Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013. "Investește în oameni!"

Cap. 3. Fundamente. Fundamente Matematice si Probleme Computationale. Elemente de Teoria Informatiei. Elemente de Teoria Probabilitatii. Elemente de Teoria Numerelor: Grupul Zn, Teoremele lui Euler si Fermat, Generatori, Congruente polinomiale, Reziduuri cvadratice, Simboluri Legendre si Jacobi. Calculul operatiilor elementare. Calculul radicinilor patrate si de ordin k. Problema factorizarii si a logaritmului discret. Generatoare de numere prime. Curbe Eliptice.*

Magnitudini

- ***Cicluri de tact într-un sistem la 1000 GHz în 1000 de ani***:
  $1000*1000*10^9*3*10^7=3*10^{22}\approx30000000000000000$
- ***Vârsta sistemului nostru solar în secunde:***
- $1,89*10^{17}\approx18900000000000000$
- ***Electroni în univers:***
  $8,37*10^{77}\approx837000000000000000000000000000000000000000000000000$
  00000000000000000000000

- ***Numarul de pasi la complexitatea*** O(1) pentru n=1024

  1

- ***Numarul de pasi la complexitatea*** $O(n^2)$ pentru n=1024

  1048576

- ***Numarul de pasi la complexitatea*** $O(2^n)$ pentru n=1024

$1,79x10^{308}=179000000000000000000000000000000000000000000000000000000$
$00000000000000000000000000000000000000000000000000000000000000000000$
$00000000000000000000000000000000000000000000000000000000000000000000$
$00000000000000000000000000000000000000000000000000000000000000000000$
$00000000000000000000000000000$

O cheie RSA de 2048 de biti (cheie medie spre mica)

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357

Recompense oferite de RSA pentru factorizarea unor intregi

RSA-768	232	768	$50,000 USD
RSA-896	270	896	$75,000 USD
RSA-1024	309	1024	$100,000 USD
RSA-1536	463	1536	$150,000 USD
RSA-2048	617	2048	$200,000 USD

Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.
"Investește în oameni!"

Teoria Probabilitatilor

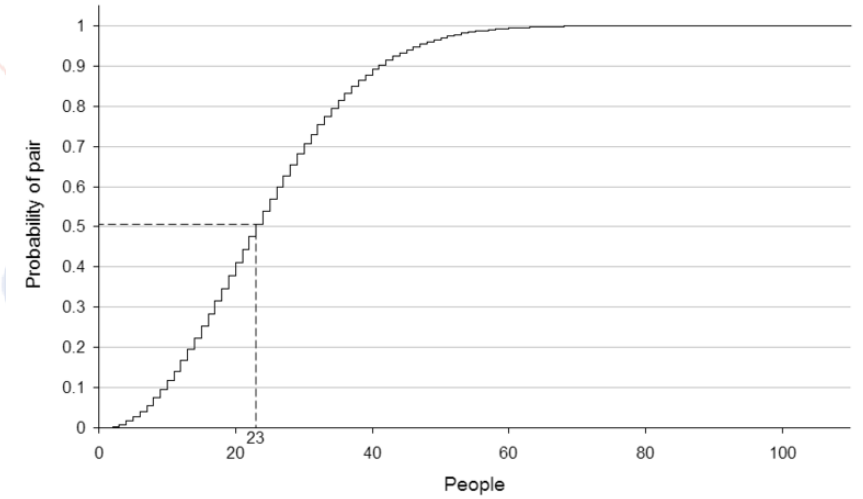
- Evenimentul este rezultatul unui experiment
- Evenimentele pot fi: independente, mutual exclusive sau complementare

Paradoxul zilelor de nastere

- Exercitiu: un programator doreste sa construiasc o tabela (hashtable) in care valori sunt retinute indexat dupa chei pe k biti. Fiecare cheie este generata aleator (sau are la baza un hash al valorii). Dupa cate valori introduse in tabela putem astepta o coliziune cu probabilitate mai mare de ½ ?
- Reformulare: Intr-o urma sunt m bile, care este probabilitatea ca dupa n extrageri (cu reintroducere) sa exista cel putin 1 coliziune
- Daca definim $m^{(n)} = \frac{m!}{n!} = m(m-1)(m-2)...(m-n+1)$
- Probabilitatea de cel putin 1 coliziune este $1 - \frac{m^{(n)}}{m^n}$
- Se poate demonstra ca daca $m \to \infty$ atunci numarul de extrageri pana la o coliziune este in medie $\sqrt{\frac{\pi m}{2}}$

Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013. "Investește în oameni!"

