

GEM 1501 Problem Solving With Computers

Lecture 11:

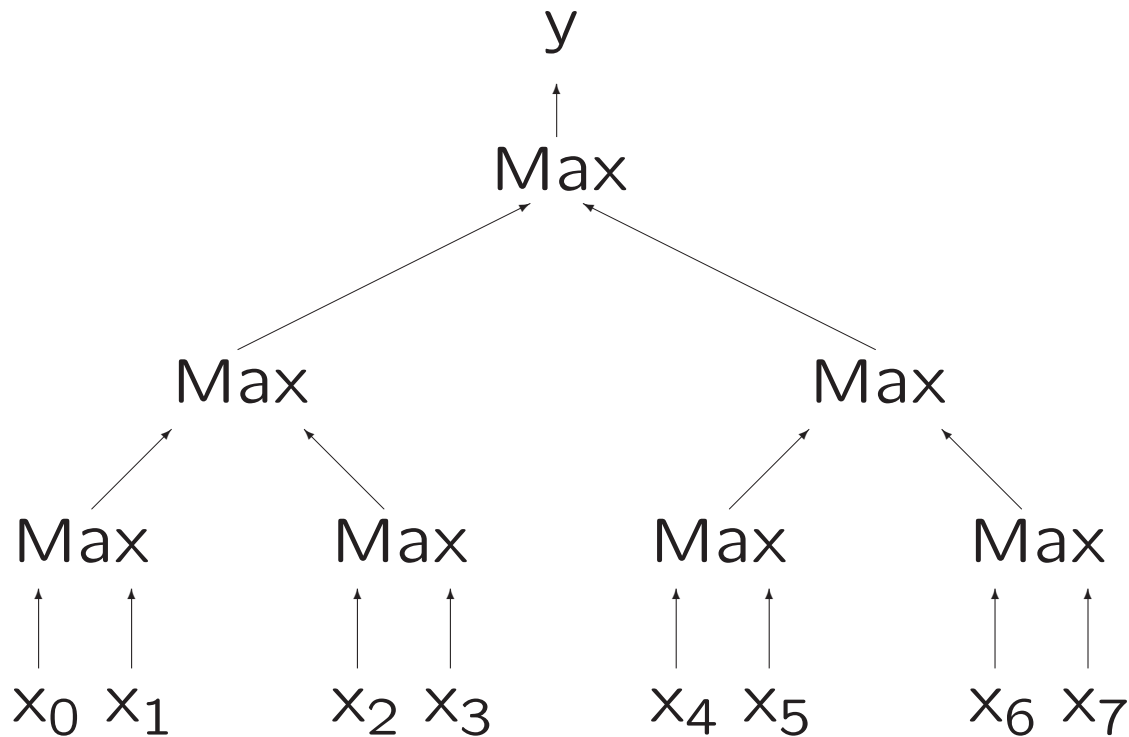
Probabilistic Algorithms

Frank Stephan

Summary of Previous Lecture

- Parallel Computing
- Parallel Sorting
- Weighted Averages
- Sequential Polynomial Space, Parallel Polynomial Time
- Nick's Class
- Coordinating Parallelism
- New Models of Computing

Parallel Divide and Conquer



Divide and Conquer implemented as a Circuit.

Each Maximum-Gate combines either two inputs or two subproblems below; $O(\log(n))$ parallel time.

Parallel Sorting and its Complexity

Several algorithms presented for parallel sorting. Primary criterion is parallel time, secondary criterion is number of processors. In order to combine both criteria, the product of time and number of processors has been considered.

Algorithm	Parallel Time	Processor Number * Time
Sequential Merge Sort	$O(n \log(n))$	$O(n \log(n))$
Parallel Merge Sort	$O(n)$	$O(n^2)$
Odd-Even Sort	$O(\log^2(n))$	$O(n \log^4(n))$
Optimal Sorting	$O(\log(n))$	$O(n \log(n))$

PSPACE and Parallelism

- PSPACE is the class of functions computable by a Turing machine using space $p(n)$ for some polynomial p where n is the length of the input.
- PSPACE can be solved in parallel polynomial time with exponentially many processors; the general algorithm says also how to distribute the work on the processors and how to connect them.
- Perhaps it can be done with less processors but this is unknown.

