Applied Cryptography
CPEG 472/672
Lecture 9A

Instructor: Nektarios Tsoutsos

Hard computational problems

- Simple to describe, impossible to solve
  - Very useful in cryptography
  - RSA, Diffie-Hellman, Lattice crypto
- Computational hardness
  - No algorithm will run in reasonable time
  - Intractable problems: practically impossible to solve on any computer
    - The target computer does not matter due to equivalence of computing models

Computational complexity

- Approximate # of alg operations as a function of the input size
  - Naïve search: Loop over *n* array elements
    - Complexity: number of loop iterations
    - Linear to number of elements n
  - Sorting: n*log(n) operations to sort a list
    - The list has *n* elements
    - Linearithmic complexity: grows faster than n
  - Brute force cipher key: 2^n attempts
    - The key is n bits
    - Exponential complexity (impractical)

Big O notation

- Used to express complexity
  - Ignores constant factors
  - E.g., $12345 * n^3$ is $O(n^3)$, $41 * n^3$ is $O(n^3)$
  - Finding the LSB is $O(1)$ (constant time)
- Big-O: upper bound for complexity
  - $O(n)$: linear
  - $O(2^n)$: exponential
  - $O(n^2)$: quadratic
  - $O(n^k)$: polynomial (polytime)
  - Superpolynomial: grows faster than any poly
    - $O(2^n)$, $O(n^{\log(n)})$, $O(n^n)$, $O(n^{f(n-1)})$ with $f(x)=x^{f(x-1)}$

Computational complexity

- Big-O examples
  - Multiply two n-bit integers: $O(n^{1.465})$
  - Multiply two n x n matrices: $O(n^{2.373})$
  - Identify an n-bit prime: $O(n^6)$
  - Brute force n-bits: $O(2^n)$
- Complexity classes
  - Problems solvable in $O(n^k)$ time: $TIME(n^k)$
    - Class P for polynomial time: union of all $TIME(n^k)$
  - Problems solvable in $O(n^k)$ mem: $SPACE(n^k)$
    - Class PSPACE: union of all $SPACE(n^k)$ (superset of P)

