CS 483 - Introduction to the Theory of Computation

Lecture 19

Complexity of Languages

How do we measure the complexity of a language $L \subseteq \Sigma^*$?

Option 1: $\min \{ |\langle M \rangle| \mid M \text{ decides } L \}$ (like in Kolmogorov complexity)

Problems:

- Some easy-to-decide languages have high complexity, e.g.

$$L = \{11001101000010101100\}$$

- Some hard-to-decide languages have small complexity.

Complexity of Languages

How do we measure the complexity of a language $L \subseteq \Sigma^*$?

Option 2: minimum number of steps that M takes to decide L.

Time Complexity

For a function $T: \mathbb{N} \to \mathbb{N}$, we say a Turing machine M has (running) time $T(n)$, if for every $w \in \Sigma^*$, $M(w)$ halts in at most $T(|w|)$ steps.

A language L has time complexity $T(n)$, if L can be decided by a Turing machine with running time $T(n)$.

Time Complexity

For a function $T: \mathbb{N} \to \mathbb{N}$, we say a Turing machine M has (running) time $T(n)$, if for every $w \in \Sigma^*$, $M(w)$ halts in at most $T(|w|)$ steps.

A language L has time complexity $T(n)$, if L can be decided by a Turing machine with running time $T(n)$.

Let **TIME**$(T(n))$ be the class of languages with time complexity $O(T(n))$.

Time Complexity

For a function $T: \mathbb{N} \to \mathbb{N}$, we say a Turing machine M has (running) time $T(n)$, if for every $w \in \Sigma^*$, $M(w)$ halts in at most $T(|w|)$ steps.

A language L has time complexity $T(n)$, if L can be decided by a Turing machine with running time $T(n)$.

Let **TIME**$(T(n))$ be the class of languages with time complexity $O(T(n))$.

We define $\mathbf{P} = \mathrm{U}_{k>0}\, \mathbf{TIME}(n^k)$

(i.e. languages decided by polynomial-time Turing machines)

Asymptotic Notations

For $f, g: \mathbb{N} \to \mathbb{N}$, we write:

- $f(n) \in O(g(n))$, if there exists $c > 0$, $n_0 \in \mathbb{N}$ such that

$$f(n) \leq c \cdot g(n), \forall n \geq n_0;$$

- $f(n) \in \Omega(g(n))$, if there exists $c > 0$, $n_0 \in \mathbb{N}$ such that

$$f(n) \geq c \cdot g(n), \forall n \geq n_0;$$

Asymptotic Notations

For $f, g: \mathbb{N} \to \mathbb{N}$, we write:

- $f(n) \in O(g(n))$, if there exists $c > 0$, $n_0 \in \mathbb{N}$ such that

$$f(n) \leq c \cdot g(n), \forall n \geq n_0;$$

- $f(n) \in \Omega(g(n))$, if there exists $c > 0$, $n_0 \in \mathbb{N}$ such that

$$f(n) \geq c \cdot g(n), \forall n \geq n_0;$$

- $f(n) \in \Theta(g(n))$, if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$;
- $f(n) \in o(g(n))$, if $\lim_{n \to \infty} f(n)/g(n) = 0$;
- $f(n) \in \omega(g(n))$, if $\lim_{n \to \infty} f(n)/g(n) = \infty$.

Nuance of Time Complexity

What type of Turing machine is used when we define time complexity?

- Single-tape or multi-tape?
- One-way infinite tape or two-way infinite?
- Tape alphabet $\Gamma = \Sigma \sqcup \{\textvisiblespace\}$ or no restriction?
- ...

Nuance of Time Complexity

The number of tapes matters:

$$\textsc{Palindrome} = \{ w \in \{0,1\}^* \mid w = w^R \}$$

has time complexity

- $O(n)$ with 2-tape Turing machines;
- $\Omega(n^2)$ with single-tape Turing machines.

Nuance of Time Complexity

The number of tapes matters:

$$\textsc{Palindrome} = \{ w \in \{0,1\}^* \mid w = w^R \}$$

has time complexity

- $O(n)$ with 2-tape Turing machines;
- $\Omega(n^2)$ with single-tape Turing machines.

Theorem. Every Turing machine with running time $T(n)$ has an equivalent single-tape Turing machine with running time $O(T^2(n))$.

Nuance of Time Complexity

The number of tapes matters:

$$\textsc{Palindrome} = \{ w \in \{0,1\}^* \mid w = w^R \}$$

has time complexity

- $O(n)$ with 2-tape Turing machines;
- $\Omega(n^2)$ with single-tape Turing machines.

Theorem. Every Turing machine with running time $T(n)$ has an equivalent single-tape Turing machine with running time $O(T^2(n))$.

Hides constant that depends on M

Nuance of Time Complexity

Theorem. There is a universal Turing machine U with a single tape, such that if $M(w)$ halts in T steps, then $U(M, w)$ halts in $c_M T^2$ steps, where c_M only depends on M (but not w).