Nick's Class

- NC is named after Nicholas Pippenger
- Nick's class contains problems that can be solved very fast (some power of the logarithm of n) with only polynomially many processors.
- Problems in NC: Salary summation, finding maximum, sorting.
- $\text{LOGSPACE} \subset \text{NC} \subset \text{PSPACE}$ and $\text{NC} \subseteq \text{P}$.

Coordinating Parallism

Various processes share common resource.

One way to coordinate parallism is to use a semaphore in form of a queue. First member has access to resource, new requests are added at the end, first one is shifted out after usage has been completed.

Here example for use of hotel shower, atomic operations have (*). Semaphore variable "shared" is a Java Script array.

```
(*)    shared.push(guestname);    // Placing a request
        while (shared[0] != guestname)
            { do something else or wait; }
        use the hotel shower;      // Request is now in first place
(*)    shared.shift();            // Removing the used request
```

Lecture 11: Probabilistic Algorithms

- **Comparing Polynomials**
- The Class RP
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- Public Key Cryptography
- Interactive Proof Systems

Comparing Polynomials 1

- Given two descriptions f, g of polynomials as nested expressions of powers, products and sums like $(x^2 - 5)^8 + (x - 3)^{10} * x^6$ of some degree n .
- Question: Describe f and g the same function?
- Bring formulas into normal form $\sum a_m x^m$ and compare the coefficients.
- Fast if n is small.
But it needs 10000 terms to represent $(x - 3)^{10000}$. This is a problem for large n .

Comparing Polynomials 2

- Use the fact that whenever $f \neq g$ then $f(m) = g(m)$ for at most $n + 1$ numbers m where n is upper bound of degrees of f and g .
- Choose random number $m \in \{0, 1, \dots, 100 * n\}$.
- Test whether $f(m) = g(m)$. If $f \neq g$ then values are with 99% probability different.
- Problem: expressions like $(x - 3)^{10000}$ give numbers with thousands of digits.

Comparing Polynomials 3

- Use the additional fact that whenever $a \neq b$ then the remainders of a and b divided by p are different for most prime numbers p .
- Expressions like $28^{10000} \% 65537$ can be evaluated much faster and which much less space than 28^{10000} ; the first expression is 18914.
- Given f, g of degree up to n with coefficients a less than n' in the formulas, choose a prime number $p > 100 * (n + n')$.
- Choose a random number $m \in \{0, 1, \dots, p - 1\}$.
- Test whether the remainders of $f(m)$ and $g(m)$ modulo p are the same.

An example

Let $f(x) = x*(x-5)+5*x*(x-1)*(x-2)*(x-5)$
and $g(x) = x*(x-5)+8*x*(x-1)*(x-2)*(x-5)$.

Thus $f(x)-g(x) == 3*x*(x-1)*(x-2)*(x-5)$
and $f(m) != g(m)$ whenever m not in $\{0,1,2,5\}$.

Let $p = 257$.

Then $f(m) != g(m)$ whenever $m \% 257$ not in $\{0,1,2,5\}$.

This is true for m in $\{3,4,6,7,8,\dots,256\}$.

So if one draws m from $\{0,1,2,\dots,256\}$ then

$f(m) != g(m)$ with probability $253/257$ which is 0.9844357 .

A more realistic example

$$f(x) = (x^5+8*x+32)^{17}-x^{85}$$

$$g(x) = (x^{17}+20*x^3+x^2+131072)^5-x^{85}$$

(1) $m = 212$; computations modulo $p = 257$:

$$212^{85} == 94;$$

$$212^{5+8*212+32} == 117; f(212) == 236;$$

$$212^{17+20*212^3+212^2+131072} == 12; g(212) == 219.$$

(2) $m = 328967378$; computations modulo $p = 2147483647$:

$$328967378^{85} == 2133896847;$$

$$m^{5+8*m+32} == 1586613046; f(m) == 947557783;$$

$$m^{17+20*m^3+m^2+121072} == 2088754410; g(m) == 489300302.$$

Second computation more desirable since $131072 > 257$.

Nevertheless, in both cases $f \neq g$ is correctly established.

Probabilistic Algorithms

- Comparing Polynomials
- **The Class RP**
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- Public Key Cryptography
- Interactive Proof Systems

NP with Turing Machines

- Let A be an NP decision problem and M a nondeterministic Turing machine.
- On input x , M might from time to time have different choices how to continue the computation.
- If $x \in A$ then some choices result in a yes-computation.
- If $x \notin A$ then all choices result in a no-computation.

