Učební texty k státní bakalářské zkoušce
Obecná informatika
Algoritmy a datové struktury

8. června 2011

3 Algoritmy a datové struktury

Požadavky

- Časová složitost algoritmů, složitost v nejhorším a průměrném případě.
- Třídy složitosti P a NP, převoditelnost, NP-úplnost.
- Metoda ,,rozděl a panuj" - aplikace a analýza složitosti.
- Binární vyhledávací stromy, vyvažování, haldy.
- Hašování.
- Sekvenční třídění, porovnávací algoritmy, přihrádkové třídění, třídící sítě.
- Grafové algoritmy - prohledávání do hloubky a do šířky, souvislost, topologické třídění, nejkratší cesta, kostra grafu, toky v sítích.
- Tranzitivní uzávěr.
- Algoritmy vyhledávání v textu.
- Algebraické algoritmy - DFT, Euklidův algoritmus.
- Základy kryptografie, RSA, DES.
- Pravděpodobnostní algoritmy - testování prvočíselnosti.
- Aproximační algoritmy.

3.1 Metoda rozděl a panuj – aplikace a analýza složitosti

TODO: všechno

3.2 Základy kryptografie, RSA, DES

Základy kryptografie[1]

Definice *(Kryptografický systém (!))*
Prostor otevřených zpráv M, šifrovaných zpráv C, šifrovacích a dešifrovacích klíčů K a K'. Efektivní generování klíčů $G : N \to K \times K'$, šifrování $E : M \times K \to C$, dešifrování $D : C \times K' \to M$.

- **Symetrické** (sdílený klíč $k_e = k_d$) rychlé, krátké klíče, potreba menit klice a bezpecne si je vymenit
- **Asymetrické** (veřejný klíč $k_e \neq k_d$) delsi klice a pomalejsi nez symetrické, neni potřeba tajná výměna, neni potřeba tak často měnit klíče

Definice *(Nahodné generátory)*
Používají se pro generování klíčů pro šifry (např RSA) a v proudových šifrách.

- **HW** zařízení často založená na jevech generujících statisticky náhodné "šumové" signály, například z tepelného šumu polovodiče.
- **SW** jsou založeny na pozorování jevů v počítači z hlediska programu náhodných, často z uživatelského vstupu (např. PuTTYgen používá pro generování RSA klíče přejízdění myší).
- **Pseudonáhodné** jsou deterministické programy generující posloupnost čísel pokud možno nerozlišitelnou od náhodné.
  - př. kongruenční generátor: $X_{n+1} = (aX_n + c) mod m$
  - používají se v proudových šifrách

[1] sestaveno podle vrazedneho zkouseni Jaghobem

Definice *(Hashovaci funkce)*
Funkci $h : U \to \{0, 1, \ldots, m-1\}$ nazýváme **hašovací funkcí**.[2]

Požadavky:

- Rovnoměrné a náhodné rozložení hodnot
- Odolnost na kolize (výpočetně složité najít pro $x \neq x$ $h(x) = h(y)$)
- Jednosměrná funkce (výpočetně složité najít y k x pro $h(x) = y$)
- Efektivní algoritmus

Využití: CRC (kontrolní součet), ukládání hesel (MD5,SHA) ...

Definice *(Model utocnika podle Doleva a Yao)*

- Může získat libovolnou zprávu putující po síti
- Je právoplatným uživatelem sítě a tudíž může zahájit komunikaci s jiným uživatelem
- Může se stát příjemcem zpráv kohokoliv
- Může zasílat zprávy komukoliv zosobněním se za jiného uživatele
- Neumi rozume resit NP-uplne problemy (ani slozitejsi)[3]
- Bez správného klíče nemůže nalézt zprávu k šifrované zprávě a nemůže vytvořit platnou šifrovanou zprávu z dané zprávy, vše vzhledem k nějakému šifrovacímu algoritmu

Definice *(Cile utoku)*

důvěrnost dat uživatel může určit kdo má data vidět, a systém skutečně dovolí pracovat s daty pouze povoleným uživatelům

celistvost dat možnost podstrčení falešných dat

dostupnost systému *DoS (Denial of Service)*

Příklad
Ukazku pouziti nejakeho sifrovaciho protokolu (zvolil jsem kombinace symetricka sifra sifrovani, asymetricka predani klicu k symetricke).

