Theory NP

Week 17

Required Activities

- Check Announcements regularly (every 2-3 days)
- Read FA book: Chapter 9
- Complete and submit Assignment 5 (due Feb 15)

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it
- Worst-case growth rate can be bounded by a polynomial
- Function of its input size
  - $P(n) = a_n n^k + . . . + a_1 n + a_0$ where k is a constant
  - $P(n)\ \varepsilon\ \theta(n^k)$
  - E.g. 2n, $3n^3$, n log n

Intractability

- **Dictionary Definition** of intractable: “difficult to treat or work.”
- **Computer Science**: problem is intractable if a computer has difficulty solving it
- A problem is intractable if it is not tractable
- Any algorithm with a growth rate not bounded by a polynomial
  - c^n, $c^{.01n}$, n^{logn}, $n!$
- Property of the problem not the algorithm

Three General Categories of Problems

- Problems for which **polynomial-time** algorithms have been found

- Problems that have been proven to be **intractable**

- Problems that have **not been proven** to be intractable, but for which polynomial-time algorithms have never been found

Algorithms

- **Polynomial-time Algorithms**
  - Θ(nlogn) for sorting
  - Θ(logn) for searching
  - Θ(n^3) for chained-matrix multiplication
- Proven to be **Intractable**
  - Unrealistic definition of the Problem (Hamiltonian Circuits)
  - Un-Decidable problems: The Halting Problem (proven un-decidable by Alan Turing).
  - Decidable intractable problems: researchers have shown some problems from automata and mathematical logic intractable
- **Not proven** to be intractable (there could exist polynomial time algorithm)
  - Traveling salesperson
  - 0-1 Knapsack
  - Graph coloring
  - Sum of subsets

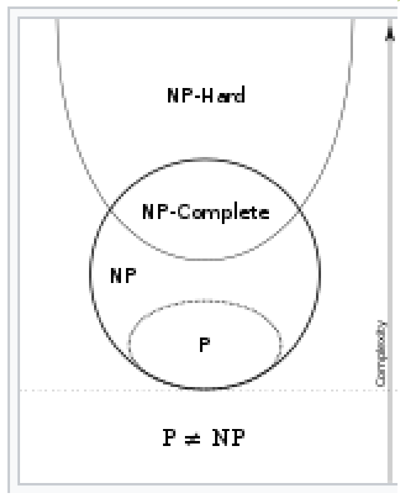
Nondeterministic Polynomial time (NP)

- **Decision Problems** – output is “yes” or “no” (optimization problems are decision problems)
  - **Traveling Salesperson**: For a given positive number d, is there a tour having length <=d?
  - **0-1 Knapsack**: For a given profit P, is it possible to load the knapsack such that total weight <=W?

- **Deterministic** - given specific input, will produce same output

- **NP**
  - Decision problems solvable in polynomial time by a theoretical **non-deterministic** Turing machine (guess about solution in non-deterministic way then verify with deterministic algorithm to verify or reject guess)
  - Decision problems where “yes” have verifiable **proofs by deterministic** computations performed in **polynomial time**

NP Problems



Author: Behnam Esfahbod

- **P –** decision problems that can be **solved** by polynomial time algorithms
- **Polynomial-time nondeterministic algorithm** – verification stage is polynomial-time algorithm
- **NP –** nondeterministic polynomial time
  - Decision problems that can be “solved” by polynomial-time nondeterministic algorithm
  - Which means there is algorithm to **verify** in polynomial time
  - May not be an algorithm to actually solve problem in polynomial-time

- **NP-complete (NP-C)**
  - Every NP-complete is NP-hard but not vice versa
  - By definition: It is NP (there is verify polynomial algorithm) and every problem in NP can be reduced to it
  - In other words, solution can be **verified** quickly (polynomial time) but there is no known fast solution
  - In practice: New problem can be proven to be NP-complete if can reduce to known NP-complete problem
  - Usually solved using heuristic methods and approximation algorithms

- **NP-hard**
  - At least as hard as any NP problem to solve (can be harder)
  - Includes non-decision problems
  - Algorithm for “solving” it can be reduced into one for solving any other NP-complete in polynomial time

Examples

- **NP-complete (also by definition NP and NP-hard)** when expressed as decision problem
  - Knapsack problem
  - Subset sum problem
  - Traveling salesman problem
  - Hamiltonian path problem
- **Halting problem** is NP-hard but not NP-complete (it is intractable)

Is P contained in NP?

- It has not been proven that there is a problem in NP that is not in P
- NP-P may be empty
- P=NP? One of the most important questions in CS
  - To show P!=NP, find a problem in NP that is not in P
  - To show P = NP, find polynomial-time algorithm for each problem in NP

Questions ?

- Post in the discussions
- Send email to RMcFadden@HarrisburgU.edu
- Respond usually within 48hours