RP – Random Polynomial Time

- If a probabilistic Turing machine has two choices it tosses a coin and follows the coin's advice. Therefore the outcome of a computation has a certain probability.
- A problem A is in RP if there is a probabilistic Turing machine which needs on all inputs only polynomial time and has the following input-output behaviour.
- If $x \in A$ then a computation is with probability $2/3$ or more a yes-computation.
- If $x \notin A$ then every computation is a no-computation.

Some Remarks on RP

- Every problem in RP is in NP.
- Assume that A has a probabilistic Turing machine M which gives yes-computations with probability $q(n)$ for $x \in A$ and no-computations with probability 1 for $x \notin A$.

Then running M $t(n)$ times and saying “yes” iff M says “yes” at least once gives probability $1 - (1 - q(n))^{t(n)}$.

So one can “lift” the correctness probability to given values.

For example from $1/2$ to $1 - 2^{-n}$ by running the algorithm n times instead of one time.

- There is also a symmetric version, called BPP, where the probability of “yes” is at least $2/3$ if $x \in A$ and the probability of “no” is at least $2/3$ if $x \notin A$.

Further Examples

- Primality test: Let p be an n -digit number, decide whether p is a prime number.

Deterministic test: $O(n^6)$;

Randomized test: $O(n^4)$.

- Equality test for multi-variate polynomials.

Probabilistic Algorithms

- Comparing Polynomials
- The Class RP
- **Fast Probabilistic Pattern Matching**
- Introduction to Cryptography
- Public Key Cryptography
- Interactive Proof Systems

Fingerprints

- A fingerprint fpr is a function which maps every word w to a representative $\text{fpr}(w)$.
- $\text{fpr}(w)$ is short, say $\text{fpr}(w)$ consists of k bits for each word or sequence of words where k is a small constant.
- If v, w are different then $\text{fpr}(v)$ and $\text{fpr}(w)$ are different with probability $1 - 2^{-k}$.
- For each string a consisting of k bits there is a list of links to all documents which contain a word with fingerprint a .
- This method is also known as “hashing”.

Example

$fpr(w) = \text{length of word} + 10 * \text{number of vowels in word};$
 $fpr(\text{"the"}) == 13; fpr(\text{"mouse"}) == 35; fpr(\text{"house"}) == 35.$

Documents:	fpr of words:
(1) Tom is a cat.	13, 12, 11, 13.
(2) Jerry is a mouse.	25, 12, 11, 35.
(3) Tyke is a dog.	24, 12, 11, 13.
(4) Tom hates Jerry.	13, 25, 25.
(5) The mouse is in the house.	13, 35, 12, 12, 13, 35.

Index:

11: 1,2,3; 12: 1,2,3,5; 13: 1,3,4,5; 24: 3; 25: 2,4; 35: 2,5.

Search for documents with "Tom" and "Jerry".

$fpr(\text{"Tom"}) == 13; fpr(\text{"Jerry"}) == 25.$

Documents for 13 are 1,3,4,5; documents for 25 are 2,4.

So document 4 qualifies.

Discussion of Fingerprints

- Advantage: Search can be speeded up.
- Computing $\text{fpr}(w)$ for search word w and finding the list of all entries for w can be done in constant time.
- The links found lead with high probability to relevant pages and permit to inspect only a small fraction of the documents in the data base.
- Frequent problem: The function fpr does not use all possible strings with same probability. So some might be overcrowded and have very long lists of references to entries for several different words and others are not used at all.

Theory of Searching the Internet

- Search Machine reads all internet pages
- For each page, search machine stores the url, the text of the page and puts for each value $\text{fpr}(w)$ of a word or word-sequence w on the page a link from the index-entry for $\text{fpr}(w)$ to the url of the page.
- On search query w_1, w_2, \dots, w_n , search machine computes $\text{fpr}(w_1)$, $\text{fpr}(w_2)$, \dots , $\text{fpr}(w_n)$ and identifies the list of pages having this combination.
- Search machine displays the size of this list.
- Search machine checks for each page on the list individually whether the searched words and not only their fingerprints occur in this page.
- Thus less pages displayed than announced.

