Számlálások adatfolyamokban

SZAKDOLGOZAT

Készítette: Lukács László

Témavezető: Lukács András

Eötvös Loránd Tudományegyetem
Természettudományi Kar

2004. június

Tartalomjegyzék

1. fejezet

Bevezetés

Az adatbányászatban az adatok megjelenésének egyik modellje az adatfolyam, melyben a tetszőleges sorrendű adatok csak egyszer olvashatóak ki az adatforrásból. Fontos feladat egy adatfolyamban előforduló adatok által kialakított eloszlás különböző paramétereinek meghatározása. A dolgozatban e téma egyik alapkérdésével, az adatfolyam mint halmaz számosságának, azaz a benne levő különböző adatok számának meghatározásával foglalkozunk. Általában nem követeljük meg, hogy a vizsgált algoritmusok egzaktak legyenek, megelégszünk közelítő algoritmusokkal, melyek megfelelően jó becslést adnak a kívánt paraméterekre.

A fönti típusú feladatok megoldására szolgáló algoritmusoknál egy algoritmus jóságának természetes mértéke elsősorban a memóriahsználata, mivel egy esetleg rendkívül nagy adatfolyam valamely globális tulajdonságát kívánjuk meghatározni egyetlen olvasással. A megadott algoritmusok mind úgy működnek, hogy az adatfolyamnak valamilyen lenyomatát tartják számon és minden egyes bejövő adatra ezt a lenyomatot frissítik, majd végül ebből készítik el az algoritmus kimenetét jelentő becslést.

Az algoritmusok fontos mérőszáma az egy lépéshez tartozó lenyomatfrissítés idő- és memóriaelérés-szükséglete. A dolgozatban ez nem okoz jelentős problémákat, mivel ezek a bemutatott algoritmusoknál természetes módon konstansok lesznek.

A feladat tárigényének vizsgálatához legyen N egy természetes szám és $0 < \varepsilon < 1$ valós, ekkor létezik olyan alsó korlát ε és N függvényében, melynél kevesebb memóriában nem lehet egy legfeljebb N számosságú tetszőleges adatfolyam számosságát legfeljebb ε relatív szórással meghatározni. Másrészt algoritmusok megadásával lehet felső korlátokat találni ugyanezen memóriaigényre, a dolgozatban a célunk az lesz, hogy ezen korlátokat minél jobban közelítsük egymáshoz.

Ilyen feladatok alkalmazására tipikus példa az online adatfolyamok esete,

amikor a teljes sorozatot eltárolni reménytelen és fölösleges volna, mégis kíváncsiak vagyunk a számosságra, ebben az esetben az egyszeri olvasás lehetősége a feladat természetéből adódik. Másik példa az offline, de nagyon nagy adatbázisok esete, amikor a feldolgozás elvárt idejébe a többszöri olvasás nem fér bele, valamint az adatok nem véletlen hozzáférésűek. Tipikus példa erre a szalagon tárolt adatok esete. Fontos tulajdonsága a bemutatásra kerülő módszereknek, hogy a becslés pontosságának növelése érdekében az adatfolyam egyes részeit párhuzamosan is földolgozhatjuk a bemutatott algoritmusok segítségével, végül a kapott eredményekből az egész adatfolyam jellemzőit megbecsülhetjük.

Az irodalomban található algoritmusokat valós körülmények között is alkalmazták, például az Intereten terjedő vírusok által megfertőzött gépek számának becsléséhez a vírus által küldött IP-csomagok feladóinak számossága volt használható.

A dolgozatban először három korábbi algoritmust mutatunk be időrendben, melyek áttekintést adnak az adatfolyamok számlálásakor felvetődő feladatokról. Ezen algoritmusokat itt nem elemezzük, csupán a lényegüket alkotó ötleteket tekintjük át.

A harmadik fejezetben az adatfolyamok és a számláló algoritmusok szabatos definícióját adjuk meg. A negyedik fejezetben a számláló algoritmusok memóriaigényére szükséges alsó becslést adunk meg, mely aszimptotikusan nem javítható. A hibatag további élesítésének azonban feltétlenül lenne gyakorlati haszna.

Az ötödik fejezetben Flajolet és Durand [4] LogLog és Super-LogLog algoritmusát részletesen elemezzük. A LogLog-algoritmusnak megadjuk egy olyan új implementációját, melynek memóriaigénye aszimptotikusan optimális. Mivel a gyakorlatban fellépő problémáknál az aszimptotikus eredmény mellett a hibatag nagysága is számottevő, ezért ennek javításához tömörítést javaslunk, illetve az 5.6. szakaszban megadjuk a Super-LogLog algoritmus elemzését, amivel Flajolet és Durand méréseken alapuló megfigyeléseit elméletileg igazoljuk.

