Curs 1/Partea 2: Problema zilei de nastere

Prezentăm în continuare o problemă celebră în teoria probabilităţilor, numită *problema zilei de naştere*, care a constituit punctul de plecare pentru dezvoltarea unor metode de analiză a algoritmilor, precum şi în criptografie.

Exemplul 1. (Problema zilei de naştere) n persoane participă la un reuniune. Care este probabilitatea ca cel puţin două dintre ele să aibă aceeaşi zi de naştere, în ani diferiţi sau nu?

Presupunem că anul are 365 de zile (ignorăm anii bisecţi). Notăm cu A mulţimea participanţilor la reuniune, $A = \{1, 2, \ldots, n\}$ şi cu $B = \{1, 2, 3, \ldots, 365\}$ mulţimea codurilor pentru zilele anului.

• Mulţimea Ω a tuturor posibilităţilor pentru zilele de naştere ale celor n persoane coincide cu mulţimea n–listelor cu elemente din B sau echivalent cu mulţimea aplicaţiilor de la A la B, adică $\Omega = B^A$. De exemplu, dacă $n = 15$ o lista posibilă de zile de naştere este:

$$(32, 24, 125, 51, 73, 84, 279, 330, 23, 127, 71, 95, 199, 211, 5),$$

unde 32 inseamna 1 februarie, 24 este 24 ianuarie, 125 calculati voi, etc.

Cardinalul lui Ω este $|\Omega| = 365^n$.

• Lucrăm în ipoteza că zilele de naştere sunt aleator (la întâmplare) distribuite în cele 365 zile şi n–listele de zile de naştere sunt echiprobabile.

• Fie E_n evenimentul "cel puţin două persoane din cele n ce participă la reuniune au aceeaşi zi de naştere". Probabilitea acestui eveniment este $P(E_n) = |E_n|/|\Omega|$. Este mai simplu însă să calculăm probabilitatea complementarei lui E_n.

• Complementul $\complement_\Omega E_n$ este evenimentul ca printre cele n persoane să nu existe două cu aceeaşi zi de naştere, adică:

$$\complement_\Omega E_n = \{f \in B^A \mid f \text{ este injecţie}\}$$

este mulţimea aplicaţiilor injective de la A la B (o injecţie asociază la oricare două per-

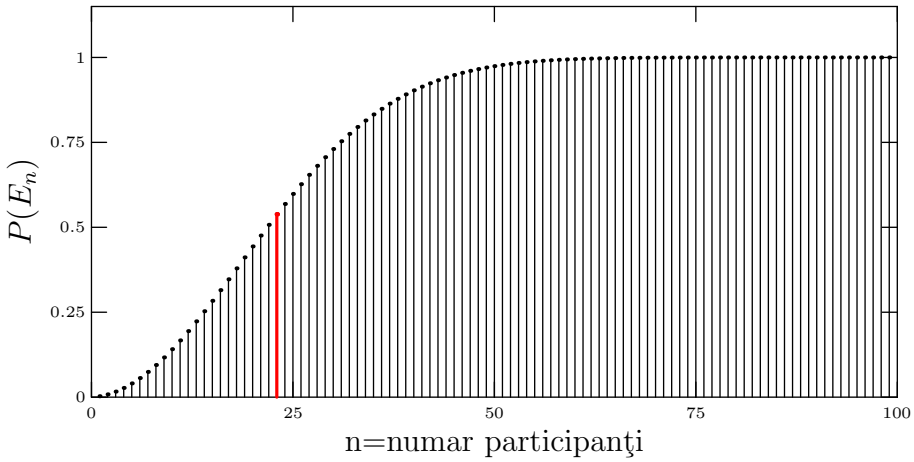

Fig. 1: Ilustrarea prin impulsuri a poziţiei punctelor $(n, P(E_n))$. Pentru $n \geq 23$, $P(E_n) > 0.5$, $n = 12, \ldots, 100$.

