Communication complexity

two parties: Alice and Bob
function \( f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \)
Alice holds \( x \in \{0,1\}^n \); Bob holds \( y \in \{0,1\}^n \)

- **Goal**: compute \( f(x, y) \) while communicating as few bits as possible between Alice and Bob
- **Example**: \( EQ(x, y) = 1 \) iff \( x = y \)
- **Deterministic protocol**: no fewer than \( n+1 \) bits
- **Randomized protocol**: \( O(\log n) \) bits

Extended Church-Turing Thesis

- Common to insert "probabilistic":

The "extended" Church-Turing Thesis
everything we can compute in time \( t(n) \) on a physical computer can be computed on a probabilistic Turing Machine in time \( t(n)^{O(1)} \) (polynomial slowdown)

Randomized complexity classes

- **model**: probabilistic Turing Machine
  - deterministic TM with additional read-only tape containing "coin flips"
  
  ```
  \[
  \begin{array}{cccc}
  \text{input tape} & 0 & 1 & 0 & 2 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & \ldots
  \\
  \text{finite control} & \text{q}_0
  \\
  \text{read/write head}
  \\
  \text{read head} & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & \ldots
  \end{array}
  \]
  ```

- **RP** (Random Polynomial-time)
  - \( L \in \text{RP} \) if there is a p.p.t. TM \( M \):
    \[
    x \in L \implies \Pr_y[M(x, y) \text{ accepts}] \geq \frac{1}{2}
    
    x \notin L \implies \Pr_y[M(x, y) \text{ rejects}] = 1
    \]

- **coRP** (complement of Random Polynomial-time)
  - \( L \notin \text{coRP} \) if there is a p.p.t. TM \( M \):
    \[
    x \in L \implies \Pr_y[M(x, y) \text{ accepts}] = 1
    
    x \notin L \implies \Pr_y[M(x, y) \text{ rejects}] \geq \frac{1}{2}
    
    "p.p.t" = probabilistic polynomial time
Randomized complexity classes

- **BPP** (Bounded-error Probabilistic Poly-time)
  - $L \in \text{BPP}$ if there is a p.p.t. TM $M$:
    - $x \in L \rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq 2/3$
    - $x \notin L \rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq 2/3$

One more important class:

- **ZPP** (Zero-error Probabilistic Poly-time)
  - $ZPP = RP \cap \text{coRP}$
  - $\Pr_y[M(x,y) \text{ outputs "fail"]} \leq 1/2$
  - otherwise outputs correct answer

Randomized complexity classes

These classes may capture “efficiently computable” better than $P$.

Relationship to other classes

- all these classes contain $P$
  - they can simply ignore the tape with coin flips
- all are in $PSPACE$
  - can exhaustively try all strings $y$
  - count accepts/rejects; compute probability
- $RP \subseteq NP$ (and $\text{coRP} \subseteq \text{coNP}$)
  - multitude of accepting computations
  - $NP$ requires only one

Polynomial identity testing

- Given: polynomial $p(x_1, x_2, \ldots, x_n)$ as arithmetic formula (fan-out 1):
  - multiplication (fan-in 2)
  - addition (fan-in 2)
  - negation (fan-in 1)

  variables take values in finite field $F$

- Question: Is $p$ identically zero?
  - i.e., is $p(x) = 0$ for all $x \in F^n$
  - (assume $|F|$ larger than degree…)

  “polynomial identity testing” because given two polynomials $p, q$, we can check the identity $p \equiv q$ by checking if $(p - q) \equiv 0$
Polynomial identity testing

• try all $|F|^n$ inputs?
  – may be exponentially many
• multiply out symbolically, check that all coefficients are zero?
  – may be exponentially many coefficients
• Best known deterministic algorithm places in EXP

Polynomial identity testing

Lemma (Schwartz-Zippel): Let $p(x_1, x_2, \ldots, x_n)$ be a total degree $d$ polynomial over a field $F$ and let $S$ be any subset of $F$. Then if $p$ is not identically 0,
$$\Pr_{r_1, r_2, \ldots, r_n \in S}[p(r_1, r_2, \ldots, r_n) = 0] \leq d/|S|.$$
Randomized complexity classes

- believed that $P = ZPP = RP = coRP = BPP$ (!)