A Függelékben a Mellin- és Poisson-transzformáltak a dolgozatban felhasznált tulajdonságait gyűjtjük össze, valamint bemutatunk egy explicit hashfüggvény-családot, amely az elméleti optimumokhoz közeli tulajdonságokkal bír, és amely alkalmazható arra, hogy tetszőleges bejövő adatfolyam elemeit véletlen bitsorozatokként kezelhessük.

2. fejezet

Korábbi algoritmusok

Noha főként az adatfolyamok számosságát meghatározó algoritmusokkal foglalkozunk, először bemutatunk két algoritmust, melyek az adatok egyéb tulajdonságait határozzák meg. Mindkettő történeti szempontból is érdekes, hiszen az első, 1978-ból származó algoritmus mutatott rá, hogy ha egyesével akarunk közelítően számlálni valamely N számig, akkor a precíz számoláshoz szükséges $O(\log N)$ memória helyett $O(\log \log N)$ memória is elegendő lehet, a második, először 1982-ben felfedezet algoritmus pedig a gyakori elemek keresésére vonatkozó alapalgoritmus.

2.1. Két érdekes algoritmus

Történetileg talán az első közelítő számláló algoritmus a Morristól [12] származó algoritmus, melynek segítségével rövid regiszterrel akar a szerző egyesével soká számolni. Ha adatfolyamon számláló algoritmusnak gondoljuk ezt, akkor az adatfolyam hosszát becsüljük.

Az algoritmus egy regisztert használ, melynek értéke kezdetben 0. Az algoritmus futása során növelési kéréseket kap egyesével, kimenetként végül a bejött növelési kérések számára ad egy becslést. Az algoritmusnak paramétere egy $a > 0$ szám, egyben kontrollálja az algoritmus pontosságát, valamint hogy milyen nagy számig lehet számlálni. Az algoritmus a következő:

- Legyen ν egy regiszter, ami kezdetben nulla.
- Mindig, amikor egy növelési kérés érkezik,

  $$\Delta_\nu = \frac{\log(1 + \alpha n)}{\log(1 + \alpha)}$$

  valószínűséggel ν értékét eggyel növeljük meg.

- Az algoritmus pillanatnyi eredményét az

$$f(\nu) = a \cdot \left(\left(1 + \frac{1}{a} \right)^{\nu} - 1 \right)$$

becslés adja

Ha n növelési kérés után a ν_n valószínűségi változó jelenti az algoritmus regiszterének tartalmát, akkor

$$Ef(\nu_n) = n \quad \text{és} \quad D^2 f(\nu_n) = \frac{n(n-1)}{2a},$$

azaz $f(\nu_n)$ jó becslés n valódi értékére. Látható, hogy ha a regiszter legfeljebb N nagyságú számokat tud tárolni, akkor az algoritmus nagyjából $f(N)$-ig tud számlálni, hiszen ennél nagyobb értékekre nagy valószínűséggel fordulna elő túlcsordulás.

Eszerint a fönti algoritmus N-ig $1/\sqrt{2a}$ relatív szórással $\log_2 \log_2 N + O(1)$ memóriában tud számolni, ahol a konstans csupán a-tól függ. Ez a memóriahasználat egyben azt is jelenti, hogy az algoritmusnak legalább $O(1) \cdot \log_2 N$ állapota van. Ennél lényegesen kevesebb állapota pedig nem is lehet, ugyanis ha kevesebb, mint

$$\frac{1}{-\log_2(1 - 2\sqrt{2a})} \log_2 N$$

állapota lenne, akkor a skatulya-elv miatt volna olyan $[x \cdot (1 - 1/\sqrt{2a}), x \cdot (1 + 1/\sqrt{2a})]$ intervallum $[1, N]$-ben, amelyben egy szám sem lehet az algoritmus kimenete. Ez esetben viszont x növelési kérés esetén a kimeneti becslés biztosan legalább $x/\sqrt{2a}$-val különbözik a valódi értéktől, azaz a relatív hiba több, mint $1/\sqrt{2a}$.

Másik hasonló feladat a következő, melyet [10]-ban írnak le, de mint [2]-ben rámutattak, már [11]-ban is szerepel az itt adott algoritmus. Valamely $0 < \varepsilon < 1$ számra meg szeretnénk határozni egy adatfolyamban a ε-nál nagyobb gyakorisággal előforduló elemeket. Ilyenből persze legfeljebb $1/\varepsilon$ darab lehet, sőt ennyi lehet is, úgyhogy ennél kevesebb memória nyilván nem lehet elég a feladat pontos megoldásához. Meg lehet mutatni azonban, hogy a gyakori elemek pontos meghatározásához egyszeri olvasással $\Omega(m \log(n/m))$ memória szükséges, ha a bejövő adatfolyam n hosszú és m fajta elem fordulhat elő a folyamban, ami sokkal rosszabb $1/\varepsilon$-nál. Másrészről azonban meg lehet határozni $1/\varepsilon$ memóriában[1] egy olyan $1/\varepsilon$ méretű adathalmazt egyszeri