soane diferite, zile de naştere diferite). Astfel:

$$\begin{aligned} P(\complement_\Omega E_n) &= \frac{\text{numărul injecţiilor de la A la B}}{\text{numărul aplicaţiilor de la A la B}} = \frac{A_{365}^n}{365^n} = \frac{365(365-1)\cdots(365-n+1)}{365^n} \\ &= \frac{365}{365}\frac{365-1}{365}\cdots\frac{365-(n-1)}{365} = \left(1-\frac{1}{365}\right)\left(1-\frac{2}{365}\right)\cdots\left(1-\frac{n-1}{365}\right) \\ &= \prod_{k=1}^{n-1}(1-\frac{k}{365}) \end{aligned}$$

Deci,

$$P(E_n) = 1 - P(\complement_\Omega E_n) = 1 - \prod_{k=1}^{n-1}\left(1-\frac{k}{365}\right)$$

Pentru a extrage informaţie din această formulă de calcul procedăm astfel: pentru $n = 2, 3, 4, \ldots 100$, calculăm probabilitatea $p_n = P(E_n)$, adică probabilitatea ca într-un grup de n persoane să fie cel puţin două cu aceeaşi zi de naştere. Într-un sistem de axe ortogonale vizualizăm punctele de coordonate (n, p_n). Fig.1 ilustrează aceste puncte.

S-a constat că probabilitatea de a găsi într-un grup de 23 de persoane cel puţin două cu aceeaşi zi de naştere este mai mare decât 1/2: $p_{23} = 0.507297$.

Acest rezultat este contrar intuiţiei umane. Pare nefiresc ca într-un grup de doar 23 persoane să existe cel puţin două cu aceeaşi zi de naştere, cu o probabilitate mai mare de 1/2. De aceea, acest fenomen este cunoscut în CS ca *the birthday paradox*.

Generalizare. Problema probabilităţii coincidenţei zilei de naştere apare într-o formă generalizată în criptografie şi analiza algoritmilor. Şi anume: avem n obiecte numerotate $1, 2, \ldots, n$ şi N containere (recipienţi), $N \geq n$. Pe rând obiectele sunt atribuite

la întâmplare containerelor (sunt aruncate la întâmplare în containere). La întâmplare înseamnă că fiecare obiect poate ateriza în oricare container cu aceeaşi probabilitate. Dacă într-un container au căzut cel puţin două obiecte spunem că s-a produs o *coliziune*. Probabilitatea să se producă cel puţin o coliziune după aruncarea celor n obiecte în cele N containere se calculează similar cu probabilitatea să participe la reuniune cel puţin două persoane cu aceeaşi zi de naştere. În cazul coliziunii 365 se înlocuieşte cu N. Astfel avem:

Propoziţia 1.0.1 *Pentru $N \geq n \geq 2$ probabilitatea a cel puţin unei coliziuni $C(N,n)$ este*

$$P(C(N,n)) = 1 - \prod_{k=1}^{n-1}(1 - \frac{k}{N})$$

şi

$$P(C(N,n)) \geq 1 - e^{-n(n-1)/2N}$$

Demonstraţie: **OPŢIONAL**

Probabilitatea de nu se produce nici o coliziune după aruncarea celor n obiecte în N containere este:

$$1 - P(C(N,n)) = \prod_{k=1}^{n-1}\left(1 - \frac{k}{N}\right)$$

Folosind inegalitatea $1 - x \leq e^{-x}$, valabilă pentru orice $x \in [0,1]$, avem că:

$$\left(1 - \frac{k}{N}\right) \leq e^{-k/N} \text{ şi } \prod_{k=1}^{n-1}\left(1 - \frac{k}{N}\right) \leq \prod_{k=1}^{n-1} e^{-k/N} = e^{-\frac{1}{N} - \frac{2}{N} - \cdots - \frac{n-1}{N}} = e^{-n(n-1)/2N}$$