Reverse Engineering: Test this theory

Reverse Engineering: Finding out how something is programmed or implemented.

Theory tested on <http://www.altavista.com> on 30.03.2005 0900hrs

Randomly chosen strings are most likely to match other fingerprints and therefore more likely to produce wrongly announced pages

Search word	Announced and displayed number of pages	
James234	15	10
3124568	22	19
3932458	47	24

This is also an indication, not a proof of this theory.

Announced webpages might also be omitted for other reasons.

Test failed for <http://www.google.com.au>; Google only omits similar pages on first listing which show up when search is repeated.

Probabilistic Algorithms

- Comparing Polynomials
- The Class RP
- Fast Probabilistic Pattern Matching
- **Introduction to Cryptography**
- Public Key Cryptography
- Interactive Proof Systems

Basic Setting

- Alice sends message to Bob.
- Goal: Other people should not be able to understand the message.
- Step 1: Alice translates message into coded version.
- Step 2: Alice sends this version to Bob.
- Step 3: Bob retranslates message from coded version into plain form.

History

- 4000 years ago: An Egyptian nobleman had the inscriptions on his tomb cryptographically coded; unusual secret symbols were used.
- 2500 years ago: The Greek town Sparta introduces a cryptography machine which changes the order of the letters in a message.
- 2050 years ago: Roman emperor Caesar used substitution technique: Letters were shifted right, A replaced by D, B replaced by E and so on.
- 500 years ago: Alberti and later Vigenère develop a system which was believed to be unbreakable for more than 300 years.
- In 1854 Babbage and in 1863 Kasiski develop a method to break Vigenère's encryption method by analyzing the frequency of symbols.

Code Breaking

- A permutation of the letters preserves their frequency.
- Caesar's cryptography.
Text: The sky is blue. A mouse is in the house.
Code: Wkh vnb lv eoxh. D prxvh lv lq wkh krxvh.
- Combinations "wkh", "lv" and "rxvh" are frequent. Frequent codes like "lv" and "wkh" are likely to correspond to frequent English words like "is" and "the". Furthermore, in long texts, even single letters can be identified by their frequency.
- Possession of clear and encrypted message makes it more feasible to crack codes. Spies were stealing messages to get both versions. In World War II the Allied forces attacked some German military ships in order to eavesdrop how they transmit the information (position of ship, number of enemy planes and so on) and to get a clue what the key of the day might be.

Natural Languages

- Translating foreign languages: the less frequent and less related to common language, the better. The USA used for the communication units of their army often native Americans speaking rare languages without written documentation (grammar, dictionary, ...). They hoped that even if additional cryptography is broken, the enemy would not be able to figure out the meaning; this worked out in both world wars.
- Nevertheless, for most languages are translators available. Only rare nondocumented languages can be used for cryptographic purposes. Furthermore, technical terms are similar in many languages and can be recognized easily.
- According to newspaper reports, German telephone surveillance computers can analyse the major Indo-European languages and Arabic.

Doors and keys

- Locks at doors: only people with keys should come in.
- Security-firms can open doors in two minutes without leaving traces; often called if people locked the door with key inside.
- But everyday crime is dramatically reduced: much less theft than if everyone could go everywhere.
- Also: people who confuse doors do not enter a wrong flat by error.

Cryptography and Computer Security

- Cryptography and access-protocols make world-wide computer nets possible. It limits computer crime and damage from hackers.
- But there is no absolute security.
In World War II, Polish and English mathematicians cracked the German codes. Russians broke the Japanese codes.
After 1989, East-German officers told the public that most knowledge on West-Germany was obtained from the fact that West-German telephone calls were eavesdropped systematically.
From time to time, hacker break into sensitive computers of the Pentagon and other organizations which claim to have high security standards.
- Cryptography gives some but not complete security. The secret-services of the “global players” can either break it or bypass it: Common techniques are eavesdropping computer-screens, stealing computer equipment, bribing programmers and manipulating computer programs.

Random-Numbers

Alice and Bob have shared resource of random bits.

At those positions where the Code-String has a 1, the bit of the message is flipped.

Same procedure for coding and decoding.

Message: 11010010001000010000010000001

Code-String: 10101100110010110101101100111

Encoded Message: 01111110111010100101111100110

Encoded Message: 01111110111010100101111100110

Code-String: 10101100110010110101101100111

Decoded Message: 11010010001000010000010000001

For each message a new code-string is taken.