Nuance of Time Complexity

Theorem. (We will skip the proof) There is a universal Turing machine U with two tapes, such that if $M(w)$ halts in T steps, then $U(M, w)$ halts in $c'_M T \log T$ steps, where c'_M only depends on M (but not w).

Nuance of Time Complexity

Theorem. (We will skip the proof) There is a universal Turing machine U with two tapes, such that if $M(w)$ halts in T steps, then $U(M, w)$ halts in $c'_M T \log T$ steps, where c'_M only depends on M (but not w).

Corollary. Every Turing machine with running time $T(n)$ has an equivalent two-tape Turing machine with running time $O(T(n) \log T(n))$.

Hides constant that depends on M

Open: Improve to $O(T(n))$ for some constant number of tapes.

Nuance of Time Complexity

Theorem. Every Turing machine with time $T(n)$ has an equivalent Turing machine using one-way infinite tapes with time $O(T(n))$.

Theorem. Every Turing machine on with time $T(n)$ has an equivalent Turing machine using tape alphabet $\Gamma = \Sigma \sqcup \{_\}$ with time $O(T(n))$.

Nuance of Time Complexity

Theorem. Every Turing machine with time $T(n)$ has an equivalent Turing machine using one-way infinite tapes with time $O(T(n))$.

Theorem. Every Turing machine on with time $T(n)$ has an equivalent Turing machine using tape alphabet $\Gamma = \Sigma \sqcup \{_\}$ with time $O(T(n))$.

Conclusion: **P** is not affected by the nuance!

The classes **TIME**$(T(n))$ are slightly affected, and we should define them with multi-tape Turing machines.

Extended Church-Turing Thesis

A function is physically efficiently computable if and only if it is computable by a Turing machine in polynomial time.

Unlike Church-Turing thesis, the extended version is heavily challenged:

- Is n^{1000} time really more efficient than n^{10} ?
- Quantum computing, etc.

Problems in **P**

- Every regular language is in **TIME**(n)
- Every context-free language is in **TIME**(n^3)

Problems in **P**

- Every regular language is in **TIME**(n)
- Every context-free language is in **TIME**(n^3)
- $\mathrm{TM} = \{ M \mid M \text{ is a Turing machine} \} \in$ **TIME**(n)
- $\mathrm{ATIME}_{\mathrm{TM}} = \{ (M, w, 1^t) \mid \mathrm{TM}\ M \text{ accepts } w \text{ in } t \text{ steps} \} \in$ **TIME**(n^2)

Problems in **P**

- Every regular language is in **TIME**(n)
- Every context-free language is in **TIME**(n^3)
- $\mathrm{TM} = \{ M \mid M \text{ is a Turing machine} \} \in$ **TIME**(n)
- $\mathrm{ATIME}_{\mathrm{TM}} = \{ (M, w, 1^t) \mid \mathrm{TM}\ M \text{ accepts } w \text{ in } t \text{ steps} \} \in$ **TIME**$(n^2 \log n)$
- $\mathrm{STCON} = \{ (G, s, t) \mid \text{Graph } G \text{ has a path from } s \text{ to } t \} \in$ **TIME**(n)
- $\mathrm{MAXMATCHING} = \{ (G, k) \mid \text{Graph } G \text{ has a } k\text{–macthing} \} \in$ **TIME**$(n^{1.5})$
- $\mathrm{PRIMES} = \{ N \mid N \text{ is a prime number in binary} \} \in$ **TIME**$(n^6 \log^c n)$
- …

Time-Hierarchy Theorem

Are there computable languages that are not efficiently computable?

In other word, is $\mathbf{P} = \mathbf{R}$?

Time-Hierarchy Theorem

Are there computable languages that are not efficiently computable?

In other word, is **P** = **R**?

Theorem. For every "nice" functions $f, g: \mathbb{N} \to \mathbb{N}$ such that

$$f(n) \log f(n) = o(g(n))$$

we have **TIME**$(f(n)) \subsetneq$ **TIME**$(g(n))$.

Time-Hierarchy Theorem

A function $f: \mathbb{N} \to \mathbb{N}$ is time-constructible, if there is a Turing machine M with time $O(f(n))$ such that $M(1^n) = 1^{f(n)}$.

Theorem. For every time-constructible functions $f, g: \mathbb{N} \to \mathbb{N}$ such that

$$f(n) \log f(n) = o(g(n))$$

we have $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n))$.

Time-Hierarchy Theorem

A function $f: \mathbb{N} \to \mathbb{N}$ is time-constructible, if there is a Turing machine M with time $O(f(n))$ such that $M(1^n) = 1^{f(n)}$.

Theorem. For every time-constructible functions $f, g: \mathbb{N} \to \mathbb{N}$ such that

$$f(n) \log f(n) = o(g(n))$$

we have $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n))$.

