Theory of Computation (CS C/CS F 351)

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Computational problems

1. Solved by algorithms

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

2. can't be solved by algorithms

Algorithm and 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}	

Class of Problems

innovate achieve lead

Computational Problems

1 P - searching, sorting solvable and verifiable in polynomial time

2 NP - Travel sales man, Su-du-ko etc.
Not solvable in polynomial time but can be verified in Polynomial time.

- 2.1 NP-Hard optimization
- 2.2 NP-Complete Decision

Class of P and NP

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.

NP-hard

The class of polynomially decided languages are denoted by *P*

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

NP -hard

Let B be a problem in NP

Then B is NP-Hard if

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

NP-Hard class is not subclass of NP but some overlapping may be there.

NP -complete

If problem A can be reduced to Problem B (which has polynomial time solution) in polynomial time, 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 and NP Hard (intersection of NP and NP-hard)

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
NP-hard

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.

Summary of the Course - TOC

innovate achieve lead

Introduction to TOC

Ch.1	1.1 – 1.3 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
	2.3 FA and REs
	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 / CFG
3.5 languages that are not CF
3.7 deterministic parsing (top-down & Bottom-up)
pumping theorem
We also discussed- Grammar simplification;
First and Follows functions

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: 2nd , Dec (Mon) AN session

45% weightage

Close book

Special Note: Seating plan given by AUGSD

Thanks and Good Luck..