For security reasons, code-strings must be agreed on before parting and should never be used twice.

Evaluation

- Flipping Bits according to random string is most secure protocol but it has some problems.
- Problem 1: The code-strings must be interchanged in a secure manner, for example when Bob and Alice meet they interchange a CD-ROM with random bits on it.
- Problem 2: Random-bits are normally not generated by coin-tossing but by using algorithms which compute them from a small random-seed; this imperfectness permits to crack the code.

Private Key Cryptography

- Encoding and Decoding is done with a private key shared by sender, receiver and few (ideally 0) other people.
- Often encoding and decoding is done by same key as in the random-number coding example.
- Messages and keys are distributed through different channels; code books might contain large list of keys.
- Example: electronic banking in Singapore; bank sends one-time password by handphone and user accesses the bank account through the internet.

Breaking Codes with Information

- Attack on code possible if information on key can be derived.
- Pairs of unencrypted and encrypted messages helpful for code breaking. Coded messages must be kept secret to protect keys.
- Keys should not be shared with too many people.
- Quantity of encrypted messages should be limited.
- Communicating keys to new partners is difficult.

Probabilistic Algorithms

- Comparing Polynomials
- The Class RP
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- **Public Key Cryptography**
- Interactive Proof Systems

Public Key Cryptography

- Encoding and decoding done with different keys.
- Everyone can encode a message with public key of the receiver.
- Receiver can decode a message only with his private key.
- Encoding key can be made publically available. Some people have a public key for a cryptography program with name PGP on their web page. No limitation of key usage.

Extended Goals

- Traditionally: secret transmission of messages, privacy of communication.
- Detection of falsified messages.
- Avoiding theft or loss of messages.
- No need to protect public encryption key and pairs of uncoded/coded messages. No limitation on usage of keys as long as private key is not shared.

Encryption and Decryption

- Requirement 1: $Decr_A(Encr_A(M)) = M$.
- Requirement 2: $Encr_A(Decr_A(M)) = M$.
- Public: $Encr_A$; everyone can access this function (by knowing the public key).
- Secret: $Decr_A$; no one except Alice can do this.
- Important: having $Encr_A$ does not allow to deduce $Decr_A$.

Communication Protocol

- Bob sends message M to Alice.
- Bob codes M with his decryption algorithm and then with Alice's encryption algorithm:

$$N = Encr_A(Decr_B(M)).$$

- Bob transmits N to Alice.
- Alice decodes N with her decryption algorithm and then encrypts it with Bob's public encryption algorithm in order to make it readable:

$$M = Encr_B(Decr_A(N)).$$

- Why does Bob apply $Decr_B$ to his message? This prevents the production of a falsification M' by people who pretend to be Bob but who do not know his private decryption procedure. The falsifiers could not produce the text N' which translates into their intended falsified message M' .

RSA Cryptosystem

- Invented twice: first by Ellis, Cocks and Williamson for GCHQ (British government department for codes) and five years later by Rivest, Shamir and Adleman in public open research.
- Based on easy primality testing and hard factoring, assumption is that one cannot compute secret key from public key.
- The public key mainly consists of product $P * Q$ used to encrypt a message.
- The secret key consists of a pair of large prime numbers P and Q used to decrypt a message.

RSA Details

- Alice chooses large prime numbers P, Q, G such that G does not divide $P - 1$ and $Q - 1$. Furthermore, let K be a number such that

$$(K * G) \% ((P - 1) * (Q - 1)) == 1.$$

- Every transmitted message can be coded as a number smaller than $P * Q$, long messages are broken into short ones which are encrypted separately. A sufficiently short message is encrypted and decrypted by

$$\begin{aligned} Encr_A(M) &= M^G \% (P * Q); \\ Decr_A(N) &= N^K \% (P * Q). \end{aligned}$$

Alice publishes as encryption key the numbers G and $P * Q$ but not the factors P and Q .

- Assumption: It is so difficult to factor $P * Q$ that the code cannot be broken. Up to now no publically known code breaking algorithm exists and RSA cryptography is used in many applications to prevent or limit computer crime.

