

Lecture 1

1 The Central Paradigm of Computer Science

- The central *paradigm* in computer science is that an algorithm **A** is good if:
 - **A** runs in *polynomial time* in the input size n .
 - That is, **A** runs in time $T(n) = O(n^k)$ for some constant number k .
 - $T(n) = 100n + 55$
 - $T(n) = \frac{1}{2}n^2 + 999 \log n$
 - $T(n) = 6n^7 + 900000n^2 - \sqrt{n}$
 - An algorithm is *bad* if it runs in exponential time.
 - $T(n) = 2^n + 100n^5$
 - $T(n) = 1.0000000001^n - n^3 - n$
 - An algorithm is *good* if it runs in *polynomial time* in the input size n .

e.g.

		Input Size n		
		10	100	1000
Runtime of Algorithm	n	10	100	1000
	n^2	100	10000	1000000
	2^n	10^3	10^{30}	10^{300}

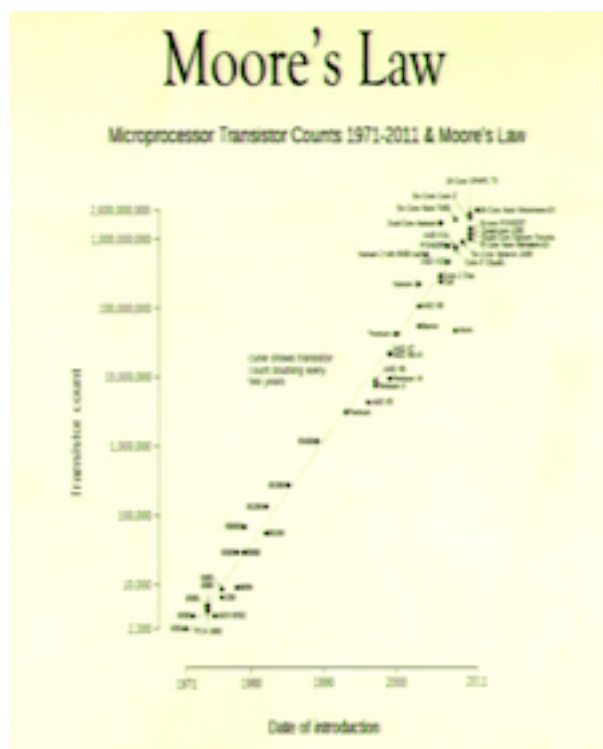
2 Good versus Bad (Algorithms)

- For example, consider the problem of sorting n numbers.
 - A Good Algorithm: **MergeSort** runs in time $O(n \cdot \log n)$
 - A Bad Algorithm: **BruteForce Search** runs in time $O(n \cdot n!) \gg 2^n$

3 An Equivalent Characterization

- This central *paradigm* has an equivalent formulation
 - A runs in *polynomial time* in the input size n .
 - The input sizes that A can solve, in a fixed amount T of time, *scales multiplicatively* with increasing computational power.

Input Sizes solved in Time T		
	Power = 1	Power = 2
n	T	$2T$
Runtime of Algorithm n^2	\sqrt{T}	$\sqrt{2} \cdot \sqrt{T}$
2^n	$\log T$	$1 + \log T$

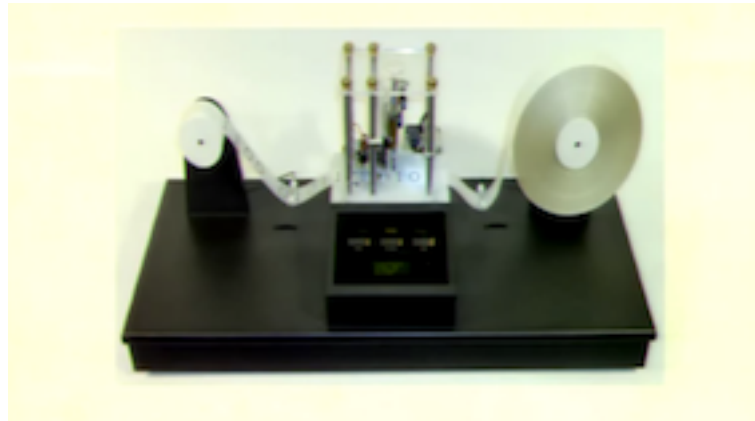


- Moore's Law: Computational power *doubles* roughly every two years.
 - Functional time algorithms will never be able to solve large problems.

- The practical implications are perhaps simpler to understand with this latter formulation.
- Thus, improvements in hardware will *never* overcome ***bad algorithm design***.
- Indeed, the current dramatic breakthroughs in computer science are based upon better (faster and higher performance) algorithmic techniques.

4 Robustness

- This measure of quality or "goodness" is ***robust***



- All reasonable models of algorithms are polynomial time equivalent.
 - Otherwise one model could perform, say, an exponential number of operations in the time another model took to perform just one.
- The standard formal model is the **Turing Machine**.