Course Review

Review

- Highest level: 2 main points

  1. Decidability
     - problem solvable by an algorithm = problem is decidable
     - some problems are not decidable (e.g. HALT)

  2. Tractability
     - problem solvable in polynomial time = problem is tractable
     - some problems are not tractable (EXP-complete problems)
     - huge number of problems are likely not to be tractable (NP-complete problems)

Review

- Highest level: 2 main points

Review

- Important ideas (continued):
  - simulation used to show one model at least as powerful as another
  - diagonalization used to show one model strictly more powerful than another
    - also Pumping Lemma
  - reduction used to compare one problem to another

Review

- Important ideas
  - "problem" formalized as language
    - language = set of strings
  - "computation" formalized as simple machine
    - finite automata
    - pushdown automata
    - Turing Machine
  - "power" of machine formalized as the set of languages it recognizes
Review

• Important ideas (continued):
  – complexity theory investigates the resources required to solve problems
    • time, space, others…
  – complexity class = set of languages
  – language L is C-hard if every problem in C reduces to L
  – language L is C-complete if L is C-hard and L is in C.

A complete problem is a surrogate for the entire class.

Summary

Part I: automata

Finite Automata

• Non-deterministic variant: NFA
• Regular expressions built up from:
  – unions
  – concatenations
  – star operations
Main results: same set of languages recognized by FA, NFA and regular expressions (“regular languages”).

Pushdown Automata

finite control

finite input tape

New capabilities:
• can push symbol onto stack
• can pop symbol off of stack

(infinite) stack

0

1

2

3

4

5

6

7

8

9

10

11
Context-Free Grammars

A → 0A1
A → B
B → #

Pushdown Automata

Main results: same set of languages recognized by NPDA, and context-free grammars ("context-free languages").
- and DPDA’s weaker than NPDA’s...

Non-regular languages

Pumping Lemma: Let L be a regular language. There exists an integer p ("pumping length") for which every w ∈ L with |w| ≥ p can be written as
w = xyz such that
1. for every i ≥ 0, xy^i z ∈ L, and
2. |y| > 0, and
3. |xy| ≤ p.

Pumping Lemma for CFLs

CFL Pumping Lemma: Let L be a CFL. There exists an integer p ("pumping length") for which every w ∈ L with |w| ≥ p can be written as
w = uvxyz such that
1. for every i ≥ 0, uv^ixy^i z ∈ L, and
2. |vy| > 0, and
3. |vxy| ≤ p.

Summary

Part II: Turing Machines and decidability

Turing Machines

- New capabilities:
  - infinite tape
  - can read OR write to tape
  - read/write head can move left and right
Deciding and Recognizing

- TM M:
  - \( L(M) \) is the language it recognizes
  - if M rejects every \( x \notin L(M) \) it decides \( L \)
  - set of languages recognized by some TM is called Turing-recognizable or recursively enumerable (RE)
  - set of languages decided by some TM is called Turing-decidable or decidable or recursive

Church-Turing Thesis

- the belief that TMs formalize our intuitive notion of an algorithm is:

  The Church-Turing Thesis
  
  everything we can compute on a physical computer can be computed on a Turing Machine

  • Note: this is a belief, not a theorem.

The Halting Problem

- The existence of \( H \) which tells us yes/no for each box allows us to construct a TM \( H' \) that cannot be in the table.

Decidable, RE, coRE...

- some problems (e.g. HALT) have no algorithms

Definition of reduction

- More refined notion of reduction:
  - "many-one" reduction (commonly)
  - "mapping" reduction (book)

Using reductions

- Used reductions to prove lots of problems were:
  - undecidable (reduce from undecidable)
  - non-RE (reduce from non-RE)
    - or show undecidable, and coRE
  - non-coRE (reduce from non-coRE)
    - or show undecidable, and RE

  Rice’s Theorem: Every nontrivial TM property is undecidable.
The Recursion Theorem

**Theorem:** Let $T$ be a TM that computes $f_T: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. There is a TM $R$ that computes the fn: $r: \Sigma^* \rightarrow \Sigma^*$ defined as $r(w) = t(w, \langle R \rangle)$.