[1] egy memóriaegység egy számlálót tartalmaz, ami tehát $\log n$ bitet igényel és egy elemet, ami pedig még $\log m$ bitet

olvasással, amiben benne van minden, legalább ε gyakoriságú elem, ezen túl pedig esetleges további elemek. Ebből persze már még egy olvasással könnyen ki lehet válogatni a valódi gyakori elemeket. Lássuk tehát az algoritmust:

- Legyen $N = \lceil 1/\varepsilon \rceil$ és legyen $x_1, x_2, \ldots, x_N$ N darab $\lceil \log n \rceil$ bites számláló, kezdetben mind legyen nulla.
- Legyen $q_1, \ldots, q_N$ N darab változó, melyekben egy-egy elemet tárolunk, ezekenk tehát $\lceil \log m \rceil$ biteseknek kell lenniük, kezdetben mind üres.
- Minden bejövő q adatra:
  - Ha $q = q_i$, akkor növeljük eggyel x_i-t.
  - Ha pedig $q \notin \{q_1, \ldots, q_N\}$, akkor
    - Ha van üres q_i, akkor legyen $q_i = q$, $x_i = 1$.
    - Ellenkező esetben legyen $x = \min x_j$, vétessen fel ez a minimum például $x_i = x$-en, és csökkentsünk minden x_j-t x-szel, legyen ezután $q_i = q$ és $x_i = 1$.

2.2. Bitmap algoritmusok

Ebben a szakaszban az adatfolyamok számosságának meghatározására egy Estantól, Varghesetől és Fisktől [5] származó algoritmus-családot mutatunk be. Ezen algoritmusok memóriaigénye az optimumnál ugyan jelentősen nagyobb, azonban érdemes őket áttekinteni, mivel az alapötletük nagyon természetes. A továbbiakban felteszük, hogy a bejövő adatok a végtelen bitsorozatok terén vannak reprezentálva és ott egyenletes eloszlásúak (lásd a 3. fejezetet), ezt egy jó hashelő függvény alkalmazásával közelíthetjük (erről részletesebben lásd a Függeléket). Precíz számláláshoz minden lehetséges adathoz egy bitet tárolnánk, kezdetben minden bitet 0-ra állítunk, és ha egy adat beérkezik, az ő bitjét 1-re billentjük. Ennek memóriatakarékosabb megoldása a bitmap-algoritmus, ahol az egyes adatoknak közvetlenül megfelelő biteket részben összevonjuk, részben elhagyjuk.

A bitmap algoritmusokban az adatoknak egy részének bittérképes lenyomatát tároljuk és abból számítunk becslést. A legegyszerűbb bitmap-lenyomatot így definiálhatjuk:

2.1. definíció. Legyenek α és m nemnegatív egészek, ekkor egy $\boldsymbol{\omega}$ adatfolyam $(2^\alpha, 2^m)$-bitmap lenyomatának nevezzük a

$$b_{2^\alpha, 2^m}(\boldsymbol{\omega}) = (\chi_0, \ldots, \chi_{2^m - 1})$$

2^m dimenziós 0-1 vektort, ahol χ_i akkor 1, ha valamely $\omega \in \boldsymbol{\omega}$ adatra ω első $\alpha + m$ bitje bináris számként i-t adja, különben pedig 0. $2^{-\alpha}$ a lenyomat mintavételi aránya, 2^m a lenyomat finomsága.

Mint látjuk, a bitmap lenyomat fölbontja a bejövő adatok terét 2^m darab egyenlő $2^{m-\alpha}$ valószínűségű részre és nyilván tartja, e részek melyikéből tartalmaz adatot az adatfolyam, ez motiválja a bitmap elnevezést.

A bitmap-lenyomatból megbecsülni az adatfolyam számosságát a következő függvénnyel lehet:

2.2. definíció.

$$B_{2^\alpha,2^m}(\boldsymbol{\omega}) = \frac{1}{-\log(1-2^{-\alpha-m})} \log \frac{2^m}{2^m - |b_{2^\alpha,2^m}(\boldsymbol{\omega})|},$$

ahol $|v|$ jelöli a v vektor koordinátáinak összegét.

