CSE211 – Formal Languages and Automata Theory

U4L11_Computational Complexity

Dr. P. Saravanan
School of Computing
SASTRA Deemed to be University

Complexity

- A problem is decidable if there is an algorithm
- How to measure the complexity of program
- The set P is the set of problems or languages that can be decided by a Turing machine or some model of computation in polynomial time

Question?

- Assume that a problem (language) is decidable. Does that mean we can realistically solve it?
- NO, not always. It can require to much of time or memory resources.

Time Complexity: The number of steps during a computation

Space Complexity: Space used during a computation

What we use

- Henceforth, we only consider decidable languages and deciders.
- Our computational model is a Turing Machine.
- Time: the number of computation steps a TM machine makes to decide on an input of size n.
- Space: the maximum number of tape cells a TM machine takes to decide on a input of size n.

Complexity functions

- some common functions, ordered by how fast they grow

constant	$O(1)$
logarithmic	$O(\log n)$
linear	$O(n)$
n-log-n	$O(n \times \log n)$
quadratic	$O(n^2)$
cubic	$O(n^3)$
exponential	$O(k^n)$, e.g. $O(2^n)$
factorial	$O(n!)$
super-exponential	e.g. $O(n^n)$

Time Complexity

- We use a multitape Turing machine

- We count the number of steps until a string is accepted

- We use the $O(k)$ notation

Some notations

- The number of steps in measured as a function of n - **the size of the string representing the input.**
- In worst-case analysis, we consider the longest running time of all inputs of length n.
- In average-case analysis, we consider the average of the running times of all inputs of length n.

TIME COMPLEXITY

Let M be a deterministic TM that halts on all inputs. The time complexity of M if the function $f : \mathcal{N} \longrightarrow \mathcal{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n.
If $f(n)$ is the running time of M we say

- M runs in time $f(n)$
- M is an $f(n)$-time TM.

ASYMPTOTIC ANALYSIS

- We seek to understand the running time when the input is “large”.
- Hence we use an asymptotic notation or big-O notation to characterize the behaviour of f (n) when n is large.
- The exact value running time function is not terribly important.
- What is important is how f (n) grows as a function of n, for large n.
- Differences of a constant factor are not important.

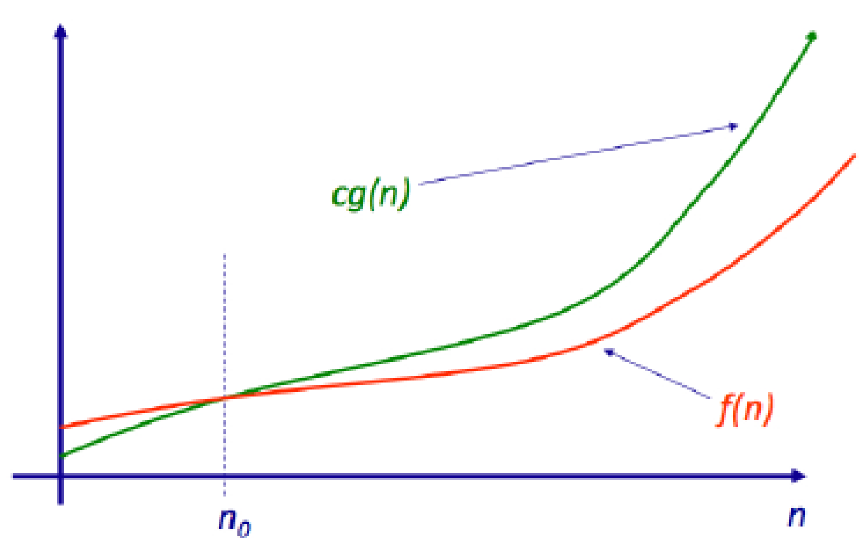
ASYMPTOTIC UPPER BOUND

Definition – Asymptotic Upper bound

Let $\mathcal{R}^+$ be the set of nonnegative real numbers. Let f and g be functions $f, g : \mathcal{N} \longrightarrow \mathcal{R}^+$. We say $f(n) = O(g(n))$, if there are positive integers c and n_0, such that for every $n \geq n_0$

$$f(n) \leq c\, g(n).$$

$g(n)$ is an asymptotic upper bound.



Reality Check

Assume that your computer/TM can perform 10^9 steps per second.

$n/f(n)$	n	$n\log(n)$	n^2	n^3	2^n
10	0.01 μsec	0.03 μsec	0.1 μsec	1 μsec	1 μsec
20	0.02 μsec	0.09 μsec	0.4 μsec	8 μsec	1 $msec$
50	0.05 μsec	0.28 μsec	2.5 μsec	125 μsec	13 days
100	0.10 μsec	0.66 μsec	10 μsec	1 $msec$	$\approxeq 4 \times 10^{13}$ years
1000	1 μsec	3 μsec	1 $msec$	1 sec	$\approxeq 3.4x10^{281}$ centuries

Clearly, if the running time of your TM is an exponential function of n, it does not matter how fast the TM is!