- Problema anterioara se numeste paradoxul zilelor de nastere deoarece, in mod oarecum surprinzator, probabilitatea ca intr-o camera cu 23 de persoane sa existe cel putin 2 persoane nascute in aceeasi zi este aprox. ½ iar intr-o camera cu 100 de persoane 0.999999
- Evolutia probabilitatii de coliziune la zile de nastere (grafic din wiki)



UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE AMPOSDRU
Fondul Social European POSDRU 2007-2013
Instrumente Structurale 2007-2013
ORGANISMUL INTERMEDIAR REGIONAL PENTRU POS DRU REGIUNEA BUCUREŞTI ILFOV

Proiect cofinanțat din Fondul Social European prin
Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.
"Investește în oameni!"

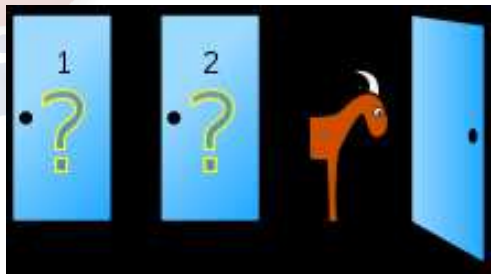
Teorema lui Bayes

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.
"Investește în oameni!"

Aplicatii are teoremei lui Bayes

- Aplicarea teoremei lui Bayes este elementara in probleme din categoria: daca s-a intamplat P atunci care e probabilitatea sa se intample Q
- Atentie sporita la faptul ca rezultatul este de foarte multe ori contraintuitiv (de ex. paradoxul cutiilor, Monty Hall)



Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013. "Investește în oameni!"

Teoria Complexităţii

- Obiectiv: clasificarea problemelor in funcţie de resursele necesare calculului
- Răspunde la întrebarea: Cât timp de calcul şi câtă memorie este necesară pentru rularea algoritmului pe măsură ce dimensiunea datelor de intrarea creşte?
- Caracterizarea nu trebuie sa depindă de un model computaţional (de sistemul pe care rulează algoritmul) – ea trebuie să ofere un model abstract, universal aplicabil. Altfel spus, caracterizarea trebuie sa ofere o masura pentru dificultatea intrinsecă a unei probleme.

Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013. "Investește în oameni!"

Ce este o problemă

- O problemă este o mulțime nevidă de întrebări între care există o relație. Pot fi una sau mai multe întrebări dar este obligatoriu ca ele să aibă o dimensiune finită.
- Exemplu de problemă. Factorizarea unui întreg: avînd un întreg n care este produsul a exact două numere prime să se afle numerele prime p, q care înmulțite dau n.

Ce este un algoritm algoritm

- Un algoritm este un set bine definit de pasi pentru rezolvarea unei probleme. Atfel spus, un algoritm este ansamblul de paşi prin care un set de date de intrare este transformat intr-un set de date de ieşire în scopul rezolvării unei probleme.
- Exemplu de algoritm. Pentru rezolvarea problemei factorizării întregului de mai sus se poate folosi urmatorul algoritm: Pentru toți întregii i de la 2 la $n/2$ verifica daca i divide n.

Caracterizarea complexității

- Dimensiunea intrării n
- Complexitate de timp $T=F(n)$ – numărul de paşi ai algoritmului necesari pentru a rezolva problema ca funcţie de dimensiunea intrării.
- Complexitate de spatiu $S=F(n)$ – memoria necesară pentru a rezolva problema ca funcţie de dimensiunea intrării.
- Observaţie: Prin complexitatea unei probleme (adeseori folosim ca sinonim dificultatea problemei) se înţelege complexitatea de timp şi spaţiu a celui mai eficient algoritm pentru problema respectivă.

UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE AMPOSDRU
Fondul Social European POSDRU 2007-2013
Instrumente Structurale 2007-2013
ORGANISMUL INTERMEDIAR REGIONAL PENTRU POS DRU REGIUNEA BUCUREŞTI ILFOV
Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.
"Investește în oameni!"

Notatii Asimptotice (Indicatori de Performanta)

- Limita asimptotica superioara (Semnificaţie intuitivă: asimptotic f nu creşte mai repede ca g eventual multiplicat cu o constantă)

$$f(n) = O(g(n)) \Leftrightarrow \exists c, n_o \; a.i. 0 \le f(n) \le c \cdot g(n) \forall n \ge n_0$$

- Limita asimptotica inferioara (Semnificație intuitivă: asimptotic f creşte mai repede ca g eventual multiplicat cu o constantă)

$$f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_o \; a.i. 0 \le c \cdot g(n) \le f(n), \forall n \ge n_0$$

- Limita asimptotica stransa (Semnificație intuitivă: o funcție care este atît limită inferioară cît şi superioară)

$$f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_o \; a.i. c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \forall n \ge n_0$$

Notatii Asimptotice (Indicatori de Performanta)

$$f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_o \; a.i. c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \forall n \geq n_0$$

- Indicatorul o si ω sau limitele superioare şi inferioare absolute (Semnificație intuitivă: asimptotic f creşte/nu creşte mai repede ca g multiplicat cu orice constantă)

$$f(n) = o(g(n)) \Leftrightarrow \exists n_o \; a.i. \forall c \;\; 0 \leq f(n) < c \cdot g(n) \forall n \geq n_0$$

$$f(n) = \omega(g(n)) \Leftrightarrow \exists n_o \; a.i. \forall c \;\; 0 \leq c \cdot g(n) < f(n) \forall n \geq n_0$$

- Atenție!!!: $f = F(g(n))$ nu se citeste: "f egal F de $g(n)$", corect este "f este de ordinul F al lui $g(n)$" sau mai simplu "f este $F(g(n))$" unde F este oricare din indicatorii Ω, Θ, O, o. Intuitiv semnul "=" nu are semnificația semnului egal, în acest caz este echivalent cu $\in$

Relaţii elementare între indicatori (Se pot demonstra direct din definiţile acestora).

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$$

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

$$f(n) = O(h(n)) \wedge g(n) = O(h(n)) \Rightarrow (f+g)(n) = O(h(n))$$

$$f(n) = O(h(n)) \wedge g(n) = O(i(n)) \Rightarrow (f \cdot g)(n) = O(h(n) \cdot i(n))$$

$$f(n) = O(f(n)) - reflexivitate$$

$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n)) - tranzitivitate$$

Relații utile (frecvent utilizate)

- Pentru polinoame: limita asimptotică strânsă a unui polinom este dată de termenul de rang maxim

$$f(n) = a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0 \Rightarrow f(n) = \Theta(n^k)$$

- Pentru probleme de combinatorică

$$n! = o(n^n) \wedge n! = \Omega(2^n)$$

- Pentru oricare ε, c a.i. $0<\varepsilon<1<c$

$$1 < \ln\ln n < \ln n < e^{\sqrt{(\ln n)(\ln\ln n)}} < n^{\varepsilon} < n^{c} < n^{\ln n} < c^{n} < n^{n} < c^{c^{n}}$$

Scurta clasificare a algoritmilor in baza complexitatii computationale

- Constant $O(1)$
- Logaritmic $O(\log n)$ - Observatie $\log_c n = \Theta(\lg n), \forall c > 0$
- Polilogaritmic $O((\log n)^c)$
- Fractional $O(n^c)$ $0<c<1$
- Linear $O(n)$
- Linear logaritmic $O(n \log n)$
- Patratic (sau Cuadratic) $O(n^2)$
- Cubic $O(n^3)$
- Polinomial $O(n^c)$, $c>1$ (patratic si cubic sunt polinomiale)
- Superpolinomial $O(c^{f(n)})$ (c – constanta, f – nu este constant dar este mai mic de $O(n)$)
- Exponential $O(c^{f(n)})$ (c – constanta, f – polinom de grad 1)
- Factorial (sau Combinatorial) $O(n!)$
- Dublu Exponential $O(2^{c\wedge n})$

De ce este importanta cunoasterea complexitatii?