A fönti becslésben ha $\alpha + m$ nagy, akkor a logaritmus szorzója nagyon jól közelíthető $2^{\alpha+m}$-mel. A fönti becslés következő tétel szerint alkalmas becslése a számosságnak:

2.1. tétel. *Ha $B_{2^\alpha,2^m}$-t Ω_n fölötti valószínűségi változónak tekintjük, akkor*

$$EB_{2^\alpha,2^m} \approx n.$$

Bizonyítás. Annak valószínűsége, hogy $B_{2^\alpha,2^m}$ egy koordinátája 0 legyen, $(1-2^{-\alpha-m})^n$, így a várható érték linearitása miatt $EB_{2^\alpha,2^m} = 2^m - 2^m(1-2^{-\alpha-m})^n$, amiből a logaritmus és a várható érték kapcsolata alapján lehet következtetni az állításra. □

Az érdekes kérdés a becslés szórása:

2.2. tétel. *Tekintsük $B_{2^\alpha,2^m}$-et ismét Ω_n fölötti valószínűségi változónak, ekkor a $\rho = n/2^{\alpha+m}$ jelöléssel*

$$D^2 \frac{B_{2^\alpha,2^m}}{n} \approx \frac{e^\rho - 1}{\rho^2 2^m} = \frac{\delta(\rho)}{2^m}.$$

A fönti tétel szerint a bitmap algoritmus akkor ad valamely fix pozitív konstans relatív hibánal nem rosszab becslést az adatfolyam n számosságára, ha n $2^{\alpha+m}$ két megfelelő fix többszöröse közé esik. Ezt általában nem tudjuk eleve biztosítani, ezért a következő *többszörös bitmap* algoritmust használjuk.

2.3. definíció. Legyen $N = 2^k$ fix kettőhatvány legfeljebb ilyen számosságú adatfolyamokat kívánunk számlálni és legyen $M = 2^m$ fix kettőhatvány. Legyen $B_1, B_2, \ldots, B_k$ k darab bitmap algoritmus, B_i legyen $(2^i, 2^m)$ paraméterű. Minden bejövő adatot minden B_i algoritmusnak adjunk a bemenetére, ha pedig minden adatot elolvastunk, minden algoritmusból számítsunk becslést az adatfolyam számosságára, legyen ez b_i, és végül azt a becslést fogadjuk el, melyre $\delta(b_i/2^{i+m})$ minimális.

Ezen algoritmusban a futásidőben jelentős javítást lehet elérni azáltal, ha az egyes bittérképeket egymásba ágyazva implementáljuk, akkor ugyanis megoldható, hogy minden adatot csak egy bittérképben kelljen elkönyvelni.

A fönti algoritmus $M \log_2 N$ memória használatával tud $O(1/\sqrt{M})$ relatív hibájú becslést adni az adatfolyam számosságára, ami jelentősen rosszabb, mint a LogLog algoritmusból kapható $(1 + o(1)) \log_2 \log_2 N$-es korlát, éppen ezért a fönti állítások bizonyítását nem részletezzük.

Megjegyzendő, hogy a bitmap algoritmus nem csak aszimptotikusan teljesít rosszabbul a LogLognál, hanem gyakorlati feladatok esetében is.

3. fejezet

A számláló algoritmusok definíciója

A továbbiakban az algoritmusunk bemenete egy rekordokból álló egyszer olvasható adatfolyam, melyben a rekordoknak bizonyos részét, esetleg az egész rekordot a rekord kulcsának tekintjük és azt akarjuk megbecsülni, hogy hány *különböző* kulcsú rekord szerepel az adatfolyamban. Ehhez a kulcsokat egy megfelelő hash-függvénnyel bináris sorozattá alakítjuk, föltesszük, hogy a kapott bitsorozat egyenletes véletlen eloszlású, majd ezzel a bitsorozattal dolgozunk tovább. Hashelésről lásd az A.3. fejezetet.

3.1. definíció. Legyen $\widetilde{P}_{\text{bit}}$ a $\{0,1\}$ halmazon vett egyenletes eloszlás által adott valószínűségi mérték és legyen $\widetilde{P}$ ennek ω-adik hatványa, $\widetilde{P}$-t nevezzük az egyenletes valószínűségi mértéknek az $\widetilde{\Omega} = \{0,1\}^\omega$ halmazon, a végtelen bitsorozatok halmazán. Legyen továbbá $\widetilde{P}_n$ a $\widetilde{P}$ n-edik hatványa, $\widetilde{P}_n$-et nevezzük az egyenletes valószínűségi mértéknek $\widetilde{\Omega}^n$-ediken, a végtelen bitsorozatok n tagú sorozatainak halmazán. Legyen $\iota_n : \widetilde{\Omega}^n \to \mathcal{P}(\widetilde{\Omega})$ a halmazképzés művelete, azaz

$$\iota_n((\omega_1, \ldots, \omega_n)) = \{\omega_1, \ldots, \omega_n\}.$$

Legyen ezután $\bar{\Omega}_n \subset \widetilde{\Omega}^n$ a csupa különböző elemekből álló sorozatok részhalmaza, azaz

$$\bar{\Omega}_n = \{\boldsymbol{\omega} \in \widetilde{\Omega}^n : |\iota(\boldsymbol{\omega})| = n\},$$

valamint $\bar{P}_n = \widetilde{P}_n|_{\bar{\Omega}_n}$ $\widetilde{P}_n$ megszorítása erre. $\bar{P}_n$ is valószínűségi mérték lesz, mivel egy elem $\widetilde{\Omega}^n$-ből 1 valószínűséggel csupa különböző elemekből áll. Legyen most $\sim$ a permutálás ekvivalenciarelációja $\bar{\Omega}_n$-en, azaz

$$\boldsymbol{\omega}_1 \sim \boldsymbol{\omega}_2 \iff \iota(\boldsymbol{\omega}_1) = \iota(\boldsymbol{\omega}_2).$$