• In the course of computation, a Turing Machine can output its own description.

Incompleteness Theorem

**Theorem:** Peano Arithmetic is not complete.

(same holds for any reasonable proof system for number theory)

Proof outline:
– the set of theorems of PA is RE
– the set of true sentences (= $\text{Th}(\mathbb{N})$) is not RE

Summary

Part III: Complexity

Complexity

• Complexity Theory = study of what is computationally feasible (or tractable) with limited resources:
  – running time
  – storage space
  – number of random bits
  – degree of parallelism
  – rounds of interaction
  – others…

main focus

not in this course

Time and Space Complexity

**Definition:** the time complexity of a TM $M$ is a function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps $M$ uses on any input of length $n$.

**Definition:** the space complexity of a TM $M$ is a function $f: \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of tape cells $M$ scans on any input of length $n$.

Complexity Classes

**Definition:** $\text{TIME}(t(n)) = \{L : \text{there exists a TM } M \text{ that decides } L \text{ in time } O(t(n))\}$

$P = \bigcup_{k \geq 1} \text{TIME}(n^k)$

$\text{EXP} = \bigcup_{k \geq 1} \text{TIME}(2^{n^k})$

**Definition:** $\text{SPACE}(t(n)) = \{L : \text{there exists a TM } M \text{ that decides } L \text{ in space } O(t(n))\}$

$\text{PSPACE} = \bigcup_{k \geq 1} \text{SPACE}(2^{n^k})$
Complexity Classes

**Definition:** $\text{NTIME}(t(n)) = \{ L : \text{there exists a NTM M that decides } L \text{ in time } O(t(n)) \}$

$\text{NP} = \bigcup_{k \geq 1} \text{NTIME}(2^{n^k})$

- Theorem: $\text{P} \subseteq \text{EXP}$
- $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP}$
- Don’t know if any of the containments are proper.

Alternate definition of NP

**Theorem:** language $L$ is in NP if and only if it is expressible as:

$L = \{ x \mid \exists y, \ |y| \leq |x|^k, (x, y) \in R \}$

where $R$ is a language in $\text{P}$.

Poly-time reductions

- Type of reduction we will use:
  - "many-one" poly-time reduction (commonly)
  - "mapping" poly-time reduction (book)

  ![Reduction Diagram](image)

  1. $f$ poly-time computable
  2. YES maps to YES
  3. NO maps to NO

Hardness and completeness

**Definition:** a language $L$ is **C-hard** if for every language $A \in C$, $A$ poly-time reduces to $L$; i.e., $A \leq^P L$.

Can show $L$ is C-hard by reducing from a known C-hard problem

**Definition:** a language $L$ is **C-complete** if $L$ is C-hard and $L \in C$

Complete problems

- EXP-complete: $\text{ATM}_B = \{ <M, x, m> : M \text{ is a TM that accepts } x \text{ within at most } m \text{ steps} \}$
- PSPACE-complete: $\text{QSAT} = \{ \phi : \phi \text{ is a 3-CNF, and } \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n \phi(x_1, x_2, \ldots x_n) \}$
- NP-complete: $\text{3SAT} = \{ \phi : \phi \text{ is a satisfiable 3-CNF formula} \}$

Lots of NP-complete problems

- Indendent Set
- Vertex Cover
- Clique
- Hamilton Path (directed and undirected)
- Hamilton Cycle and TSP
- Subset Sum
- NAE3SAT
- Max Cut
- Problem sets: max/min Bisection, 3-coloring, subgraph isomorphism, subset sum, (3,3)-SAT, Partition, Knapsack, Max2SAT...
Other complexity classes

• coNP – complement of NP
  – complete problems: UNSAT, DNF-TAUTOLOGY

• NP intersect coNP
  – contains (decision version of ) FACTORING

• PSPACE
  – complete problems: QSAT, GEOGRAPHY

Complexity classes

decidable languages

all containments believed to be proper