- Pentru a stii daca ne putem astepta la un rezultat in timp util şi dacă avem suficiente resurse de memorie.
- Exemplu: Fie doi algoritmi de complexitate O(n) respectiv $O(2^n)$. Dublarea lui n duce la dublarea timpului de calcul pentru primul algoritm in timp ce cresterea lui n cu un singur bit duce la dublarea timpului de calcul pentru cel de-al doilea.

Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.
"Investește în oameni!"

Teoria Informatiei

Definiția 3.4. (Entropia) Pentru o variabilă X ce poate lua aleator valorile $x_1, x_2, ..., x_n$ cu probabilitățile $p_1, p_2, ..., p_n$, i.e. $P(X = x_i) = p_i$, definim entropia lui X ca fiind $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$ (se consideră $p_i \log_2 p_i = 0$ dacă $p_i = 0$).

Teoria Numerelor

- Fundament pentru aproape toate criptosistemele cu cheie publica

- **Definitie:** Definim $Z_n = \{0,1,2,3,...,n-1\} = \{x \in N \mid 0 \le x < n\}$ ca fiind multimea resturilor modulo n.
- **Definitie:** Definim $Z_n^* = \{x \in Z_n \mid cmmdc(x,n) = 1\}$ ca fiind multimea intregilor din Z_n relativ primi la n. In particular pentru un numar prim n avem $Z_p^* = \{1,2,3...,p-1\}$
- **Remarca:** $(Z_n^*, \cdot)$ formeaza un grup abelian (simbolul $\cdot$ denota inmultire) deoarece urmatoarele proprietati sunt satisfacute:

1) Legea este asociativa pentru ca $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c \bmod n$
2) Legea este comutativa pentru ca $a \cdot b \equiv b \cdot a \bmod n$
3) Exista element neutru deoarece $a \cdot 1 \equiv 1 \cdot a \equiv a \bmod n$
4) Fiecare element are un invers multiplicativ (acest invers este usor de calculat cu algoritmul Euclidian Extins) $\forall a, \exists a^{-1} \mid a \cdot a^{-1} \equiv 1 \bmod n$

- **Definitie:** Functia Euler Φ pentru un intreg n a carui factorizare este cunoscuta se poate calcula ca

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_r^{e_r} \Rightarrow \quad \phi(n) = n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_r}\right)$$

- **Observatie:** Functia Euler Φ reprezinta numarul de intregi mai mici decat n si relativ primi la acesta si este ordinul grupului $(Z_n^*, \cdot)$

Teorema lui Euler

- **Teorema (Euler)**: Pentru orice intreg n supraunitar avem:

$$x^{\phi(n)} \equiv 1 \bmod n \; \forall x \in Z_n^*$$

- **Demonstratie** (a fost facuta la tabla):
- In criptografie se lucreaza in general cu intregi care sunt produsul a doua numere prime, in acest caz:

$$n = p \cdot q \Rightarrow \phi(n) = (p-1)\cdot(q-1)$$

$$\Rightarrow x^{\phi(n)} = x^{(p-1)\cdot(q-1)} \equiv 1 \bmod n \; \forall x \in Z_n^*$$

- Exemplu

$$n = 13 \cdot 17 = 221 \Rightarrow \phi(n) = (13-1)\cdot(17-1) = 192$$

$$\Rightarrow x^{\phi(n)} = x^{192} \equiv 1 \bmod 221 \; \forall x \in Z_n^*$$

Ce se poate calcula eficient in $(Z_n^*, \cdot)$

1) **Adunare**, pentru a aduna doi intregi de l digiti in baza b sunt necesare $O(l)$ adunari la nivel de digit
2) **Scadere**, pentru a aduna doi intregi de l digiti in baza b sunt necesare $O(l)$ scaderi la nivel de digit
3) **Inmultire**, pentru a inmulti doi intregi de l respectiv t digiti in baza b sunt necesare $O(lt)$ inmultiri la nivel de digit
4) **Impartire**, pentru a imparti un intreg de l digiti la un intreg de t digiti sunt necesare $O(lt)$ inmultiri si $O(l\text{-}t)$ impartiri la nivel de digit

