Impagliazzo's Five Worlds

Lecture 1

January 8, 2019

Outline

Worst case complexity
Average case complexity
Impagliazzo's five worlds

Impagliazzo's Five Worlds

Algorithmica
Heuristica
Pessiland
Minicrypt
Cryptomania

Worst-Case Complexity

P
NP

Problems

Add two n-digit numbers

Algorithms

Algorithm $A(x, y)$:
starting from rightmost digit, add digit by digit with carry
return the answer
example: $A(12345, 54321)$ returns 66666

Worst-Case Time Complexity

Adding two n-digit numbers using algorithm A takes n steps.
Worst-case time complexity of algorithm A for adding two n-digit numbers is $O(n)$

Languages

$L_{add} = \{(x, y, z) | z = x + y\}$
$(12345, 54321, 66666) \in L_{add}$
$(12345, 54321, 66667) \notin L_{add}$

Languages

Algorithm $B(x, y, z)$:
if $z = A(x, y)$ then output 1
else output 0
$(x, y, z) \in L_{add} \Leftrightarrow B(x, y, z) = 1$
We say that algorithm B recognizes L_{add}
Worst-case time complexity of algorithm B for adding is $O(n)$, where n is the input size:
$n = |(x, y, z)|$

P

We say that a language L belongs to the class **P** if we can find an algorithm B such that:
$x \in L \Leftrightarrow B(x) = 1$,
and worst-case time complexity of algorithm B is $O(n^c)$, where n is the input size, and c is a constant
example: $L_{add} \in \mathbf{P}$

P

example: $L_{mult} = \{(x, y, z) | z = xy\}$
$(2, 3, 6) \in L_{mult}$
$(2, 3, 7) \notin L_{mult}$
example: $L_{mult} \in \mathbf{P}$
two n digit numbers can be multiplied in $O(n^2)$ time using the well known multiplication algorithm

P

Problems in **P** can be efficiently *solved*

NP

Problems in **NP** can be efficiently *verified*

NP

We say that a language L belongs to the class **NP** if we can find an algorithm B such that:
if $x \in L$ then there exists a certificate u (u may be a possible solution of the problem corresponding to input x) of size $O(|x|^d)$ such that $B(x, u) = 1$,
and if $x \notin L$ then there there does not exist any certificate u of size $O(|x|^d)$ such that $B(x, u) = 1$,
and worst-case time complexity of algorithm B is $O(n^c)$, where n is the input size, and c, d are constants

$\mathbf{P} \subseteq \mathbf{NP}$

Algorithm $B(x, u)$
use the algorithm A for $L \in \mathbf{P}$:
return $A(x)$

Example of **NP**

Factorization problem:
given an integer n, find its prime factorization
example: for $n = 30$, its prime factorization is 2.3.5

Example of **NP**

$L_{fact} = \{(n, l, u) |$ there exists a prime p such that $p | n$, and $l \leq p \leq u\}$
example: $(30, 2, 6) \in L_{fact}$
example: $(30, 6, 10) \notin L_{fact}$

Polynomial-Time Reduction

If there exists a polynomial-time algorithm A such that:
$x \in L_1 \Leftrightarrow A(x) \in L_2$
then we say that L_1 reduces to L_2 in polynomial-time:
$L_1 \leq_p L_2$
example:
$L_{odd} = \{1, 3, 5...\}$
$L_{even} = \{2, 4, 6...\}$
$A(x) = x + 1$
$L_{odd} \leq_p L_{even}$
$L_{even} \leq_p L_{odd}$
$\leq_p$ is transitive:
if $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$ then $L_1 \leq_p L_3$

NP-Complete

If $L \in \mathbf{NP}$ and for all $L_1 \in \mathbf{NP}$:
$L_1 \leq_p L$
then L is called **NP-Complete**
Example: SAT:
find whether a Boolean formula in CNF form is satisfiable or not

NP-Complete

An **NP-Complete** problem L is called complete for **NP** because an algorithm for solving that problem (B) can be used to solve any problem $L_1 \in \mathbf{NP}$ (A is a polynomial-time reduction from L_1 to L)
Algorithm $C(x)$
return $B(A(x))$

Algorithmica

P $=$ **NP**

To prove the world to be Algorithmica, find a polynomial-time algorithm (B) for any **NP-Complete** problem (for example SAT)
the polynomial-time algorithm B can be used to make a polynomial-time algorithm for any problem in **NP**

Algorithmica

Problems that can be verified easily can also be solved easily

$\mathbf{P} = \mathbf{NP}$ implies no cryptography

Negligible Functions

A function $\epsilon(N) \rightarrow [0, 1]$ is called negligible if for every c and sufficiently large n:
$\epsilon(n) < 1/n^c$

One-way functions

A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ that can be computed in polynomial-time is called a one-way function if for every probabilistic algorithm A that runs in polynomial-time there exists a function $\epsilon(N) \rightarrow [0,1]$ which is a negligible function such that for every n:

$P_{x \in_R \{0,1\}^n, y=f(x)}[A(y) = x_1, f(x_1) = y] < \epsilon(n)$

One-way functions

Example:
$mult(x, y) = xy$
$mult(12345, 54321) = 670592745$ is easy to calculate
Finding the factors of 670592745 is difficult