Example: $L = \{a^n b^n : n \geq 0\}$

Algorithm to accept a string w :

- Use a two-tape Turing machine
- Copy the a on the second tape
- Compare the a and b

$$L = \{a^n b^n : n \geq 0\}$$

Time needed:

- Copy the a on the second tape $\quad O(|w|)$

- Compare the a and b $\quad O(|w|)$

Total time: $\quad O(|w|)$

$$L = \{a^n b^n : n \geq 0\}$$

For string of length n

time needed for acceptance: $O(n)$

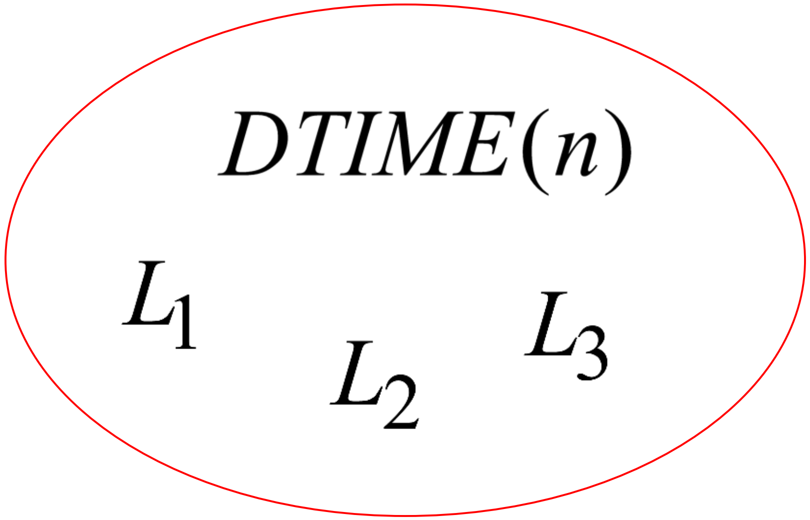
Complexity Classes

Definition – Time Complexity Class TIME($t(n)$)

Let $t : \mathcal{N} \longrightarrow \mathcal{R}^+$ be a function.
$\text{TIME}(t(n)) = \{L(M) \mid M \text{ is a decider running in time } O(t(n))\}$

- TIME($t(n)$) is the class (collection) of languages that are decidable by TMs, running in time $O(t(n))$.
- $\text{TIME}(n) \subset \text{TIME}(n^2) \subset \text{TIME}(n^3) \subset \ldots \subset \text{TIME}(2^n) \subset \ldots$
- Examples:
  - $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n^2)$
  - $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n \log n)$ (next slide)
  - $\{w\#w \mid w \in \{0,1\}^*\} \in \text{TIME}(n^2)$

Language class: $DTIME(n)$



A Deterministic Turing Machine accepts each string of length n in time $O(n)$

$$DTIME(n)$$

$$\{a^n b^n : n \geq 0\}$$

$$\{ww\}$$

In a similar way we define the class

$$DTIME(T(n))$$

for any time function: $T(n)$

Examples: $DTIME(n^2), DTIME(n^3), \ldots$

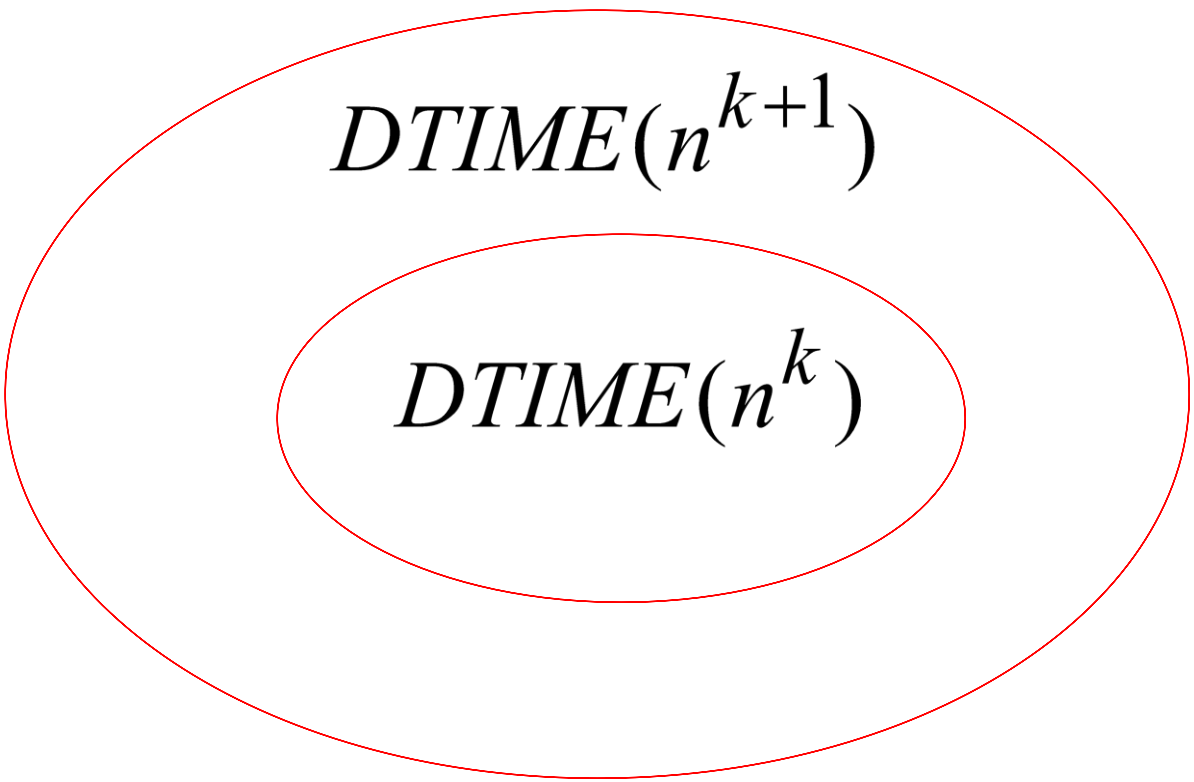
Example: The membership problem for context free languages

$$L = \{w : w \text{ is generated by grammar } G\}$$

$$L \in DTIME(n^3)$$ (CYK - algorithm)

Polynomial time

Theorem: $DTIME(n^k) \subset DTIME(n^{k+1})$



Polynomial time algorithms: $DTIME(n^k)$

Represent tractable algorithms:

For small k we can compute the result fast