KRIPTOGRÁFIAI HASH FÜGGVÉNYEK

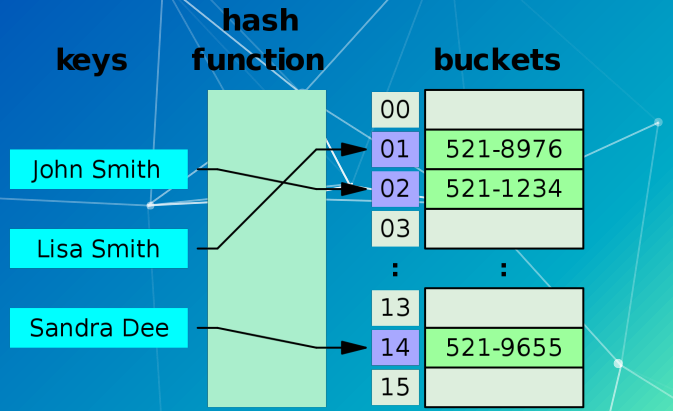
Szabó Gergely András – AX2M1L

HASH FÜGGVÉNYEK, HASH TÁBLÁK

HASH TÁBLÁK

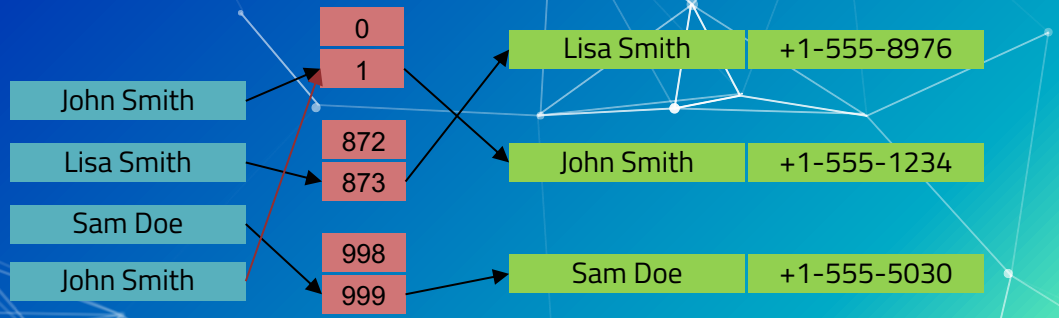
- 1953-ban találták fel, a minél jobb kereső algoritmusok kidolgozásának motivációja által
- Asszociatív tömböt implementáló adatszerkezet
- Indexének meghatározása egy úgynevezett hash függvény segítségével történik
- Az adatok tárolása kulcs érték alapján történik

Tárigény	O(n)
Beszúrás	O(1)
Keresés	O(n)
Törlés	O(n)



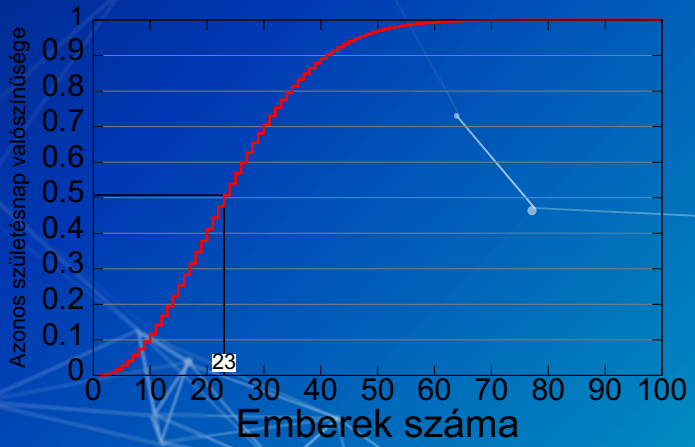
HASH ÜTKÖZÉSEK

- Pl: Születésnapi paradoxon - alacsony számú beszúrás után magas a valószínűsége az ütközések előfordulásának.
- Ha a hash táblában az embereket a születésnapjuk alapján hasheljük, akkor már 23 beszúrás esetén 50%-os valószínűséggel jelentkezik hash ütközés.
- Számolnunk kell azzal, hogy két különböző kulcsra ugyanazt a hash-t kapjuk, és kezelnünk kell az ilyenkor fellépő hash ütközést.



SZÜLETÉSNAPI PARADOXON

- A születésnap-paradoxon az a jelenség, miszerint megdöbbentően nagy a valószínűsége annak, hogy viszonylag kevés egy szobában lévő személy közül lesz kettő, akiknek a születésnapja azonos hónap azonos sorszámú napjára esik.
- 23 ember esetén 23 × 22 / 2 = 253 pár van, mindegyik pár egy lehetséges egyezés.
- A valószínűség közelítő kiszámításához elhanyagolunk pár részletet, így a szökőéveket, azt, hogy az emberek között lehetnek ikrek, valamint a különböző születési statisztikákat.