5) **Ridicarea la putere**, pentru a ridica un intreg la puterea e sunt necesare $O(3/2lg(e))$ inmultiri

6) **Inverse multiplicative si cmmdc**, inversul multiplicativ al unui intreg si cmmdc a doi intregi se poate calcula cu Algoritmul Euclidean Extins, complexitate $O((lgn)^2)$

7) **Sistemele de congruente lineara** (vezi mai jos) care respecta Teorema Chineza a resturilor folosind algoritmul lui Gauss, complexitate $O((lgn)^2)$

Teorema chineza a resturilor: sistemul $\begin{cases} x = a_1 \bmod n_1 \\ x = a_2 \bmod n_2 \\ \cdots\cdots\cdots \\ x = a_3 \bmod n_k \end{cases}$ are solutie unica in $Z_n, n = \prod_{i=1}^{k} n_i$ daca $cmmdc(n_i, n_j) = 1, \forall i \neq j$

Agoritmul lui Gauss: solutia sistemului este $x = \sum_{i=1}^{k} a_i N_i M_i \bmod n, n = \prod_{i=1}^{k} n_i,\ N_i = \frac{n}{n_i},\ M_i = N_i^{-1} \bmod n_i$

Ce nu se poate calcula eficient in $(Z_n^*, \cdot)$ (I)

1) **Problema factorizarii** intregului n **(PFACT)**

Pentru un intreg n, $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k} = \prod_{i=1}^{k} p_i^{e_i}$

2) **Problema radacinii patrate (PRP)** se poate calcula eficient **daca si numai daca** se cunoaste factorizarea lui n (problema calculului radacinii patrate este echivalenta cu problema factorizarii lui n)

Pentru un intreg x, a astfel incat $x=a^2 mod n$

3) **Problema radacinii de ordin *k* (Problema RSA - RSAP)**, $cmmdc(k, \Phi(n))=1$: se poate calcula eficient **daca** se cunoaste factorizarea lui n (Nu exista nici o demonstratie ca aceasta problema ar fi echivalenta problemei factorizarii)

Pentru un intreg x, a astfel incat $x=a^k mod n$

4) **Problema Cheilor RSA (PCRSA)**, *pentru* ε, $cmmdc(\varepsilon, \Phi(n))=1$, *calculeaza* δ *astfel incat* $\varepsilon\delta=1 mod\ \Phi(n)$: se poate calcula eficient **daca si numai daca** se cunoaste factorizarea lui n

Problema Radacinii de ordin k in Z_n

- Definim functia: $f : Z_n^* \to Z_n^*,\ f(x) = x^k \bmod n$
- Daca k este relativ prim la $\Phi(n)$ are invers multplicativ modulo $\Phi(n)$:

$$\exists d\ a.i.\ d \cdot k \equiv 1 \bmod \phi(n)$$

- In acest caz functia este inversabila si inversa este:

$$f^{-1} : Z_n^* \to Z_n^*,\ f^{-1}(x) = x^d \bmod n$$

- Calculul functiei inverse poate fi eficient facut daca se cunoaste factorizarea lui n, altfel nu se cunoaste nici o metoda eficienta de extragere a radacinii

Ordinul unui element in $(Z_n^*, \cdot)$

- ***Definitie:*** Ordinul unui element a din Z_n este cel mai mic intreg t pentru care:

$$a^t \equiv 1 \bmod n$$

- Elementele de ordin $\Phi(n)$ se numesc generatori ai lui Z_n are generatori(orice element din Z_n poate fi scris ca putere a generatorului)

- Z_n are generatori daca si numai daca: $n = 2, 4, p^k, 2p^k$

Ce nu se poate calcula eficient in $(Z_n^*, \cdot)$ (II)

