
  If some branches compute "accept" and others compute "reject", then what would be a flipped answer?
  If allowed exponential time, then they can compute all possible certificates and solve the problem, but within an increase of time bound by a polynomial factor, it may not be possible.

Lazy Diagonalisation

Lazy diagonalization: Here, the machine executing the diagonalization will not try to flip the answer of a subroutine TM on every input, but on a crucial input. This will be enough to get the contradiction we are aiming for using diagonalization.

Nondeterministic Time Hierarchy Theorem

- Just show $NTIME(n) \subsetneq NTIME(n^{1.5})$. All other pairs will have similar arguments.
- Define $h : \mathbb{N} \rightarrow \mathbb{N}$ such that $h(1) = 2, h(i+1) = 2^{h(i)^{1.2}}$.
- Given n, find in $n^{1.5}$ time i such that $h(i) < n \leq h(i+1)$.
- Diagonalisation machine M: try to flip answer of M_i on some input in set $\{1^n | h(i) < n \leq h(i+1)\}$.
- Machine M: On input x, if $x \notin 1^*$ then reject.
  If $x = 1^n$, then compute i such that $h(i) < n \leq h(i+1)$.
  1. If $h(i) < n < h(i+1)$, then simulate M_i on 1^{n+1} using nondeterminism in $n^{1.1}$ time and output the answer. (If M_i does not halt in this time , then halt and accept.)
  2. If $n = h(i+1)$, accept 1^n iff M_i rejects $1^{h(i)+1}$ in $(h(i)+1)^{1.1}$ time.

Nondeterministic Time Hierarchy Theorem

- Point 2: All possible $2^{(h(i)+1)^{1.1}}$ branches of M_i on input $1^{h(i)+1}$ have to be computed. - input size is $h(i+1) = 2^{h(i)^{1.2}}$.
- M runs in $O(n^{1.5})$ time.
- $L = L(M)$.

Nondeterministic Time Hierarchy Theorem

- Claim: $L \notin NTIME(n)$.
- Suppose there is an NDTM N running in cn steps for L.
- Pick an i large enough such that $N = M_i$ and on inputs of length $n \geq h(i)$, M_i can be simulated in less than $n^{1.1}$ steps.
- Target: Try to flip the answer of N with M on an input in $\{1^n | h(i) < n \leq h(i+1)\}$.

Nondeterministic Time Hierarchy Theorem

- Description of M ensures: If $h(i) < n < h(i+1)$, then $M(1^n) = M_i(1^{n+1})$ (which is same as $M(1^{n+1})$)
  Otherwise, $M(1^{h(i+1)}) \neq M_i(1^{h(i)+1})$.
- M_i and M agree on all inputs 1^n for $n \geq h(i)$, and in particular in the interval $(h(i), h(i+1)]$
  By definition: $M(1^{h(i)+1}) = M_i(1^{h(i)+2}) = M(1^{h(i)+2})$
  $= M_i(1^{h(i)+3}) = M(1^{h(i)+3}) \ldots$
  $= M_i(1^{h(i+1)}) = M(1^{h(i+1})$ $(\rightarrow\leftarrow)$.
- Thus, there is a string in $\{1^n | h(i) < n \leq h(i+1)\}$ on which M and M_i do not agree.