SZÜLETÉSNAPI PARADOXON

- Számoljuk ki, hogy mi a valószínűsége annak, hogy n emberből mindenki más napon született:

$$p = \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365 - n + 1}{365}$$

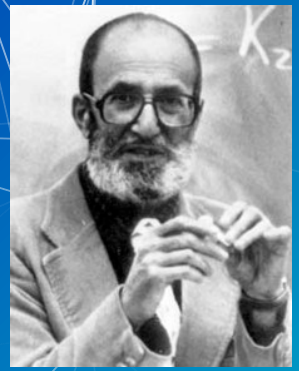
- A faktoriális jelölést használva ugyanezt így is felírhatjuk:

$$p = \frac{\frac{365!}{(365-n)!}}{365^n} = \frac{365!}{365^n (365 - n)!}$$

- Ezek után 1 – p annak a valószínűsége, hogy legalább két embernek egy napra esik a születésnapja. n = 23-ra ez az érték kb. 0,507.

SZÜLETÉSNAPI PARADOXON - MÁSIK MEGKÖZELÍTÉS

- Halmos Pál magyar születésű amerikai matematikus.
- Számos területen ért el jelentősebb eredményeket, többek között hozzájárult a matematikai logika, valószínűségszámítás, statisztika, halmazelmélet, funkcionálanalízis (Hilbert-tér), mértékelmélet és ergodelmélet fejlődéséhez.
- Életrajzában említette: „A probléma megközelítésének egyik módja, hogy megfordítva tesszük fel a kérdést: »Legalább hány embernek kell a szobában lennie ahhoz, hogy kevesebb, mint 1/2 valószínűséggel legyen csupa különböző születésnapjuk?«”



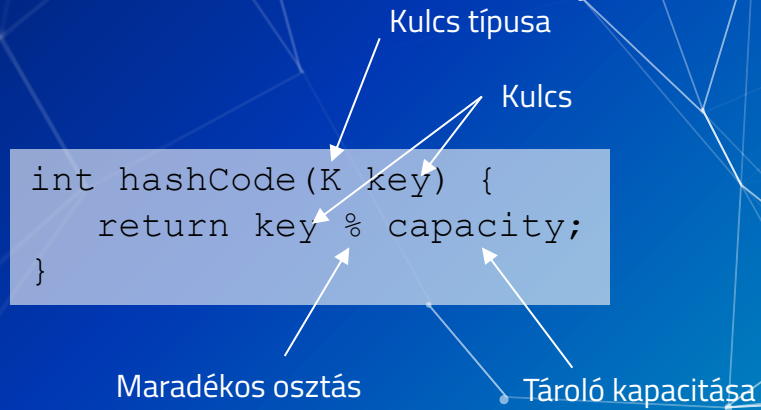
Halmos Pál (1916 - 2006)

$$\prod_{k=1}^{n-1}\left(1-\frac{k}{365}\right) < \left(\frac{1}{n-1}\sum_{k=1}^{n-1}\left(1-\frac{k}{365}\right)\right)^{n-1} = \left(1-\frac{n}{730}\right)^{n-1} < \left(e^{-n/730}\right)^{n-1} = e^{-(n^2-n)/730}.$$

- ahol az első egyenlőtlenség a számtani és mértani közepek egyenlőtlensége, a második pedig az $1 - x < e^{-x}$ összefüggésből következik.

HASH FUNKCIÓ MŰKÖDÉSE

- C++ hashkulcs generálás implementációja szám típusú hashmap kulcsra



MIK AZOK A KRIPTOGRÁFIAI HASH FÜGGVÉNYEK?

"A hash függvények olyan informatikában használt eljárások, amelyekkel bármilyen hosszúságú adatot adott hosszúságra képezhetünk le. Az így kapott véges adat neve hash/hasító érték."

Hello World!