4) **Problema Logaritmului Discret (PLD)**

Avand α, β gaseste γ astfel incat $\beta = \alpha^{\gamma} \bmod n$

- In sisteme criptografice asimetrice problema logaritmului discret se foloseste pentru grupul Z_p cand:
  1) p este numar prim
  2) p-1 are un divizor prim p' suficient de mare
  3) α este generator al lui Z_p
- In practica uneori se recurge la alegerea unui numar care nu este generator al lui pentru a creste viteza de criptare/decriptare – duce la scaderea nivelului de securitate
- Problema se poate generaliza pe orice grup algebric care are generatori: de exemplu grupul format de punctele unei curbe eliptice definite pe un camp finit

Ce nu se poate calcula eficient in $(Z_n^*, \cdot)$ (III)

5) **Problema Diffie-Hellman (PDH)** *Avand pe a generator in* Z_p *si* a^{α}, a^{β} *gaseste-l pe* $a^{\alpha\beta}$ *mod p*

- ***Definitie:*** Spunem ca problema P_1 se reduce polinomial la P_2 daca exista un algoritm A_1 care rezolva P_1 si care are ca subrutina un algoritm A_2 care rezolva pe P_2 iar A_1 ruleaza in timp polinomial daca A_2 ruleaza in timp polinomial (se noteaza $P_1 \leq_P P_2$)
- Cu privire la problemele introduse in slideurile (I), (II), (III) sunt valabile relatiile:

(1) $PDH \leq_P PLD$

(2) $PRP \leq_P PFACT$

(3) $PFACT \leq_P PRP$

(4) $PRSA \leq_P PFACT$

(5) $PRSAC \leq_P PFACT$

(6) $PFACT \leq_P PRSAC$

(7) $?\, PLD \leq_P PDH\, ?$

(8) $??\, PFACT \leq_P PRSA??$

(7) nu se stie dar se pare ca e adevarat

(8) nu se stie, cu toate ca majoritatea expertilor cad de acord ca ar trebui sa fie adevarat, recent au aparut cateva rezultate care arata ca s-ar putea sa nu fie adevarat

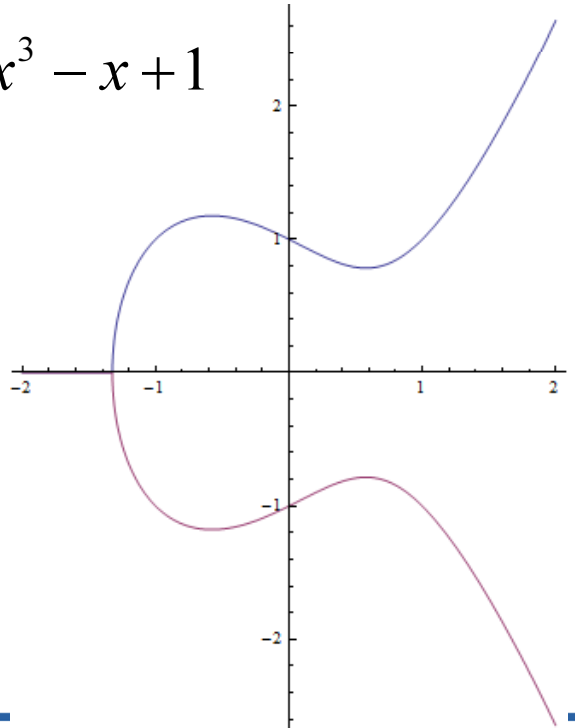
Campuri

- Un grup este o multime G alaturi de o operatie binara * cu proprietatile: * este asociativa, exista element identitate 1, si fiecare element din G are un invers
- Obs 1. Daca * este comutativ, R se numeste abelian
- Un inel este o multime alaturi de doua operatii binare + si * cu proprietatile: R si + formeaza un grup abelian cu element identitate 0, * este asociativ, exista element identitate fata de * notat cu 1 si 1!=0, * este distributiv fata de +
- Un camp este un inel comutativ (a*b=b*a) in care toate elementele nenule au invers multiplicativ
- Caracteristica unui camp este numarul de adunari ale elementului 1 astfel incat rezultatul sa fie 0 (sau implicit 0 in cazul adunarea lui 1 nu conduce niciodata la 0)