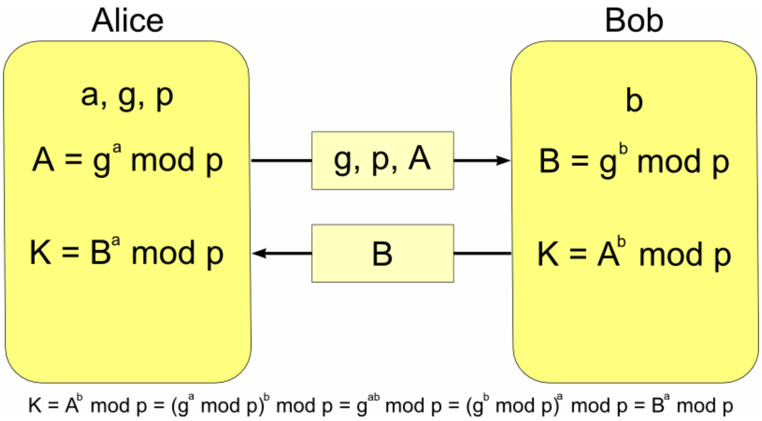
TODO

Definice *(protokol Diffie-Hellman)*

- Diffie-Hellman výmena klícu je kryptografický protokol, který umožnuje navázat bezpecné spojení. Pro bezpecné spojení je potreba si vymenit klíc k symetrické šifre pres ješte nezabezpecený kanál. Práve tento protokol to umožnuje aniž by byl klíc jednoduše poslán v otevrené forme.
- Alice si vymyslí velké prvocíslo p, generátor g konečné grupy $G = (Z_p^*, \cdot)$ a $a \in [1, p-1)$ vypocte A pošle Bobovi [g,p,A], Bob vypocte B a pošle ho Alici oba si vypocítají K $\Rightarrow$ muzou zacit symetricky sifrovanou komunikaci

- Puvodne nezabezpecoval autentifikaci ucastniku = nachylny k utoku man-in-the-middle. Man-in-the-middle muze vytvorit komunikaci s dvema ruznymi Diffie-Hellman klici, jeden s Alici a druhej s Bobem, a pak se tvarit jako Alice k Bobovi a obracene, treba pomoci dekodovani a rekodovani zprav mezi nimi. Nejaka metoda autentifikace mezi temito osobami je nutna.
- Problému nalezení císla a ze znalosti ga mod p se ríká problém diskrétního logaritmu. Tento problém je stále považován za velmi obtížný.

[2] viz otázku Hašování
[3] tzn. i slabší: Nemůže odhadnout náhodné číslo z dostatečně velkého prostoru



Obrázek 1: D-H protokol

RSA (Rivest-Shamir-Adleman)

Asymetrická šifra (různé klíče pro šifrování a dešifrování), použitelná jako šifra s veřejným klíčem. Kryptoschéma je založeno na Eulerove formuli.

Alice a Bob se verejne dohodnou na hranici N a chtejí si vymenovat tajné zprávy $0 \leq m < N$.
Inicializace:

1. vybrat dvě dostatečně velká prvočísla p, q tak aby $n = p \cdot q < N$
2. Alice spočítá $\varphi(n) = (p-1) \cdot (q-1)$
   (Eulerova funkce $\varphi(n)$ je počet čísel menších než n, která jsou s n nesoudělná)
3. vybrat e takové, že $1 < e < \varphi(n)$ a e je nesoudělné s $\varphi(n)$
   – dvojice (n, e) bude *veřejný klíč (public key)*
4. vybrat d tak, aby
   $$d \cdot e \equiv 1 \mod \varphi(n)$$
   takové d lze najít rozšířeným euklidovým algoritmem
   – dvojice (n, d) bude *dešifrovací klíč (private key)*

Šifrování:

1. Alice posílá public key Bobovi (čísla n a e), nechává si private key
2. Bob chce Alici poslat zprávu m tak spočítá :
   $$c = m^e \mod n$$
3. Bob odešle c Alici

Dešifrování:

1. Alice přijala c
2. Spočítá:
   $$m = c^d \mod n$$

Šifra (to, že to vůbec funguje, tedy, že $m = (m^e)^d$) se opírá o několik netriviálních vět algebry...

