Outline

• Overview
• Definitions
• Computing Machines
• Church-Turing Thesis
• Polynomial Time (Class $\mathbf{P}$)
• Class $\mathbf{NP}$
• Non-deterministic Turing machines
• Reducibility
• Cook-Levin theorem
• Coping with $\mathbf{NP}$ Hardness
Overview from First Lecture

- Study of the complexity of algorithms
- Based on Turing machines
- Often, one compares algorithms for best scaling in large problems

Alan Turing (from University of Calgary Centenary event)
Definitions

• Algorithm – set of rules for computing a function mapping input data to output
• Specific realizations of computers:
  – Turing machines
  – Lambda calculus
  – μ-recursive functions
  – Register machines
• Running time – number of basic operations required
• Efficient computability
Turing Machine

• Performs the following elementary operations:
  – Read a bit of input and byte of data from ‘work’
  – Write a bit to ‘work’
  – Choose next state $Q$. Either:
    • Stop and output a 0 or 1
    • Choose a new rule to apply next

$\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^{k-1} \times \{L, S, R\}^k$
Turing Machine: Behavior in Each State

- **q_{start}**: write start symbol, change state to q_{copy}
- **q_{copy}**: copy non-blank symbol, otherwise go to q_{right}
- **q_{right}**: go back to start, then q_{test}
- **q_{test}**: if at start and ‘work’ is blank, write 1; else, write 0
- **q_{halt}**: finished
Time Constructible Functions

• A function $T: \mathbb{N} \rightarrow \mathbb{N}$ is time constructible iff
  - $T(n) \geq n$
  - TM $M$ computes $x \rightarrow T(x)$ in time $T(n)$

• Examples: $n$, $n \log n$, $2^n$
Turing Machine: Key Properties

• If $f$ is computable in $T(n)$....
  – with alphabet $\Gamma$, it is computable in $4 \log \Gamma T(n)$ with TM
  – with $k$ tapes, it’s computable in $5kT(n)^2$ with TM
  – with bidirectional tape, computable in $4T(n)$ with TM
Universal Turing Machines

- A universal Turing machine can simulate every other TM
- Efficient process: takes $CN \log N$ time to simulate a process taking $N$ steps
Other Computing Machines

• Register machines
  – Post-Turing machine: adds jump and erase functions
  – Many other variations, getting closer to assembly language

• Lambda calculus – basis for Scheme and LISP

• \( \mu \)-recursive functions – partial functions mapping \( \mathbb{N} \rightarrow \mathbb{N} \)

• Which one should we use, in which situations?
Church-Turing (CT) Thesis

“Every physically realizable computing machine can be simulated by a Turing machine, or any other type of machine.”

• This thesis has not been rigorously proven
• Not applicable for non-deterministic systems
• Quantum computers have caused some to re-evaluate CT thesis
Uncomputability

• It may seem obvious everything is computable with enough time, but...
• There’s always an uncomputable function UC
• Another example: the halting problem
  – Computes whether another function will finish
  – Cannot be computed in general
Big-Oh Notation

• Take two functions $f, g$ mapping $\mathbb{N} \rightarrow \mathbb{N}$, there exists a constant $c$ s.t. $f(n) \leq c \cdot g(n)$
  – Sometimes write $f(n) = O(g(n))$

• Examples
  – If $f(n) = 100n^2 + 24n + 2 \log n$ and $g(n) = n^2$, then $f = O(g)$
  – If $f(n) = e^n$ and $g(n) = n^c$ for every $c$ in $\mathbb{N}$, then $g = O(f)$
Polynomial Complexity

• Meant to capture decision problems that are feasible
• Might think of efficient computations as being $O(N)$ or $O(N^2)$
• Often symbolized by $P$
• Formally, $P = \bigcup_{c \geq 1} DTIME(n^c)$, where $DTIME(T)$ is computable in $cT(n)$ time
• Strong form of CT thesis says all simulations can be done with $P$ overhead
Other Classes

- Turing machines with randomness – class BPP
- Quantum computers – class BQP
Examples

- CONNECTED – the set of all connected graphs
- TRIANGLEFREE – the set of all graphs without a triangle
- BIPARTITE – the set of all bipartite (distinct) graphs
- TREE – the set of all trees
Summary

• Church-Turing thesis used to demonstrate several models of computation:
  – Turing machines
  – Register machines
  – Lambda calculus
• Big-oh notation example: $\text{DTIME}(n^2) = \mathcal{O}(n^2)$
• Polynomial complexity: $P = \bigcup_{c \geq 1} \text{DTIME}(n^c)$,
Class **NP** Definition

- Decision problems that can be efficiently verified
- The language $L$ subset $\{0,1\}^*$ is in **NP** if there exists a polynomial $p: \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial time TM $M$ s.t. for every $x$:
  - $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$
- If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfies $M(x,u) = 1$, then $u$ is a **certificate** for $x$
Examples of \textbf{NP}

