CURS 9
CRIPTOGRAFIA ASIMETRICĂ/CU CHEIE PUBLICĂ

Estimarea timpului de executare în funcție de complexitatea computațională

Orice operație elementară durează maxim 5 cicli de procesor! Dacă frecvența unui procesor este f_P = 3GHz => procesorul execută $3\text{GHz} = 3 \times 10^9$ de cicli pe secundă => 1 ciclu de procesor durează aproximativ $\frac{1}{3\text{GHz}} = \frac{1}{3\times10^9} = 0.3 \times 10^{-9}\text{secunde} = 3 \times 10^{-10}\text{secunde}$ => 1 operație elementară durează aproximativ $5 \times 3 \times 10^{-10}\text{secunde} = 15 \times 10^{-10}\text{secunde}$

1. Considerăm un algoritm cu complexitatea $\mathcal{O}(n^2)$ pe care vrem să-l rulăm pentru $n = 20000 = 2 \times 10^4$. Cât este, aproximativ, timpul de executare t?

$t =$ $\underbrace{(2 \times 10^4)^2}_{\text{numărul total de operații elementare}} \times \underbrace{15 \times 10^{-10}}_{\text{durata unei operații elementare}}$ secunde =

$= 4 \times 10^8 \times 15 \times 10^{-10}$ secunde = 6×10^{-1} secunde = 0.6 secunde

2. Considerăm un algoritm cu complexitatea $\mathcal{O}(2^n)$. Care este timpul său de executare pentru $n = 100$?

Vom folosi aproximarea $2^{10} \approx 10^3 \Longrightarrow 2^{100} \approx 10^{30}$

$t =$ $\underbrace{2^{100}}_{\text{numărul total de operații elementare}} \times \underbrace{15 \times 10^{-10}}_{\text{durata unei operații elementare}}$ secunde =

$= 10^{30} \times 15 \times 10^{-10}$ secunde = 15×10^{20} secunde = $\frac{15\times10^{20}}{3600}$ ore = $\frac{15\times10^{16}\times\cancel{10^4}}{\cancel{3600}}$ ore = $15 \times 10^{16} \times 3$ ore = 45×10^{16} ore = $\frac{45\times10^{16}}{24}$ zile = $\frac{45\times10^{14}\times\cancel{10^2}}{\cancel{24}}$ zile = $45 \times 10^{14} \times 4$ zile = 180×10^{14} zile = $\frac{180\times10^{14}}{365}$ ani = $\frac{180\times10^{11}\times\cancel{10^3}}{\cancel{365}}$ ani = $180 \times 10^{11} \times 3$ ani = 540×10^{11} ani = 54000×10^9 ani = 54000 de miliarde de ani!!!

Presupunem că se inventează un procesor astfel încât pentru $n = 100$ să obținem $t = 1$ secundă.

$$2^{n+1} = 2 \cdot 2^n => t_{n+1} = 2 \cdot t_n$$

$n = 100 \Rightarrow t_{100} = 1$ secundă
$n = 101 \Rightarrow t_{101} = 2 \cdot t_{100} = 2$ secunde
$n = 102 \Rightarrow t_{102} = 2 \cdot t_{101} = 2^2$ secunde $= 4$ secunde
.....
$n = 200 \Rightarrow t_{200} = 2^{100}$ secunde $\approx \frac{54000 \times 10^9}{15 \times 10^{-10}}$ ani $\approx 3.5 \times 10^{19}$ ani

Complexitatea unui algoritm TREBUIE să fie exprimată în funcție de dimensiunea datelor de intrare (reprezentate binar), ci nu în funcție de valorile datelor de intrare!!!

Exemplu:
Testarea primalității unui număr natural n:

```
int n;
printf("n = ");
scanf("%d", &n);

int prim = 1;
for(d = 2; d <= n/2; d++)
	if(n % d == 0)
	{
		prim = 0;
		break;
	}

if(prim == 1)
	printf("Numar prim");
else
	printf("Numar compus");
```

Complexitate: ~~$\mathcal{O}(n/2) \approx \mathcal{O}(n)$~~ → estimarea NU este corectă, deoarece n este o valoare de intrare, ci nu dimensiune datelor de intrare reprezentate binar!!!

Complexitate: $\mathcal{O}(2^{1+[\log_2 n]}) \approx \mathcal{O}(2^{[\log_2 n]})$ => complexitatea exponențială în raport de lungimea reprezentării binare a lui n!!!

În criptografia cu cheie publică se utilizează numere prime pe 512 sau 1024 de biți!!!

Observație: Orice număr natural n se reprezintă binar folosind $1 + [\log_2 n]$ biți!

Demonstrație: Presupunem că numărul n are x cifre în reprezentarea binară => $\underbrace{10\ldots0}_{x \text{ cifre}} \leq n < \underbrace{10\ldots0}_{x+1 \text{ cifre}}$ => $2^{x-1} \leq n < 2^x$ => $x - 1 \leq \log_2 n < x$ => $[\log_2 n] = x - 1$ => $x = [\log_2 n] + 1$.

Observație: Dacă un număr natural n se reprezintă binar folosind cel puțin x biți atunci $2^{x-1} \leq n < 2^x$.

Problema logaritmului discret

Fie p un număr prim și $a, b \in \mathbb{Z}_p$ $(a \neq 0)$. Să se determine $x \in \mathbb{Z}_{p-1}$ astfel încât $b^x \equiv a (\mathrm{mod}\, p)$. Dacă există numărul x, atunci el este unic și se numește *logaritmul discret* al numărului a în baza b în grupul $\mathbb{Z}_p$. De obicei, logaritmul discret se notează cu $\mathrm{dlog}_b\, a$ sau $\log_b a$.

Exemplul 1: $\mathbb{Z}_{11}$ și $b = 6$

k	0	1	2	3	4	5	6	7	8	9
$6^k (\mathrm{mod}\, 11)$	1	6	3	7	9	10	5	8	4	2

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{dlog}_6\, a$	0	9	2	8	6	1	3	7	4	5

Exemplul 2: $\mathbb{Z}_{11}$ și $b = 3$

k	0	1	2	3	4	5	6	7	8	9
$3^k (\mathrm{mod}\, 11)$	1	3	9	5	4	1	3	9	5	4

Se observă faptul că nu există $\mathrm{dlog}_3\, a$ pentru $a \in \{2, 6, 7, 8, 10\}$!

Complexitatea celor mai buni algoritmi pentru calculul logaritmului discret într-un grup $\mathbb{Z}_p$ (Baby-step giant-step, Pohlig–Hellman, Pollard's rho, Index calculus algorithm) este $\mathcal{O}(\sqrt{p}) \approx \mathcal{O}(2^{[\log_2 p]/2})$, deci este o complexitate exponențială!!!