

# FUNDAMENTALS OF SOFTWARE ENGINEERING - CHAPTER 5

Timothy Hnat

Department of Computer Science  
University of Virginia

June 7, 2007



# MEASURING COMPLEXITY

- $O, o, \Theta, \theta, \Omega, \omega$
- Analyzing Algorithms

## EXAMPLE

- 



# COMPLEXITY RELATIONSHIPS

- Deterministic vs Non-deterministic Turing Machines



# THE CLASS P

- Polynomial time
- All reasonable deterministic computational models are polynomially equivalent



# THE CLASS P

- Polynomial time
- All reasonable deterministic computational models are polynomially equivalent

## DEFINITION

**P** is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine

$$P = \bigcup_k \text{TIME}(n^k)$$



# THE CLASS NP

- Non-deterministic Polynomial time



# THE CLASS NP

- Non-deterministic Polynomial time

## DEFINITION

**NP** is the class of languages that have polynomial time verifiers



# THE CLASS NP

- Non-deterministic Polynomial time

## DEFINITION

**NP** is the class of languages that have polynomial time verifiers

## EXAMPLES

- k-Clique
- Subset-Sum
- HamPath
- and many more...



# P vs NP

- P = the class of languages for which membership can be **decided** quickly.
- NP = the class of languages for which membership can be **verified** quickly.



# P vs NP

- P = the class of languages for which membership can be **decided** quickly.
- NP = the class of languages for which membership can be **verified** quickly.

## QUESTION

Does  $P = NP$ ?



# NP-COMPLETENESS

## THEOREM

**Cook-Levin theorem** -  $\text{SAT} \in \text{P}$  iff  $\text{P} = \text{NP}$ .

SAT is NP-complete.

Tableau proof



# NP-COMPLETENESS

## THEOREM

**Cook-Levin theorem** -  $SAT \in P$  iff  $P = NP$ .

SAT is NP-complete.

Tableau proof

## EXAMPLE

3SAT is polynomial time reducible to CLIQUE



# NP-COMPLETENESS

## THEOREM

**Cook-Levin theorem** -  $SAT \in P$  iff  $P = NP$ .

SAT is NP-complete.

Tableau proof

## EXAMPLE

3SAT is polynomial time reducible to CLIQUE

## DEFINITION

A language B is **NP-complete** if it satisfies two conditions:

- B is in NP, and
- every A in NP is polynomial time reducible to B



# SPACE COMPLEXITY

## DEFINITION

- $\text{SPACE}(f(n)) = \{L \mid L \text{ is a language decided by an } O(f(n)) \text{ space deterministic Turing machine}\}$
- $\text{NSPACE}(f(n)) = \{L \mid L \text{ is a language decided by an } O(f(n)) \text{ space nondeterministic Turing machine}\}$



# SAVITCH'S THEOREM

## SAVITCH'S THEOREM

For any function  $f: N \rightarrow \mathbb{R}^+$ , where  $f(n) \geq n$ ,  $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$



# THE CLASS PSPACE

## DEFINITION

**PSPACE** is the class of languages that are decidable in polynomial space on a deterministic Turing machine

$$PSPACE = \bigcup_k SPACE(n^k)$$



# THE CLASS PSPACE

## DEFINITION

**PSPACE** is the class of languages that are decidable in polynomial space on a deterministic Turing machine

$$PSPACE = \bigcup_k SPACE(n^k)$$

- $P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$
- We know that  $N \neq EXPTIME$



# PSPACE-COMPLETENESS

## DEFINATION

A language  $B$  is **PSPACE-complete** if it satisfies two conditions:

- $B$  is in PSPACE, and
- every  $A$  in PSPACE is polynomial time reducible to  $B$ .

If  $B$  merely satisfies condition 2, we say that it is **PSPACE-hard**.



# PSPACE-COMPLETENESS

## DEFINITION

A language  $B$  is **PSPACE-complete** if it satisfies two conditions:

- $B$  is in PSPACE, and
- every  $A$  in PSPACE is polynomial time reducible to  $B$ .

If  $B$  merely satisfies condition 2, we say that it is **PSPACE-hard**.

Examples of PSPACE-complete

- TQBF (true fully quantified Boolean formula)
- FORMULA-GAME
- GG (Generalized Geometry)



# THE CLASSES L AND NL

## DEFINITION

**L** is the class of languages that are decidable in logarithmic space on a deterministic Turing machine.

$$L = \text{SPACE}(\log n)$$

**NL** is the class of languages that are decidable in logarithmic space on a nondeterministic Turing machine.

$$NL = \text{NSPACE}(\log n)$$



# NL-COMPLETENESS

## DEFINITION

A language  $B$  is **NL-complete** if it satisfies two conditions:

- $B$  is in NL, and
- every  $A$  in NL is log space reducible to  $B$



# NL-COMPLETENESS

## DEFINITION

A language  $B$  is **NL-complete** if it satisfies two conditions:

- $B$  is in NL, and
  - every  $A$  in NL is log space reducible to  $B$
- 
- Log space transducer - a Turing machine with a read-only input tape, a write-only output tape, and a read/write work tape.
  - The work tape may contain  $O(\log n)$  symbols



# NL = coNL

- Does  $NL = coNL$ ?



# NL = coNL

- Does  $NL = coNL$ ?
- Yes
- Proof?

## Summary

- $L \subseteq NL = coNL \subseteq P \subseteq PSPACE$



# INTRACTABILITY

What is Intractability?

- Computational problems that are solvable in principle, but the solutions require so much time or space that they can't be used in practice.



# HIERARCHY THEOREMS

- **Space hierarchy theorem:** For any space constructible function  $f: \mathbb{N} \rightarrow \mathbb{N}$ , a language  $A$  exists that is decidable in  $O(f(n))$  space but not in  $o(f(n))$  space.



# TIME HIERARCHY

- **Time hierarchy theorem:** For any time constructible function  $f: \mathbb{N} \rightarrow \mathbb{N}$ , a language  $A$  exists that is decidable in  $O(t(n))$  space but not in  $o(t(n)/\log t(n))$ .



## DEFINITION

A language  $B$  is **EXPSPACE-complete** if:

- $B$  is in EXPSPACE, and
- every  $A$  in EXPSPACE is polynomial time reducible to  $B$



# RELATIVIZATION

- Allows a TM to have an **Oracle**
- What does this get us?
- To solve the P vs. NP question, we must analyze computations, not just simulate them



# CIRCUIT COMPLEXITY

- Boolean circuit: a collection of gates and inputs connected by wires. Cycles are not permitted
- AND, OR, and NOT gates
- Similar to SAT (3SAT)