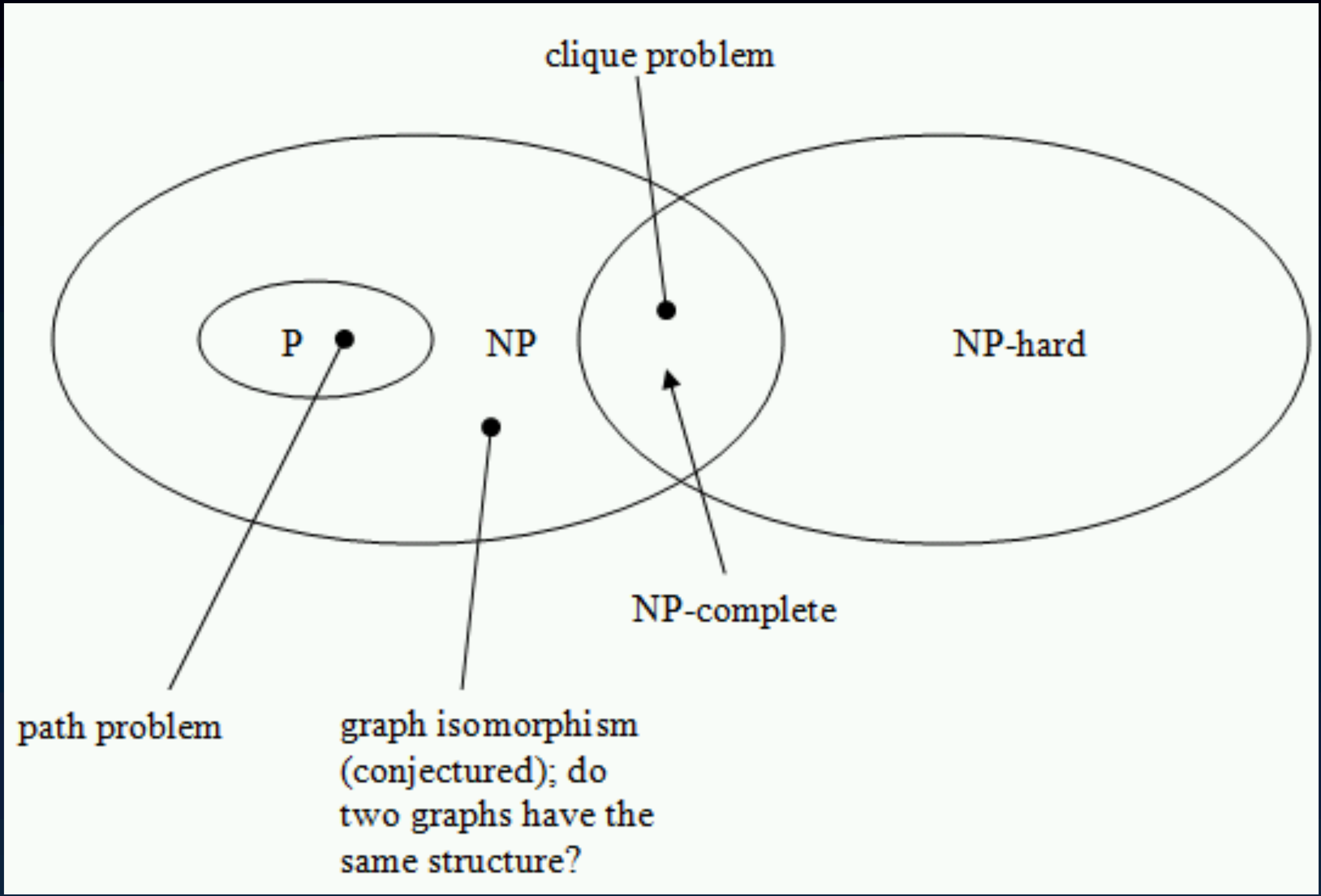
NP complexity class

- Solution can be verified in polytime
  - Still, solution hard to find
    - E.g., check is cipher key is correct
    - Finding the key is hard, verification is easy
- Hardest problems in NP: NP-complete
  - Traveling salesman, knapsack problem
  - Tetris, Super Mario, Candy Crush
  - All NP-complete problems are equally hard
  - Not all instances are hard: easy special cases

NP complexity class

- NP-Hard: at least as hard as NP-cmpl
  - What it takes to solve NP-hard also solves NP-complete
- Some problems are not in NP
  - Verify that no solution exists to a problem
  - Need to go through all possible inputs
- P vs NP: Is there a way to solve NP problems in polytime?
  - No proof yet: we believe P is not equal to NP

Complexity classes



Factoring

Problem in NP, but probably not NP-complete

- Find primes p,q given N=p*q
  - Widely used in RSA
  - Probably not NP-complete
- Different methods to factor integers
  - Naïve: try all numbers less than N
    - For n-bit integer N, the complexity is $O(2^n)$
  - Smarter Naïve: try all numbers up to $\sqrt{N}$
    - For n-bit integer N, the complexity is $O(2^{n/2}/n)$
  - GNFS: $O(e^{1.91n^{1/3}(\log n)^{2/3}})$
    - 1024-bit int: 2^{70} ops, 2048-bit int 2^{90} ops
    - 768-bit int factorized in 2009

Discrete log problem

Problem in NP, but probably not NP-complete

- Find y so that $g^y = x$ given base $g \in \mathbb{Z}_p^*$
- Math background (see HandoutA)
  - Group: a set of elements. E.g., $\mathbb{Z}_5^* = \{1,2,3,4\}$
  - Axioms: closure, associativity, identity, inv
    - $\mathbb{Z}_5^*$ is also commutative
  - Cyclic group: has at least 1 generator g so that $g^1, g^2, g^3, \ldots$ spans all elements of $\mathbb{Z}_p^*$
    - $\mathbb{Z}_5^*$ is cyclic with generators 2 and 3
- In general p is 1000s of bits long
  - $\mathbb{Z}_p^*$ contains about 2^m elements for m-bit p

Things can go wrong

- Factoring is easy if integer N is the product of powers of small primes
  - E.g., factoring $2^{800}*641*6700417$ is easy
  - Factoring $N=p^r q^s$ and $r > \log(p)$ is easy
- Implementation errors
  - Generating 128-bit RSA keys
  - Invoking secure libraries with small prime sizes
  - Using a 500-bit prime p in $\mathbb{Z}_p^*$

Hands-on exercises

- Pollard Rho method for factorization
- Comparison of factorization algorithms
- Computing discrete log (Pohlig-Hellman)

Reading for next lecture

- Aumasson: Chapter 10 until “Full Domain Hash Signatures”
- HandoutA on Canvas
- We will have a short quiz on the material