Faktorizáljuk az $(\bar{\Omega}_n, \bar{P}_n)$ valószínűségi teret $\sim$ szerint, a kapott tér legyen

$$(\Omega_n, P_n) = (\bar{\Omega}_n, \bar{P}_n)/\sim$$

az n számosságú bitsorozatokon az *egyenletes eloszlás*. Az Ω_n elemeit nevezzük a továbbiakban n hosszú vagy számosságú adatfolyamoknak. Legyen még $\Omega = \cup_{n=1}^{\infty}\Omega_n$, ezen valószínűséget nem értelmezünk, pusztán halmaznak tekintjük, Ω elemei az adatfolyamok. Ettől megkülönböztetendő $\widetilde{\Omega}^n$ elemeit adatsorozatoknak nevezzük.

3.2. definíció. *Számláló algoritmusnak* nevezünk egy $(\mathcal{A}, |\mathcal{A}|, o)$ hármast, ahol $|\mathcal{A}|$ pozitív egész. Az $I = \{1, \ldots, |\mathcal{A}|\}$ számhalmazt az algoritmus állapotainak nevezzük, ha

$$\mathcal{A} : I \times \widetilde{\Omega} \to I$$

az algoritmus állapotátmenet-függvénye és

$$o : I \to \mathbb{R}_{\oplus} \qquad 0 = o_1 < o_2 \leq \cdots \leq o_{|\mathcal{A}|}$$

az algoritmus kimenetei, az $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \widetilde{\Omega}^n$ inputra elért végállapotát

$$\mathcal{A}(\boldsymbol{\omega}) = \mathcal{A}(\ldots(\mathcal{A}(1, \omega_1), \omega_2), \ldots, \omega_n)$$

jelöli, és teljesülnek a következők:

1. $\mathcal{A}$ monoton, azaz minden $i \in I$ állapotra és $\omega \in \widetilde{\Omega}$ elemre $o_i \leq o_{\mathcal{A}(i,\omega)}$
2. $\mathcal{A}$ kimenete nem függ sem a bejövő adatok sorrendjétől, sem attól, hogy egy elem hányszor fordul elő az inputban, azaz minden $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \in \widetilde{\Omega}^n$-re

$$\iota(\boldsymbol{\omega}_1) = \iota(\boldsymbol{\omega}_2) \quad \text{esetén} \quad \mathcal{A}(\boldsymbol{\omega}_1) = \mathcal{A}(\boldsymbol{\omega}_2)$$

Egy $\mathcal{A}$ számláló algoritmusra minden n természetes számhoz definiálhatunk egy $\mathcal{A}_n$ valószínűségi változót az (Ω_n, P_n) téren, az algoritmus kimenetét az adott bemenetre, azaz legyen

$$\mathcal{A}_n : \Omega_n \to \mathbb{R}, \quad \mathcal{A}_n(\boldsymbol{\omega}) = o_{\mathcal{A}((\omega_1,\ldots,\omega_n))},$$

ahol $\boldsymbol{\omega}$ az $(\omega_1, \ldots, \omega_n)$ sorozatnak megfelelő adatfolyam, azaz $\iota((\omega_1, \ldots, \omega_n))$ ekvivalenciaosztálya $\sim$-re nézve.

A számlálási feladat ekkor úgy fogalmazható meg, hogy adott N természetes számhoz keresendő olyan $\mathcal{A}$ számláló algoritmus, melynek $|\mathcal{A}|$ állapotszáma kicsi és $1 \leq n \leq N$ esetén $\mathcal{A}_n$ közel esik a konstans n valószínűségi változóhoz. A későbbiekben látjuk majd, hogy ha $E_n\mathcal{A}_n = n$ és $D_n\mathcal{A}_n = O(n)$, akkor $|\mathcal{A}| = \Omega(\log N)$, ahol E_n a várható értéket, D_n pedig a szórást jelenti az (Ω_n, P_n) téren. Itt is és a következőkben log alatt, hacsak más alapot nem írunk ki, a természetes logaritmust érjük és a szokásos jelöléseket használjuk a nagyságrendek jelölésére:

- $f(x) = O(g(x))$ x_0 körül, ha valamely $C > 0$ számra és x_0 egy U környezetére $x \in U$ esetén $|f(x)| \leq Cg(x)$.
- $f(x) = o(g(x))$ x_0 körül, ha létezik $\lim_{x \to x_0} f(x)/g(x)$ és értéke 0.
- $f(x) = \Omega(g(x))$ x_0 körül, ha valamely $C > 0$ számra és x_0 egy U környezetére $x \in U$ esetén $|f(x)| \geq Cg(x)$.
- $f(x) = \Theta(g(x))$ x_0 körül, ha egyszerre $f(x) = O(g(x))$ és $f(x) = \Omega(g(x))$.