- Independent set – find a $k$-size subset of a given graph $G$’s vertices
- Traveling salesman – given a set of $n$ nodes and pairwise distances, find a closed circuit that visits every node once of length no more than $k$
- Linear/integer programming – given a set of $m$ linear inequalities for $n$ variables, find a set of rational/integer solutions
- Graph isomorphism – given two $n \times n$ adjacency matrices $M_1$ and $M_2$, decide if they define the same graph
- Composite numbers – decide if number $N$ is prime
- Factoring – for a number $N$, find a factor $M$ in the interval $[L, U]$
Relationship of $P$ & $NP$

- $P$ is a subset of $NP$
- Formally, $P \subseteq NP \subseteq \bigcup_{c \geq 1} \text{DTIME}(2^{nc})$
- Recall $\text{DTIME}(T)$ is computable in $cT(n)$ time
Relationship of P & NP

• Key open question: does \( P = NP \)?

• Some \textbf{NP} problems are in \textbf{P}:
  – Connectivity – breadth-first search
  – Composite numbers – Agrawal, Kayal, and Saxena solved this a few years ago
  – Linear programming – ellipsoidal algorithm of Khachiyan
Relationship of P & NP

• Problems *not* shown to be in P:
  – Independent set
  – Traveling salesman
  – Subset sum
  – Integer programming

• This general group is known as NP complete:
  i.e., not in P unless P=NP
Non-deterministic Turing machines

• Has two transition functions $\delta_0$ and $\delta_1$
• The NDTM has another internal state, $q_{\text{accept}}$
• M outputs 1 if some (non-deterministic) sequence of choices $\rightarrow q_{\text{accept}}$
• Can also define $\text{NTIME}$ s.t. NDTM takes $cT(n)$ time $\forall x \in L \iff M(x) = 1$
• Alternative definition: $\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$
Reducibility

- Karp reductions relate the various example problems to one another
- Language $A$ reduces to $B$ if there is a polynomial-time computable function $f$, s.t. $\forall x \in \{0,1\}^*, x \in A$ iff $f(x) \in B$
- $B$ is **NP-hard** if $A \leq_p B$, $\forall A \in \text{NP}$
NP completeness

• By transitivity of Karp reduction, there should be a language that’s “hardest,” known as NP-complete

• The TMSAT language is NP-complete
  – $TMSAT = \{\langle \alpha, x, 1^n, 1^t \rangle : \exists u \in \{0,1\}^n \}$
  – $M_\alpha$ denotes TM represented by $\alpha$
  – $M_\alpha$ outputs 1 on $\langle x, u \rangle$
Cook-Levin theorem

• CNF: conjunctive normal form used for a Boolean formula consisting of AND’s and OR’s
• kCNF: CNF consisting of k AND’s: \( \bigwedge_{i=1}^{k} (\bigvee j \, \nu_{ij}) \)
• Cook-Levin Theorem:
  – Let SAT be the language of all satisfiable CNF formulae: SAT is NP-complete
  – Let 3SAT be the language of all satisfiable 3CNF formulae: 3SAT is NP-complete
Web of reductions

∀L ∈ NP

Theorem 2.10 (Lemma 2.12)

SAT

Theorem 2.10 (Lemma 2.15)

INTEGERPROG

Theorem 2.17

3SAT

Ex 16

Exactone3SAT

Ex 16

SUBSETSUM

Theorem 2.16

INDSET

Ex 14

VERTEXCOVER

Ex 15

MAXCUT

THEOREMS

Ex 6

TSP

Ex 17

HAMPATH

dHAMPATH

Ex 17

HAMCYCLE

Ex 18

QUADEQ

Ex 17

Theorem 2.18

27
Coping with \textbf{NP}-hardness

• Approximate/heuristic solutions
• Example: traveling salesman
• Key considerations:
  – Maximum error
  – Average-case complexity
Related Complexity Classes

- \( \text{coNP} - x \in L \iff \forall u \in \{0,1\}^{|x|} \text{ s.t. } M(x,u)=1 \)
- \( \text{EXP} = \bigcup_{c \geq 0} \text{DTIME}(2^{n^c}) \)
- \( \text{NEXP} = \bigcup_{c \geq 0} \text{NTIME}(2^{n^c}) \)
- Must have \( P \subseteq \text{NP} \subseteq \text{EXP} \subseteq \text{NEXP} \)
What if $P=NP$?

- Any decision (or equivalently, search) problem could be solved in a reasonable amount of time
- Perfectly optimal designs for all areas of engineering could be chosen
- Automated theory generation for experimental results
- No privacy in the digital domain
Next Class

• Discussion of methods for assessing code performance
• This will include some realistic examples