Deci probabilitatea producerii a cel puţin unei coliziuni este cel puţin $1 - e^{-n(n-1)/2N}$, adică:

$$P(C(N,n)) \geq 1 - e^{-n(n-1)/2N} \geq 1 - e^{-(n-1)^2/2N}$$

Notând această probabilitate cu $P(n)$ din $P(n) \geq 1 - e^{-(n-1)^2/2N}$ obţinem prin inversare

$$n(P) \leq 1 + \sqrt{2N \ln \frac{1}{1-P}} \tag{1}$$

adică numărul de obiecte ce trebuie aruncate în N containere pentru a obţine o coliziune cu probabilitatea P este cel mult $1 + \sqrt{2N \ln \frac{1}{1-P}}$. □

Bazat pe acest rezultat s-a concluzionat că într-o căutare a unui element, obiect, etc, este mult mai uşor să găsim obiecte identice, decât un obiect particular. Astfel s-au dezvoltat algoritmi de coliziune. Ideea de bază a unui astfel de algoritm este să se constituie două liste de elemente şi apoi să se caute un element ce apare în ambele, adică să se identifice o coliziune.

Să ilustrăm ce informaţie obţinem din această dependenţă a numărului n de probabilitatea P, relativ la şansa pe care o are un adversar să "spargă o valoare hash".

Atacul birthday asupra unei funcţii hash

O funcţie hash $h : \{0,1\}^{\leq L} \to \{0,1\}^n$ este o funcţie ce asociază unui string de biţi de lungime variabilă, mai mică cel mult egală cu L fixat, un string de lungime fixă, n. Hashing înseamnă *chopping and mixing*, adică a ciopârţi şi a mixa. Principiul de bază al construirii unei funcţii hash criptografice este următorul: stringul iniţial de biţi este divizat în substringuri şi acestea sunt mixate prin operaţii pe biţi şi rezultatul este un string de lungime n.

Un exemplu de funcţie hash este semnătura digitală. Pentru autentificarea expeditorului unui mesaj sau document, de către destinatar, se adaugă mesajului un bloc de date ce constituie aşa numita semnătură digitală a expeditorului. Ea se constituie asociind unui string s de lungime variabilă $\leq L$ (ce conţine informaţii despre expeditor), un string de lungime fixă, de n biţi, ca rezultat al hashing-ului stringului s

O funcţie hash se construieşte astfel încât să poată fi rapid evaluată, dar dacă un adversar interceptează o valoare hash $v \in \{0,1\}^n$ (semnătura digitală, de exemplu) să fie teoretic imposibil ca într-un timp limitat, să determine unui string s cu proprietatea că $h(s) = v$, chiar dacă acesta dispune de o putere mare de calcul.

Paradoxul zilei de naştere a sugerat şi necesitatea asigurării unei alte proprietăţi de securitate a funcţiei hash, numită rezistenţa la coliziune. Şi anume, într-un atac, numit *birthday attack*, adversarul încearcă să găsească două stringuri s, s' ce au aceeaşi valoare hash: $h(s) = h(s')$.

Să analizăm în cazul functiei cu valori hash pe 64 de biţi, câte stringuri trebuie să genereze adversarul, la întâmplare, pentru a obţine cu probabilitate mai mare de $1/2$, cel puţin o coliziune, adică două stringuri s, s' ce au aceeaşi valoare hash. Cu alte cuvinte evaluăm $n(1/2)$ din relaţia (1) în cazul $N = 2^{64}$, unde N este numărul valorilor hash posibile pentru funcţia h:

$$n(1/2) \leq 1 + \sqrt{2^{65} \ln \frac{1}{1-1/2}} \approx 1 + 5.056937540686587e + 009$$

Dacă programul adversarului generează într-o secundă un milion de stringuri şi valorile lor hash, atunci pentru a evalua $5.056937540686587e + 009$ (neglijăm 1) valori hash are nevoie de

$5.056937540686588e + 003$secunde ≈ 84 minute $= 1$ oră şi 24 minute(deci nu prea mult!)

În concluzie pentru a preveni succesul unui atac birthday, se construiesc funcţii hash cu valori exprimate printr-un număr mare de biţi. Funcţia hash criptografică, `SHA1`, folosită pentru semnături digitale generează valori hash în $\{0,1\}^{160}$, în timp ce `MD5` în $\{0,1\}^{128}$.