Example

- Alice selects some codes, here small numbers for clarity.
- Let $P = 11$, $Q = 13$. Then $P * Q$ is 143.
Let $G = 7$, 7 divides neither 10 nor 12.
Let $K = \min\{H : 7 * H \% ((P - 1) * (Q - 1)) == 1\}$, $K == 103$.
- $Encr_A(M)$ is $M^7 \% 143$, $Decr_A(M)$ is $M^{103} \% 143$.
- Each message is a number between 0 and 142.

Message	Encr(M)	Decr(M)
0	0	0
1	1	1
2	128	63
3	42	16
28	63	128
63	2	28
128	28	2
142	142	142

How to encrypt and decrypt fast

- How to compute $M^G \% (P * Q)$ fast?
- Starting with $N = 1$ and then G times doing $N = N * M \% (P * Q)$ is too slow.
- Assume $G == 11001$ in binary notation. Compute $M^G \% (P * Q)$ step per step per digit of G :

$$\begin{aligned}N_1 &= M \% (P * Q); \\N_{11} &= N_1 * N_1 * M \% (P * Q); \\N_{110} &= N_{11} * N_{11} \% (P * Q); \\N_{1100} &= N_{110} * N_{110} \% (P * Q); \\N_{11001} &= N_{1100} * N_{1100} * M \% (P * Q).\end{aligned}$$

Here N_w is $M^{binval(w)} \% (P * Q)$.

- So $O(\log(G))$ many operations instead of $O(G)$ many. The remainder is taken in each step in order to avoid too large numbers.

The Essence of RSA

- It is hard to factor a given large number.
- Factoring is believed not to be in P .
- RSA is based on the assumption that factoring is intractable.
- If one has a fast factoring algorithm, RSA would come apart!

Probabilistic Algorithms

- Comparing Polynomials
- The Class RP
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- Public Key Cryptography
- **Interactive Proof Systems**

Publication in Scientific Journals

- Author submits paper.
- Reviewer checks paper.
- Author has to convince reviewer that his results are correct.
- Author and Reviewer might interchange messages if the paper is not well-written or if the paper cannot contain all material.
- Reviewer should accept paper with high probability if the author can supply enough evidence for the correctness of his statements.
- Reviewer should reject paper with high probability if it is either incorrect or the author cannot provide enough evidence for its correctness or the author tries to cheat like falsifying or inventing experimental data.

Same example more formal

- Prover (Author) and Verifier (Reviewer)
- Author has unlimited resources (colleagues and students can work for him, large computers are in his laboratory, ...).
- Reviewer has only limited time since he has to deal with many submissions and other work is waiting as well.
- Reviewer can use random decisions on how to proceed (for deciding which parts of the work he wants to have explained in detail and so on).
- Due to this random decisions there is a probability that the author can succeed to make the reviewer to accept something or not.
- This probability should be high for correct and low for incorrect work.

Interactive Proof Systems

- IP denotes class of all sets which can be accepted by an interactive proof system.
- L is in IP iff there are a prover P and a verifier V such that:
 - For all $x \in L$, P can convince V with probability $2/3$ or more to accept and output the statement “ $x \in L$ ”.
 - No prover P' (whether P or someone else) can make V to output a wrong statement with probability $1/3$ or more.
 - The whole time used for communication and V 's computations is polynomially bounded; the computation time of P is not bounded.
- Every L in NP is also in IP. If $x \in L$ then x has some solution and the prover can provide the solution to the verifier which can check it before outputting “ $x \in L$ ”.

The Power of IP

- (1) $IP = PSPACE$. So computing with polynomial space and having a polynomially bounded interactive proof system is equivalent.
- (2) If L is in IP so is its complement. Thus there is a proof system (P, V) with the following properties:
 - For all $x \in L$, P can convince V with probability $2/3$ or more to output “ $x \in L$ ”.
 - For all $x \notin L$, P can convince V with probability $2/3$ or more to output “ $x \notin L$ ”.
 - No prover P' (whether P or someone else) can make V to output a wrong statement with probability $1/3$ or more.
 - The whole time used for communication and V 's computations is polynomially bounded; the computation time of P is not bounded.

Summary: Probabilistic Algorithms

- Comparing Polynomials
- The Class RP
- Fast Probabilistic Pattern Matching
- Introduction to Cryptography
- Public Key Cryptography
- Interactive Proof Systems

Next Week

- Midterm Test
- Software Engineering