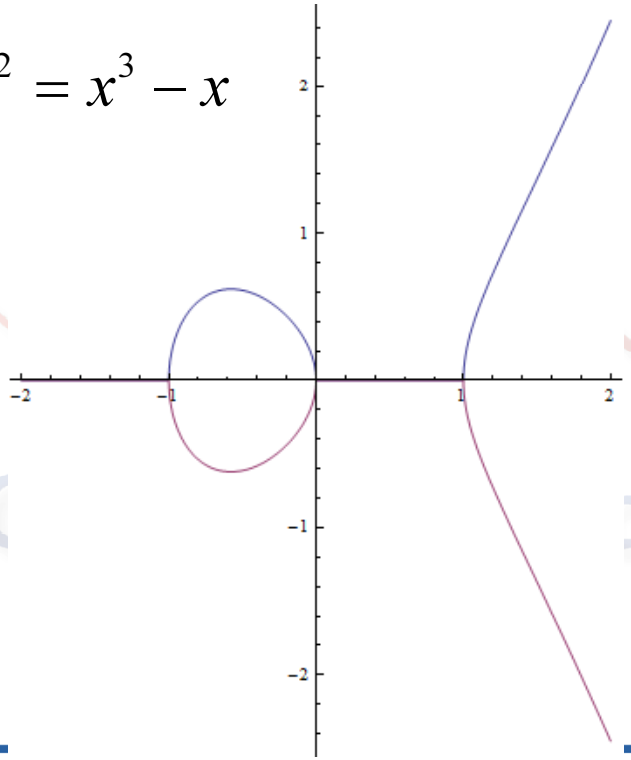
Curbe Eliptice

- Curbe a caror ecuatie, intr-un camp de caracteristica diferita de 2 sau 3, este: $y^2 = ax^3 + bx + c$

$y^2 = x^3 - x + 1$

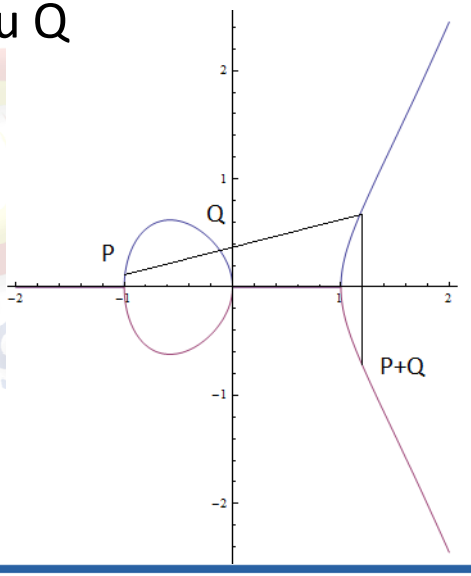


$y^2 = x^3 - x$



Adunarea punctelor unei curbe

- Obs 1. Daca P este O atunci –P=O si P+Q=Q (i.e., P este elementul neutru)
- Obs 2. -P este are aceeasi coordonata x cu P si coordonata y negativa (un astfel de punct tot timpul exista datorita termenului y^2 din stanga ecuatiei)
- Regula de adunare: Daca P si Q au coordonata x diferita, atunci dreapta care trece prin P si Q intersecteaza curba intotdeauna in exact 3 puncte. Fie al treilea punct R, imaginea in oglinda a acestui punct, i.e., -R, este suma lui P cu Q



- Obs 3. Usor de observat ca daca P=-Q atunci P+Q=O
- Obs 4. Daca P=Q atunci dreapta este tangenta la curba (nu schimba cu nimic definitia adunarii)

Ce este simplu/greu de calculat

- O curba eliptica poate fi definita pe un camp finit Fq
- Operatia analoaga ridicarii la o putere modulo n este adunarea punctelor de pe o curba eliptica. Adica x^k devine k*P unde P este un punct de pe curba
- Multiplicarea cu k se poate calcula eficient in maniera similara cu algoritmul Repeated-Square-and-Multiply prin adunari succesive
- De ex. 99P= 2(P+2*2*2*2*2(P+2*P)) +P care inseamna 3 adunari si 7 multiplicari cu 2
- Folosind aceasta observatie se pot construi criptosisteme analoage Diffie-Hellman si ElGamal