One-way functions

One-way functions are used in cryptography

Average-Case Complexity

distP
distNP
sampP

Distributions

$D = \{D_n\}$ is a sequence of distributions
D_n is a distribution over $\{0,1\}^n$
Example: Uniform Distribution:
$U = \{U_n\}$
Each element in $\{0,1\}^n$ has probability $1/2^n$
$U_2 = \{((0,0),1/4),((0,1),1/4),((1,0),1/4),((1,1),1/4)\}$

Distributional Problem

A language L together with a distribution D: (L, D)
Example: (L_{add}, U)

Real-Life Distributions

Assumption: nature will not use a sophisticated supercomputer to give us problem instances

P-computable distributions

P-samplable distribution

P-Computable Distributions

Cumulative probability can be computed in polynomial time:
$\mu D_n(x) = \sum_{y \in \{0,1\}^n : y \leq x} P_{D_n}[y]$
$P_{D_n}[x] = \mu D_n(x) - \mu D_n(x-1)$
Example: U is **P**-computable:
$\mu U_n(x) = (x+1)/2^n$

P-Samplable Distributions

If a polynomial time probabilistic algorithm can produce samples from the distribution, then that distribution is called **P**-samplable distribution
Example: U is **P**-samplable:
Select each sample with uniform probability

P-Samplable Distributions

A **P**-computable distribution is also a **P**-samplable distribution
Sample x with probability:
$P_{D_n}[x] = \mu D_n(x) - \mu D_n(x-1)$

distP

$(L, D) \in$ **distP** if there exists an algorithm A for L and constants C and $\epsilon > 0$ such that for every n:

$E_{x \in_R D_n}[time_A(x)^\epsilon / n] \leq C$

$P \subseteq$ **distP**

If $time_A(x) = O(|x|^c)$, then select $\epsilon = 1/c$

distNP

$(L, D) \in$ **distNP** if $L \in$ **NP** and D is **P**-computable

distNP-Complete

(L, D) is **distNP**-Complete if $(L, D) \in$ **distNP**
and for all $(L_1, D_1) \in$ **distNP**:
$(L_1, D_1) \leq_p (L, D)$

sampNP

$(L, D) \in$ **sampNP** if $L \in$ **NP** and D is **P**-samplable

sampNP-Complete

(L, D) is **sampNP**-Complete if $(L, D) \in$ **sampNP**
and for all $(L_1, D_1) \in$ **sampNP**:
$(L_1, D_1) \leq_p (L, D)$

distNP $\subseteq$ sampNP

A **P**-computable distribution is also a **P**-samplable distribution

distNP $\subseteq$ sampNP

If (L, D) is **distNP**-complete then it is also **sampNP**-complete

Heuristica

$\mathbf{P} \neq \mathbf{NP}$
$\mathbf{sampNP} \subseteq \mathbf{distP}$

Heuristica

To prove $\mathbf{P} \neq \mathbf{NP}$:
prove a super-polynomial lower bound for some **NP-complete** problem
To prove **sampNP** $\subseteq$ **distP**:
find an algorithm for some **sampNP-complete** problem that efficiently solves almost all instances

Heuristica

Every problem in **NP** can be solved efficiently on almost all inputs
No cryptography

Pessiland

P $\neq$ **NP**
distNP $\not\subseteq$ **distP**
One-way functions do not exist

Pessiland

To prove $\mathbf{P} \neq \mathbf{NP}$:
prove a super-polynomial lower bound for some **NP-complete** problem
To prove $\mathbf{sampNP} \not\subseteq \mathbf{distP}$:
prove a super-polynomial average-case lower bound for some **sampNP-complete** problem

Pessiland

Problems that can be efficiently verified cannot be solved efficiently on many inputs
No cryptography

Minicrypt

$\mathbf{P} \neq \mathbf{NP}$
$\mathbf{distNP} \not\subseteq \mathbf{distP}$
One-way functions exist
Public key cryptography does not exist

Minicrypt

Prove a super-polynomial lower bound for some **NP-complete** problem
Prove a super-polynomial average-case lower bound for some **sampNP-complete** problem
Prove that no efficient algorithm exists for inverting some one-way function
Find polynomial-time algorithm for breaking public key cryptography (for example the factorization problem)

Cryptomania

$\mathbf{P} \neq \mathbf{NP}$
$\mathbf{distNP} \not\subseteq \mathbf{distP}$
One-way functions exist
Public key cryptography exists

Cryptomania

Prove a super-polynomial lower bound for some **NP-complete** problem
Prove a super-polynomial average-case lower bound for some **sampNP-complete** problem
Prove that no efficient algorithm exists for inverting some one-way function
Prove super-polynomial lower bounds for public key cryptographic functions (for example the factorization problem)

Conclusion

We do not know in which world we live

Most researchers believe that the world is Cryptomania

References

(1) R. Impagliazzo, A Personal View of Average-Case Complexity. In Structure in Complexity Theory Conference, pp. 134-147, 1995.
(2) L. A. Levin, Average case complete problems, SIAM J. Comput., 15(1):285-286, 1986.
(3) R. Imgagliazzo, L. A. Levin, No better ways to generate hard NP instances than picking uniformly at random, in FOCS, pp. 812-823, 1990.
(4) S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

Thank You