Hash függvény

```
7f83b1657ff1fc53b92dc18148a1d65d
fc2d4b1fa3d677284addd200126d9069
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sit amet est auctor, condimentum orci vel, accumsan libero.

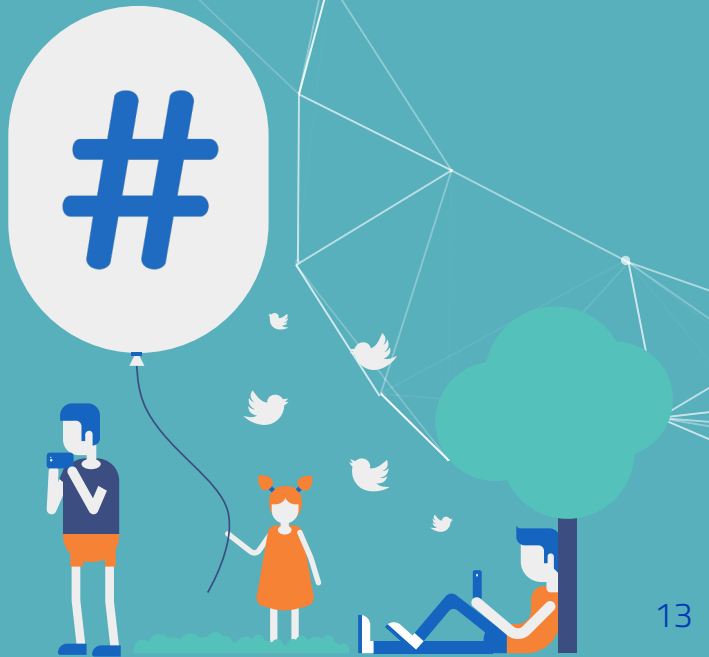
```
2f5c419fc34121c179761824c76d2781
3c269226e8048b2a62612a7bd53ee2d8
```

A HASH FÜGGVÉNYEK TULAJDONSÁGAI

TULAJDONSÁGOK

- Csekély valószínűséggel két különböző fájl azonos hasht produkálhat
- Egy irányú, vissza nem fejthető
- Lavina hatás – kis adatváltozás esetén, jelentős kiement változás

MIÉRT IS KELLENEK EZEK NEKÜNK?



HASZNÁLATI ESETEK

- Adatok, fájlok ellenőrzése
- Nagy szövegek, fájlok összehasonlítása
- Jelszavak titkosítása
- Kliens oldali érzékeny adatok tárolása azonosításra
- Azonosítási rendszerek
- Digitális aláírás

NÉPSZERŰBB FAJTÁI

MD5

- 1991-ben fejlesztette ki Ronald L. Rivest (1947-) professzor az RSA algoritmus egyik megalkotója
- 128 bites, egyirányú kódolási algoritmus (a lehetséges kimenetek száma 2^{128} –on)
- Visszafejtése nem lehetséges, viszont létezik egy jelenleg több mint 15 millió hasht és a hozzá tartozó jelentést tartalmazó oldal: www.md5decrypter.com
- 1996-ban fedeztek fel egy súlyos hibát az MD5 kódjában, új támogatott algoritmus: **SHA-1**
- 2004-ben biztonsági réseket találtak benne
- 2005 óta elektronikus aláírás területen használata nem javasolt
- 2010. december 31-évél az utódja, az SHA-1 algoritmus is kiváltandó az SHA-256 algoritmussal.

```
MD5("majom"): bcb559cd9d05046da8ec6ea3175a834c
MD5("bajom"): e20c0bddf6416a2021f18b6b05784e88
```

SHA-0

- 1993-ban publikálta az NSA (**National Security Agency) rádióelektronikai, jelhírszerzéssel foglalkozó (SIGINT) hírszerző szervezete**
- 160 bites, egyirányú kódolási algoritmus (a lehetséges kimenetek száma 2^{160} –on)
- Secure Hash Algorithm rövidítése
- 1993-ban szabványosította a NIST (National Institute of Standards and Technology)
- 1995-ben visszavonták a szabványt biztonsági okokra hivatkozva

SHA-1

```
SHA1("mattori"): 5a474c141f6b69998a228fe6970741f2c3cf169a
```

- 1995-ban publikálták és szabványosították
- Leváltotta az SHA-0-át, ez is 160 bites
- Technikailag 1 bit forgatásban tér el az SHA-0-tól
- 2004-ben, majd 2010-ben talált sebehetőségek miatt nem ajánlják a használatát
- 2017-ben 320 millió SHA-1-el hashelt jelszó, 99.9999%-át sikerült visszafejteniük

SHA-2

- 2001-ben publikálták a FIPS PUB 180-2 számú dokumentumban
- Tipusai: 224, 256, 384 vagy 512 bitesek
- Kezdetekben kevésbé terjedt el, okai: az SHA-1-nél sem találtak akkor még ütközést, Windows XP SP2 vagy régebbi verziók nem támogatják
- Napjaink ajánlott hash algoritmusa

```
SHA256("mattori"): aacf71bddcb3fe47eeaf9703cec77336c6050f12536dd213dd3b54352b4644e5
```

SHA-256 MŰKÖDÉSE

