CS F364
Design & Analysis of Algorithms

Algorithms - Complexity

Complexity Classes

- Time Complexity Classes vs. Space Complexity Classes

- Tractability: Polynomial Time vs. Exponential Time

COMPLEXITY CLASSES - DEFINITION

- A ***complexity class*** is – typically – a class of problems
  - each of which is solvable by at least one algorithm of certain time complexity and/or space complexity under a specific machine model
- Typically the machine model is the deterministic Turing Machine (DTM) model OR the non-deterministic Turing Machine (NDTM) model
  - The DTM model may often be substituted by the RAM model or equivalently a general purpose programming language
  - The word "deterministic" refers to the nature of the computation: *from a given state, on a given input, the machine will go to one specific state*

Complexity Classes - Definition [2]

- Typically we restrict ourselves to (complexity) classes of ***decision problems*** unless otherwise specified.
- Any complexity function f referred – in the definition of a complexity class – must be a *proper complexity function*, i.e.
  - f : N --> N
  - f is monotonic i.e. f(n+1) >= f(n) for all n
  - f(n) itself can be computed in O(n+f(n)) time using O(f(n)) space.
- Question:
  - Which of the following are proper complexity functions?
    - sin(x)
    - f(N) = 1 if N is even, √N if N is odd.
    - f(N) = p the largest prime factor of N

Complexity Classes – Generic Definitions

- Define **TIME(f(n))** as the *complexity class of problems* that
  - can be solved by (RAM) algorithms of time complexity O(f(n)) in the worst case, where n is the size of the input
    - TIME(f(n)) =

      $\{ \pi \mid \exists \text{ algorithm } A: A \text{ solves } \pi \text{ in } O(f(n)) \text{ time} \}$

Complexity Classes – Generic Definitions

- TIME(f(n)) =

  { π | ∃ algorithm A: A solves π in O(f(n)) time}

- Similarly, define **SPACE(f(n))** as the *complexity class of problems* that
  - can be solved by (RAM) algorithms of space complexity O(f(n)) in the worst case, where n is the size of the input:
    - SPACE(f(n)) =

      { π | ∃ algorithm A: A solves π in O(f(n)) space}

- Question:
  - Given a function f(n), is there a relation between TIME(f(n)) and SPACE(f(n))?

Relation Between Time(F(n)) and Space(F(n)) ?

Complexity Classes – Examples [2]

- An algorithm is a *polynomial time algorithm* if
  - the time taken by the algorithm is $O(N^k)$ for some positive integer constant k, where N is the input size.
- Define $\mathbb{P} = \{ \pi \mid \pi$ is a decision problem that can be solved by a polynomial time algorithm $\}$
  - i.e. $\mathbb{P} = \mathrm{U}_{k \in \mathbb{N}}\ \mathrm{TIME}(n^k)$

Complexity Classes - Examples [3]

- Define $\mathbb{EXP}$ = { π | π is a decision problem that can be solved by an exponential time algorithm }
  - i.e. $\mathbb{EXP} = \cup_{k \in \mathbb{N}} \text{TIME}(2^{poly(n)})$ where poly(n) is a polynomial function of n.
- $\mathbb{P} \subseteq \mathbb{EXP}$

  By definition.
  - Is $\mathbb{P} \subset \mathbb{EXP}$?

Complexity Classes - Hierarchy

- Define a function H_f such that
  - $H_f(A,x) = 1$ if $A(x)$ halts after $f(|x|)$ steps and returns 1
    $= 0$ otherwise
- Lemma 3:
  - H_f can be solved in $O((f(n))^3)$ time i.e. $H_f \in TIME((f(n))^3)$
  - Proof:
    - By construction.
- Lemma 4:
  - $H_f \notin TIME(f(\lfloor n/2 \rfloor))$
  - Proof:
    - By diagonalization.
- The Time Hierarchy Theorem:
  - $TIME(f(n)) \subset TIME((f(2n+1))^3)$ for any $f(n) >= n$

Complexity Classes - Hierarchy [2]

- The Time Hierarchy Theorem:
  - $\text{TIME}(f(n)) \subset \text{TIME}((f(2n+1))^3)$ for any $f(n) >= n$
- Corollary:
  - $\mathbb{P} \subset \mathbb{EXP}$
  - Proof:
    - $\mathbb{P} \subseteq \text{TIME}(2^n)$ by definition of polynomial functions and $\mathbb{P}$
    - $\text{TIME}(2^n) \subset \text{TIME}((2^{2n+1})^3)$ by the time hierarchy theorem
    - $\text{TIME}((2^{2n+1})^3) = \text{TIME}(2^{6n+3}) \subseteq \mathbb{EXP}$ by definition of $\mathbb{EXP}$

Complexity Classes - Tractability

- $\mathbb{P} \subset \mathbb{EXP}$
  - Example:
    - GENERALIZED_CHESS $\in \mathbb{EXP} - \mathbb{P}$
- We say that
  - a problem π is *tractable* if $\pi \in \mathbb{P}$ and
  - it is *intractable* otherwise

(SPACE) COMPLEXITY CLASSES -

- Define
  - PSPACE = SPACE(f(n))
    - where f(n) is n^k for some constant k>=0
  - e.g. 3-SAT $\in$ PSPACE
  - e.g. QSAT $\in$ PSPACE
- QSAT is defined as follows:
  - Given a quantified propositional logic formula of the form
    - $\forall x_1 \exists x_2 \forall x_3 \exists x_4 \ldots \forall x_{2n-1} \exists x_{2n} \varphi$
  - find whether it evaluates to TRUE.

- Compare this with SAT.

(SPACE) COMPLEXITY CLASSES -

- Define
  - PSPACE = SPACE(f(n))
    - where f(n) is n^k for some constant k>=0
  - e.g. QSAT $\in$ PSPACE
- Define
  - L = SPACE(f(n))
    - where f(n) is log(n)