# Theory of Computation (CS C/CS F 351)


BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

# *Class of P and NP problems*

## *Computational problems*

*1.* Solved by algorithms

- Practically solvable
- Not practical (due to excessive time requirements)

2. can't be solved by algorithms

# Algorithm's time complexity

Logarithmic $\log n$

Polynomial $n^c$

Exponential $2^n$

| n | Log n | n log n | $n^2$ | $n^3$ | $2^n$ | |
|---|---|---|---|---|---|---|
| 5 | 3 | 15 | 25 | 125 | 32 | |
| 10 | 4 | 40 | 100 | $10^3$ | $10^3$ | |
| 100 | 7 | 700 | $10^4$ | $10^6$ | $10^{30}$ | |
| 1000 | 10 | $10^4$ | $10^6$ | $10^9$ | $10^{300}$ | |

A Turing Machine *M* is said to be *polynomially bounded* if the machine always halts after *p(n)* steps.

*p(n)* is a polynomial function; *n* is the input size.

A language is polynomially bounded if there exists a polynomially bounded TM that decides it.

The class of polynomially decided languages are denoted by $\mathcal{P}$

The class of Nondeterministically polynomial (*NP*)
are the languages that are decided by polynomially bounded nondeterministic TMs.

# NP completeness

Let p1 and p2 are two problems.

A reduction from p1→p2 is an algorithm which converts instances of p1 to p2

, if the time taken by the algorithm is polynomial in terms of length of input to p1 , then reduction is called polynomial reduction.

Let B be a problem in NP

Then B is NP-Complete if

1. B is in NP
2. for every A in NP there exists a polynomial reduction of A to B

NP-Complete is subclass of NP

# Summary

*P* if there exists a TM with polynomial time to decide that

*NP* if there exists a Nondeterministic TM with polynomial time to decide that

*NP-Complete* if there exists a polynomial time reduction algorithm

*NP-hard* no such algorithm exists

# Applications

Finite automata: Lexical analysis, Spell checkers, Spelling advisors, dictionaries.
CFG: language description, Syntax Analysis, Develop XML.
Turing Machines: To model unrestricted language acceptors, Theory of undecidability which tells what problems can't be solved by computers.

# Conclusion to TOC

innovate achieve lead

Introduction to ToC

| | |
|---|---|
| Ch.1 | 1.1 to 1.4 Sets, Relations, Functions |
| | 1.7 Alphabets & languages |
| | 1.8 Finite representation of languages |
| Ch.2 | 2.1 Intro to FA and DFA |
| | 2.2. NDFA and conversions |
| | 2.4 Languages not regular & pumping theorem |
| | 2.5 State minimization |

Ch.3

3.1 Intro to CFL and CFG
3.2. parse trees
3.3 PDA
3.4 PDA & CFL
3.5 languages that are not CF
3.7 deterministic parsing (top-down & Bottom-up)
pumping theorem

We also discussed-

Grammar classification, normal forms and simplification.

| | |
|---|---|
| Ch. 4 | 4.1 Intro to TM |
| | 4.2. Computing with TM |
| | 4.3 Extension to TM |
| | 4.4 Random access TM |
| | 4.5 NDTM |
| | 4.6 Grammar |
| Ch. 5 | 5.1 Intro to undecidability |
| | 5.2 Universal TMs |
| Ch. 6&7 | Class of P, NP, NP complete and NP hard |

## Comprehensive examination

Date: 12th Dec AN session

35% weightage

# Thanks and Good Luck..