Corollary. $\mathbf{TIME}(n) \subsetneq \mathbf{TIME}(n \log^2 n) \subsetneq \mathbf{TIME}(n^{1.5}) \subsetneq \mathbf{TIME}(n^2) \subsetneq \cdots \subsetneq \mathbf{P}$

Time-Hierarchy Theorem

A function $f: \mathbb{N} \to \mathbb{N}$ is time-constructible, if there is a Turing machine M with time $O(f(n))$ such that $M(1^n) = 1^{f(n)}$.

Theorem. For every time-constructible functions $f, g: \mathbb{N} \to \mathbb{N}$ such that

$$f(n) \log f(n) = o(g(n))$$

we have $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n))$.

Corollary. $\mathbf{TIME}(n) \subsetneq \mathbf{TIME}(n \log^2 n) \subsetneq \mathbf{TIME}(n^{1.5}) \subsetneq \mathbf{TIME}(n^2) \subsetneq \cdots \subsetneq \mathbf{P}$
$\subsetneq \mathbf{TIME}(2^{\sqrt{n}}) \subsetneq \mathbf{TIME}(2^n) \subsetneq \cdots \subsetneq \mathbf{E} \subsetneq \mathbf{TIME}(2^{n^2}) \subsetneq \cdots \subsetneq \mathbf{EXP} \subsetneq \cdots \subsetneq \mathbf{R}$

$\mathbf{E} = \bigcup_{k>0} \mathbf{TIME}(2^{kn})$

$\mathbf{EXP} = \bigcup_{k>0} \mathbf{TIME}(2^{n^k})$

Time-Hierarchy Theorem

If we can list all Turing machines M with running time $f(n)$, flipping the diagonal should take $O(f(n)\log f(n))$ time by running $U(M_i, M_i)$.

List of all deciders that run in time $f(n)$

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	0	1	1	1	0	
M_1	1	1	0	1	1	
M_2	0	0	1	0	1	
M_3	0	1	1	0	0	
M_4	1	0	0	1	0	
$\vdots$						$\ddots$
L	1	0	0	1	1	$\cdots$

Time-Hierarchy Theorem

If we can list all Turing machines M with running time $f(n)$, flipping the diagonal should take $O(f(n)\log f(n))$ time by running $U(M_i, M_i)$.

List (?) of all deciders that run in time $f(n)$

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_0	0	1	1	1	0	
M_1	1	1	0	1	1	
M_2	0	0	1	0	1	
M_3	0	1	1	0	0	
M_4	1	0	0	1	0	
$\vdots$						$\ddots$
L	1	0	0	1	1	$\cdots$

Time-Hierarchy Theorem

Take $L = \{ M \mid U(M, M) \text{ rejects in } g(|M|) \text{ steps} \}$. Then $L \in \textbf{TIME}(g(n))$.

Time-Hierarchy Theorem

Take $L = \{ M \mid U(M, M) \text{ rejects in } g(|M|) \text{ steps} \}$. Then $L \in \textbf{TIME}(g(n))$.

Assume $L \in \textbf{TIME}(f(n))$, then L is decided by some Turing machine M with running time $c \cdot f(n)$.

Time-Hierarchy Theorem

Take $L = \{ M \mid U(M, M) \text{ rejects in } g(|M|) \text{ steps} \}$. Then $L \in \mathbf{TIME}(g(n))$.

Assume $L \in \mathbf{TIME}(f(n))$, then L is decided by some Turing machine M with running time $c \cdot f(n)$. Therefore $U(M, M)$ runs in time

$$c'_M \cdot c \cdot f(|M|) \log f(|M|) .$$

Time-Hierarchy Theorem

Take $L = \{ M \mid U(M, M) \text{ rejects in } g(|M|) \text{ steps} \}$. Then $L \in \textbf{TIME}(g(n))$.

Assume $L \in \textbf{TIME}(f(n))$, then L is decided by some Turing machine M with running time $c \cdot f(n)$. Therefore $U(M, M)$ runs in time

$$c'_M \cdot c \cdot f(|M|) \log f(|M|) .$$

Construct M' that is functionally the same as M (so that $U(M', w)$ runs the same as $U(M, w)$), but has $n = |M'|$ large enough such that

$$c'_M \cdot c \cdot f(n) \log f(n) < g(n).$$

Time-Hierarchy Theorem

Take $L = \{ M \mid U(M, M) \text{ rejects in } g(|M|) \text{ steps} \}$. Then $L \in \textbf{TIME}(g(n))$.

Assume $L \in \textbf{TIME}(f(n))$, then L is decided by some Turing machine M with running time $c \cdot f(n)$. Therefore $U(M, M)$ runs in time

$$c'_M \cdot c \cdot f(|M|) \log f(|M|).$$

Construct M' that is functionally the same as M (so that $U(M', w)$ runs the same as $U(M, w)$), but has $n = |M'|$ large enough such that

$$c'_M \cdot c \cdot f(n) \log f(n) < g(n).$$

Then $M' \in L \Leftrightarrow U(M', M') \text{ rejects in } g(n) \text{ steps} \Leftrightarrow M' \notin L$.