4. fejezet

Alsó becslés

Megadunk egy alsó becslést a számláló algoritmusok memóriaigényére, hasonlóan, mint ahogyan 2.1. szakaszban mutattuk meg, hogy N-ig adott relatív hibán belüli számláláshoz $\log_2 \log_2 N$ memória szükséges. Ha ugyanis egy algorimusnak kevesebb mint $O(\log_2 N)$ állapota van, akkor a skatulya-elv miatt van olyan $1 \leq x \leq N$ szám, hogy az algoritmus egyik állapotából nyerhető kimenet sem esik $[x-\sigma x, x+\sigma x]$-be. Ekkor azonban egy x számosságú adatfolyamot nem lehet σ relatív szórással közelíteni, mivel minden előállítható becslés legalább σx-szel különbözik x-től. Ennek pontos megfogalmazása az alábbi tétel.

4.1. tétel. *Ha $\mathcal{A}$ számláló algoritmus, N természetes szám, $0 < c < 1/2$ valós szám és $E_n\mathcal{A}_n = n$, $D_n\mathcal{A}_n \leq cn$ minden $n \leq N$ természetes számra, akkor $|\mathcal{A}| \geq \log N/(-\log(1-2c)) - 1/8$.*

Bizonyítás. Legyen $d(n) = \min_{1\leq i\leq|\mathcal{A}|} |n - o_i|$ és ezzel

$$q = \max_{1\leq n\leq N} d(n)/n \tag{4.1}$$

és legyen n_0 olyan szám, ahol a maximum fölvétetik. A feltételek szerint $E_{n_0}\mathcal{A}_{n_0} = n_0$ valamint

$$D^2_{n_0}\mathcal{A}_{n_0} = E_{n_0}|\mathcal{A}_{n_0} - n_0|^2 \geq (d(n_0))^2,$$

mivel az $|\mathcal{A}_{n_0} - n_0|$ valószínűségi változó minden lehetséges értéke $\geq d(n_0)$, vagyis a $D_n\mathcal{A}_n \leq cn$ feltétel szerint $c^2n_0^2 \geq q^2n_0^2$, azaz $c \geq q$. Legyen most j az az index, amelyikre $o_j \leq N < o_{j+1}$ (ha esetleg $o_{|\mathcal{A}|} \leq N$, akkor $j = |\mathcal{A}|$ és legyen $o_{j+1} = \infty$), ekkor legyen $n_i = \left\lfloor \frac{1}{2}(o_i + o_{i+1}) \right\rfloor \leq o_{i+1}$, ezzel

$$d(n_i) \geq \frac{o_{i+1} - o_i}{2} - 1, \quad \text{így} \quad c \geq q \geq \max_{1\leq i<j} \frac{d(n_i)}{n_i} \geq \max_{1\leq i<j} \frac{o_{i+1} - o_i}{2o_{i+1}} - \frac{1}{o_{i+1}},$$

vagyis minden $1 \leq i < j$-re

$$(1-2c)o_{i+1} \leq o_i + 2,$$

amiből indukcióval adódik, a $d = 1/(1-2c)$ jelöléssel $o_1 = 0$ figyelembevételével, hogy

$$o_j \leq \frac{2}{d-1}(d^j - 1).$$

Ha most esetleg $n_j \leq N$ is igaz, akkor a föntiekben $< j$ helyett mindenütt írható $\leq j$, és így

$$N < o_{j+1} \leq \frac{2}{d-1}(d^{j+1} - 1) \leq \frac{2}{d-1}(d^{|\mathcal{A}|+1} - 1).$$

Ha pedig $n_j > N$, akkor $d(N) = N - o_j$, azaz (4.1) és $c \geq q$ miatt $cN \geq N - o_j$, vagyis

$$N \leq \frac{1}{1-c}o_j \leq \frac{1}{1-c} \cdot \frac{2}{d-1}(d^j - 1) \leq \frac{1}{1-c} \cdot \frac{2}{d-1}(d^{|\mathcal{A}|+1} - 1).$$