```
// Változók inicializálása
// Az első 8 prím négyzetgyökének tört részének 2^32 -szerese
// hexadecimálisan
// Pl.: sqrt(2) ≈ 1.4142135624
// => 0.4142135624 * 2^32 ≈1779033703,952099384902777 = 0x6a09e667

h0 := 0x6a09e667
h1 := 0xbb67ae85
h2 := 0x3c6ef372
h3 := 0xa54ff53a
h4 := 0x510e527f
h5 := 0x9b05688c
h6 := 0x1f83d9ab
h7 := 0x5be0cd19
```

SHA-256 MŰKÖDÉSE

```
// Változók inicializálása
// Az első 64 prím köbgyökének tört részének 2^32 -szerese hexadecimálisan

k[0..63] := 0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b,
0x59f111f1, 0x923f82a4, 0xab1c5ed5, 0xd807aa98, 0x12835b01, 0x243185be,
0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174, 0xe49b69c1,
0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc,
0x76f988da, 0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3,
0xd5a79147, 0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc,
0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85, 0xa2bfe8a1,
0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585,
0x106aa070, 0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3,
0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814,
0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

SHA-256 MŰKÖDÉSE

```
// Az üzenet minden 512 bites darabjára a következő műveletek végrehajtása:
w[i]: az 512 bites rész egy 32 bites részlete, i≤0≤15
for each 512-bit chunk of message

// A 16 db 32 bites szó kiterjesztése 64 bitesre
for i from 16 to 63
    s0 := (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)
    s1 := (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor (w[i-2] rightshift 10)
    w[i] := w[i-16] + s0 + w[i-7] + s1

// Inicializálja az aktuális hash értékeket
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    f := h5
    g := h6
    h := h7
```

SHA-256 MŰKÖDÉSE

```
// Fő kompressziós folyamat
for i from 0 to 63
        S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) xor ((not e) and g)
        temp1 := h + S1 + ch + k[i] + w[i]
        S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) xor (a and c) xor (b and c)
        temp2 := S0 + maj

        h := g
        g := f
        f := e
        e := d + temp1
        d := c
        c := b
        b := a
        a := temp1 + temp2
```

SHA-256 MŰKÖDÉSE

```
// A kompresszált értékek hozzáadása az aktuális hash értékekhez
h0 := h0 + a
h1 := h1 + b
h2 := h2 + c
h3 := h3 + d
h4 := h4 + e
h5 := h5 + f
h6 := h6 + g
h7 := h7 + h


// A folyamat legvégén a végső hash érték létrehozása

digest := hash := h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7
```

ÖSSZEHASONLÍTÁS

Algoritmus	Kimenet mérete	Talált ütközés	Ajánlott	Publikálás	Sebesség (1 MB)
MD5	128 bit	5 db	nem	1992	80 ms
SHA-0	160 bit	10 db	nem	1993	88,6 ms
SHA-1	160 bit	10 db	nem	1995	88,6 ms
SHA-2	224, 256, 384, 512 bit	3 db	igen	2001	87,5 ms

FELHASZNÁLÓI

- Biztonsági alkalmazások, protokollok SHA-1-et használnak: TLS és SSL, PGP, SSH, S/MIME, és IPsec.
- A bitcoin digitális fizetőeszköz és klónjai SHA-256 algoritmust használnak.
- Digitális aláírás: mindig a legújabb ajánlott algoritmust használja
- Jelszavak tárolása: SHA-256 és felette
- Üzenetintegritás: biztosítás hogy az adott vevő ellenőrizhesse a kapott adatok épségét.

  Pl.: Linux disztribúciók letöltése esetén.

ÖSSZEHASONLÍTÁS

HASH FÜGGVÉNY ÉS KRIPTOGRÁFIAI HASH FÜGGVÉNY

- Mindegyik egy adathalmazból képez le
- A hash függvények adattárolásnál azonosító létrehozására használjuk a gyorsabb keresés érdekében
- A kripográfiai hash függvényeket adatok ellenőrzésére használjuk

KÖSZÖNÖM A FIGYELMET!

FORRÁSOK

- https://hu.wikipedia.org/wiki/Kriptográfiai_hash_függvény
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- https://en.wikipedia.org/wiki/SHA-1
- https://dea.lib.unideb.hu/dea/bitstream/handle/2437/90313/Szakdolgozat_KathiFerenc.pdf;jsessionid=3AFAA609F9877AF9295EA5564CCF6E58?sequence=1
- https://en.wikipedia.org/wiki/MD5
- https://en.wikipedia.org/wiki/SHA-2
- https://hu.wikipedia.org/wiki/Has%C3%ADtófüggvény
- https://hu.wikipedia.org/wiki/Hash_tábla
- https://hu.wikipedia.org/wiki/Születésnap-paradoxon
- https://hu.wikipedia.org/wiki/Halmos_Pál