- Pro reálné použití císla približne 100 až 200 bitu. Klíc e volíme jako prvocíslo vetší než $(p-1)$ a $(q$–1). Hranice bezpecnosti pro modul n je $N = 1024$ bitu, rozumné 1500 bitu, lépe 2048
- Není známa metoda vedoucí k rozbití tohoto algoritmu
- Slabostí je hypotetická možnost vytvorit elektronický podpis zprávy bez znalosti dešifrovacího klíce na základe zachycení vhodných predchozích zašifrovaných zpráv.
- například SSH protokol používá RSA klíče

3.3 Pravděpodobostní algoritmy – testování prvočíselnosti

Pravděpodobnostní (náhodnostní) algoritmy jsou nedeterministické algoritmy, které se snaží najít řešení rychleji nebo řešení těžko řešitelných problémů, často tzv. NP-úplných problémů. Pravděpodobnostní algoritmus se může náhodně rozhodovat mezi různými možnostmi jak pokračovat. Pro stejný vstup může dávat takový algoritmus různé výsledky, které mohou být dokonce nesprávné. Mnohdy se tedy na daném vstupu spustí pravděpodobnostní algoritmus vícekrát, aby se s větší pravděpodobností dospělo ke správnému výsledku. [edit] Motivation

As a motivating example, consider the problem of finding an ' a ' in an array of n elements.

Input: An array of n elements, in which half are ' a 's and the other half are ' b 's.

Output: Find an ' a ' in the array.

We give two versions of the algorithm, one Las Vegas algorithm and one Monte Carlo algorithm.

Las Vegas algorithm:

¡source lang="pascal"¿ findingA$_LV(arrayA, n)begin$

repeat Randomly select one element out of n elements. until 'a' is found

end ¡/source¿

This algorithm succeeds with probability 1, but the running time is random and its expectation is upper-bounded by Failed to parse (Missing texvc executable; please see math/README to configure.): O(1) .

Monte Carlo algorithm: ¡source lang="pascal"¿ findingA$_MC(arrayA, n)begin$

i=1 repeat Randomly select one element out of n elements. i = i + 1 until i=k

end ¡/source¿ If an ' a ' is found, the algorithm succeeds, else the algorithm fails. After k times execution, the probability of finding an ' a ' is:

Failed to parse (Missing texvc executable; please see math/README to configure.): Pr[find_'a']=1-(1/2)^k

This algorithm does not guarantee success, but the run time is fixed. The selection is executed exactly k times, therefore the runtime is Failed to parse (Missing texvc executable; please see math/README to configure.): O(k) .

Randomized algorithms are particularly useful when faced with a malicious "adversary" or attacker who deliberately tries to feed a bad input to the algorithm (see worst-case complexity and competitive analysis (online algorithm)) such as in the Prisoner's dilemma. It is for this reason that randomness is ubiquitous in cryptography. In cryptographic applications, pseudo-random numbers cannot be used, since the adversary can predict them, making the algorithm effectively deterministic. Therefore either a source of truly random numbers or a cryptographically secure pseudo-random number generator is required. Another area in which randomness is inherent is quantum computing.

In the example above, the Las Vegas algorithm always outputs the correct answer, but its running time is a random variable. The Monte Carlo algorithm(related to the Monte Carlo method for simulation) completes in a fixed amount of time (as a function of the input size), but allow a small probability of error. Observe that any Las Vegas algorithm can be converted into a Monte Carlo algorithm (via Markov's inequality), by having it output an arbitrary, possibly incorrect answer if it fails to complete within a specified time. Conversely, if an efficient verification procedure exists to check whether an answer is correct, then a Monte Carlo algorithm can be converted into a Las Vegas algorithm by running the Monte Carlo algorithm repeatedly till a correct answer is obtained.

[edit] Varianty pravděpodobnostních algoritmů

* Výpočetní strom je binární, v každém uzlu se provede hod mincí. * V každém výpočetním uzlu je definováno pravděpodobnostní rozložení na hranách. * Na začátku se vybere náhodně deterministický algoritmus, který provede výpočet.

Všechny tři varianty jsou ekvivalentní. [edit] Poznámky

Pravděpodobnostní algoritmy jsou většinou jednoduché, avšak analýza jejich časové složitosti je často náročná.

http://en.wikipedia.org/wiki/Randomized$_algorithm[edit]prvočístelnost$

* http://cs.wikipedia.org/wiki/Miller* http://en.wikipedia.org/wiki/Miller* http://en.wikipedia.org/wiki/Compo
$http : //en.wikipedia.org/wiki/Witness$

3.4 Aproximační algoritmy

NP-hard problems vary greatly in their approximability; some, such as the bin packing problem (baleni batohu), can be approximated within any factor greater than 1 (such a family of approximation algorithms is often called a polynomial time approximation scheme or PTAS). Others are impossible to approximate within any constant, or even polynomial factor unless P = NP, such as the maximum clique problem (hledani maximalni kliky).

* http://en.wikipedia.org/wiki/Approximation$_algorithm*http : //en.wikipedia.org/wiki/Bin_packing_problem$