Ezekből kifejezve $|\mathcal{A}|$-ra kapjuk, hogy

$$|\mathcal{A}| \geq \frac{1}{-\log(1-2c)} \log\left(\frac{1-c^2}{1-2c}N + 1\right) - 1 = \frac{\log N}{-\log(1-2c)} + \frac{\log(1-c^2)}{-\log(1-2c)}, \tag{4.2}$$

amint azt állítottuk. □

Ez a tétel valójában tehát azt adja nekünk, hogy ha c hibával akarjuk egy legfeljebb N számosságú adatfolyam számosságát becsülni, akkor ehhez

$$\log_2 \log_2 N - \log_2 \log \frac{1}{1-2c} + O(1)$$

memória kell, ahol az $O(1)$ egy minden N-re és c-re korlátos függvényt jelent. Az LogLog-algoritmus azonban (tömörítéses implementálással) $\log_2 \log_2 N + O(1/c^2)$ memóriát igényel, ezért kérdéses, hogy a $\log_2 \log_2 N$ főtag mellett a megengedett hibától függő tagnak mi valójában az optimális nagyságrendje. Most azt mutattuk meg, hogy minden $[(1-2c)x, x]$ intervallumban kell lennie legalább egy állapotának az algoritmusnak, ennél azonban valószínűleg többet is be lehetne bizonyítani, vagyis hogy egy ilyen intervallumban egynél több állapotnak is kell lennie, aminek segítségével éppen ezen a plusz tagon lehetne javítani. Ennek ötlete az lehet, hogy az adatfolyamok nagy részére az adatfolyam végén és a közepénél is c-nél kisebb relatív hibája volna az algoritmusunkból adódó becslésnek.

[1]-ben a szerzők számos más számlálási feladat megoldására szolgáló algoritmusra megadnak nagyságrendi alsó és felső becsléseket.

5. fejezet

A LogLog algoritmus

Ebben a fejezetben Durand és Flajolet [4] LogLog és Super-LogLog algoritmusát elemezzük. Az algoritmusnak egy olyan implementációját adjuk meg, amelynek memóriaigénye aszimptotikusan optimális, illetve elvégezzük a Super-LogLog algoritmus elemzését is.

5.1. A LogLog algoritmus definíciója

A LogLog algoritmus a bementén olvasott adatfolyamból kiszámítja az adatoknak egy *lenyomatát*, majd ebből számít ki egy becslést az adatfolyam számosságára. Első számú szempontunk az lesz, hogy a lenyomat kiszámításához minél kevesebb memória legyen szükséges, második pedig, hogy egy-egy bejövő adatra lehetőleg kevés számítást, memóriaelérést kelljen végezni.

A továbbiakban m fix kettőhatványt fog jelölni, ez a LogLog-algoritmus paramétere, mely egyben vezérli az algoritmus memóriaigényét és pontosságát, e kettő természetesen fordított arányban áll egymással.

5.1. definíció. Legyen $m = 2^k$ és legyen minden $\omega = (x_1, x_2, \dots)$ elemhez $I_m(\omega)$ az $\overline{x_1 x_2 \dots x_k}$ bináris szám valamint $v_m(\omega) = (\omega_{k+1}, \omega_{k+2}, \dots)$ a maradék bitek, legyen az input $\boldsymbol{\omega} = (\omega_1, \dots, \omega_N)$. Legyen $\rho : \widetilde{\Omega} \to \mathbb{N}$ az a függvény, amelyik megmondja, hogy egy $\omega = (x_1, x_2, \dots)$ elemben hányadik helyen áll az első egyes bit, azaz

$$\rho((x_1, x_2, \dots)) = \min\{i : x_i = 1\}.$$

5.2. definíció. Egy $\boldsymbol{\omega} \in \widetilde{\Omega}^n$ adatsorozat *LogLog-lenyomatának* nevezzük a

$$p_m(\boldsymbol{\omega}) = (\max\{\rho(v_m(\omega)) : \omega \in \boldsymbol{\omega}, I_m(\omega)) = 0\}, \dots, \\ \dots, \max\{\rho(v_m(\omega)) : \omega \in \boldsymbol{\omega}, I_m(\omega)) = m-1\})$$

m dimenziós egész vektort.

5.1. megjegyzés. Az egyes koordinátákat néha esetleg vödröknek is nevezzük, egy bejövő adatról pedig azt mondjuk, hogy az i-edik vödörhöz vagy koordinátához került, ha $I_m(\omega) = i - 1$, azaz ha az i-edik koordinátát definiáló maximum argumentumai közé esik.

5.2. megjegyzés. Egy véletlenül választott ω elemre ρ $1/2^k$ valószínűséggel vesz fel k értéket, így nagyjából azt várhatjuk, hogy 2^k darab ω-ra a ρ-k maximuma körülbelül k, azaz p_m egy koordinátája nagyjából becsli azon elemek számának logaritmusát, amelyek hozzá kerültek. Mivel pedig a bejövő adatok véletlenszerűen szóródnak szét az egyes koordinátákhoz, várhatóan minden koordináta értéke $\log(n/m)$ körül lesz. Egy ilyen állítás igaz is, ez motiválja a LogLog algoritmust, ennek bizonyításáról szól az 5.5. szakasz. Hasonló állítás előfordul már például [9]-ben is, de ott nem jut a szerzők eszébe ilyen irányú alkalmazás.