5 Cryptography (Just an example of the course)

- Alice wants to send Bob a message.
 But she is worried that Eve might intercept the message.
 She decides to encrypt the message M as $\hat{M} = f(M)$.
 Bob can then decrypt the message via $f^{-1}(\hat{M}) = f^{-1}(f(M)) = M$
 Eve cannot understand the message.
 Alice encrypts the message M with the (encryption) lock f .
 Bob decrypts the message \hat{M} with the (decryption) key f^{-1} .
 Because Eve does not have the key she cannot decipher the message.
 Two major problems occur.
- Problem one:
 - Eve might be able to break the code.
 - That is, given \hat{M} she may be able to reconstruct f and then f^{-1} .
 - Standard techniques for code-breaking include:
 - Frequency Analysis
 - *e.g.* "e" is the most common letter and "the" is the most common word in English.

→ Cribs.

- *e.g.* German weather reports were exploited to help decode messages from the "unbreakable" **Enigma Machine**.

- Problem Two:

- Alice and Bob need to agree on what the encryption code (lock) f is.
- But to do this they need to exchange a message discussing the code.

- Public-Key Cryptography


- In fact, Bob gives absolutely everyone a copy of the (encryption) lock.
- Everyone can send the message and Bob has the key.
- The lock is made public. This is called public-key cryptography.
- But this idea sounds completely crazy.
 - Doesn't this solution to Problem Two make Problem One inevitable?
 - That is, if Eve has the lock f then won't she use it to decode $f(M)$?
- No, not if f is hard to invert for anybody except Bob himself.
- But do such functions f that are hard to invert **exist**?
- Yes!

- RSA Encryption

1. Bob chooses two large prime numbers q_1, q_2 and a large number p that is co-prime to $(q_1 - 1) \cdot (q_2 - 1)$
2. Bob's public key is (p, n) where $n = q_1 \cdot q_2$
Encryption: $\hat{M} = M^p \bmod n$
3. Bob's private key is (q_1, q_2, x) where x is the inverse of p modular $(q_1 - 1) \cdot (q_2 - 1)$
Decryption: $M = \hat{M}^x \bmod n$
 - $p = 2^{16} + 1$
 - $q_1, q_2 = 2048$ bits
4. An Example
 - (a) Prime numbers q_1, q_2 and number p co-prime to $(q_1 - 1) \cdot (q_2 - 1)$
 $p = 7, q_1 = 3, q_2 = 11$ valid as $\gcd(7, 20) = 1$
 - (b) Public key (p, n) where $n = q_1 \cdot q_2$
 $n = 33$
 - (c) Private Key (q_1, q_2, x) where x is the inverse of $p \bmod (q_1 - 1) \cdot (q_2 - 1)$
 $x = 33$ as $3 \cdot 7 = 1 \bmod 20$
Public: $n = 33, p = 7$
Private $q_1 = 3, q_2 = 11, x = 3$


Public: $n = 33, p = 7$
Private: $q_1 = 3, q_2 = 11, x = 3$

Encryption f .
 $\hat{M} = M^p \bmod n$

 $M=2$

$$\begin{aligned}\hat{M} &= 2^7 \bmod 33 \\ &= 128 \bmod 33 \\ &= 29\end{aligned}$$

Decryption f^{-1} .
 $M = \hat{M}^x \bmod n$



$$\begin{aligned}M &= 29^3 \bmod 33 \\ &= (-4)^3 \bmod 33 \\ &= -64 \bmod 33 \\ &= 2\end{aligned}$$

- Is RSA Encryption Safe?
 - Public-key cryptography lies at the heart of the modern economy.
 - Financial Services
 - Online Shopping
 - Secure Messaging
 - So it is extremely important that the method is safe.
 - We claim it is because:
 - Bob has a good algorithm for decryption.
 - Eve only has a bad algorithm for decryption.
- Bob has a Good Decryption Algorithm
 - Initially, Bob can do the following in polynomial time.
 - Choose the primes q_1, q_2
 - Choose a number p that is co-prime with $(q_1 - 1) \cdot (q_2 - 1)$
 - Find x the inverse (Using Euclid's Algorithm) of p and $(q_1 - 1) \cdot (q_2 - 1)$
 - Using fast exponentiation, encoding and decoding is polynomial time.

Encryption: $\hat{M} = M^p \bmod n$
- Eve has a Bad Decryption Algorithm

Decryption: $M = \hat{M}^x \bmod n$

 - To decrypt Eve needs to find x the inverse of p and $(q_1 - 1) \cdot (q_2 - 1)$
 - She knows p , but does not know q_1, q_2
 - Instead, she only knows $n = q_1 \cdot q_2$
 - So to find q_1, q_2 , she needs to find the prime factorization of n .

- But it is believed that finding the prime factors of a b-bit number is hard.
 - Instead, the obvious algorithm attempts to divide n by each integer in the range $\{2, 3, \dots, 2^b\} \leftarrow 4096 \text{ bits}$
- In fact, if Eve could find x without q_1, q_2 , then she could use x to find q_1, q_2 . That is, she could factor n.
 - ⇒ Eve only has exponential time algorithm to decrypt.