A LogLog-lenyomat következő tulajdonságai a definícióból azonnal látszanak:

5.1. állítás. *p_m monoton, azaz $p_m((\omega_1, \ldots, \omega_n)) \geq p_m((\omega_1, \ldots, \omega_n, \omega_{n+1}))$, ahol a $\geq$-t koordinátánként értjük.*

5.2. állítás. *p_m sem az adatsorozat permutációira, sem az elemek többszöri előfordulására nem változik.*

Bizonyítás. Valóban, egy halmaz maximuma nem változik attól, ha az elemeit más sorrendben vagy egynél többször vesszük. □

Ezek miatt értelmezhető p_m az Ω_n tér fölötti valószínűségi változóként, a továbbiakban így gondolunk rá.

5.3. állítás. *$p_m((\omega_1, \ldots, \omega_n, \omega_{n+1}))$ értéke kiszámítható $p_m((\omega_1, \ldots, \omega_n))$-ből és ω_{n+1}-ből.*

Bizonyítás. Valóban, ha max jelöli a koordinátánkénti maximumképzést és e azt az m dimenziós vektort, melynek $I(\omega_{n+1})$-edik koordinátája $\rho(v_m(\omega_{n+1}))$, a többi pedig 0, akkor

$$p_m((\omega_1, \ldots, \omega_n, \omega_{n+1})) = \max(p_m((\omega_1, \ldots, \omega_n)), e).$$

□

A föntiek miatt a LogLog-lenyomatot kiszámító algoritmusok mind számláló algoritmusok.

Jó tulajdonsága a lenyomatnak, hogy ha két adatfolyam lenyomatát már elkészítettük, akkor az azok összefűzésével kapható adatfolyam lenyomatát is azonnal meg tudjuk kapni, ugyanis igaz ez:

5.4. állítás. *Ha $\boldsymbol{\omega}_1$ és $\boldsymbol{\omega}_2$ két adatfolyam és $\boldsymbol{\omega}$ az ő uniójuk, akkor*

$$p_m(\boldsymbol{\omega}) = \max(p_m(\boldsymbol{\omega}_1), p_m(\boldsymbol{\omega}_2)).$$

Most megadjuk a lenyomatból a becslést elkészítő függvényt, ennek van egy $1 \leq \mu \leq m$ paramétere:

5.3. definíció. A LogLog-algoritmus kimenete a $p_m(\boldsymbol{\omega}) = (M_1, \ldots, M_m)$ lenyomatú adatfolyamon

$$q_\mu(\boldsymbol{\omega}) = \frac{1}{\mu} \sum_{i=1}^{\mu} M_i^*,$$

amiben $(M_1^*, \ldots, M_m^*) = r(M_1, \ldots, M_m)$, ahol r a nagyság szerinti rendezés függvénye, azaz $M_1^* \leq \cdots \leq M_m^*$ és az $\{M_1, \ldots, M_m\} = \{M_1^*, \ldots, M_m^*\}$ multihalmaz értelemben.

5.2. A LogLog algoritmus implementációja

Mivel a LogLog-lenyomatban m darab tetszőlegesen nagy egész számot kell tárolni, ezért annak kiszámítása véges memóriában nem valósítható meg. Ezért korlátozzuk a memóriát, ezzel némi hibát megengedve, de e hibát uralmunk alatt tudjuk tartani. Ehhez először rögzítsünk egy N számot, aminél nagyobb számosságú adatfolyamokra nem kívánjuk alkalmazni az algoritmust, illetve elfogadjuk, hogy N-nél nagyobb adatfolyamokon az algoritmus nem fog használható becslést szolgáltatni.

Ideális LogLog algoritmusnak azt a változatot nevezzük, melyben minden M_i koordináta bármilyen értéket fölvehet.

5.4. definíció. Nevezzünk p-számlálónak egy olyan számlálót, mely a $0, 1, \ldots, p-1$ számokat tudja tárolni, hozzá $\lceil \log_2 p \rceil$ bit memória szükséges. Ha $p-1$-nél nagyobb számot akarunk benne tárolni, értéke legyen $p-1$, és jegyezzük meg, hogy történt benne túlcsordulás.

Flajolet implementációjában az egyes M_i-ket egy-egy $O(\log_2 N)$ számlálóként valósítjuk meg, mi most ennél valamivel jobbat csinálunk. Legyen az implementált LogLog algoritmus az alábbi.

- Legyen A egy $2 \log_2 N$-számláló, kezdetben legyen 0.
- Legyenek $Y_1, \ldots, Y_m$ d-számlálók, kezdetben mind 0-k. d értékét végül $O(\log_2 \log_2 N)$-nek fogjuk választásra érdemesnek találni.

- Minden bejövő ω adatra legyen $i = I(\omega)$ és $r = \rho(v(\omega))$.

  Ha $r \leq A + Y_i$, ne tegyünk semmit.

  Ha $A + Y_i < r < A + d$, akkor legyen $Y_i = r - A$.

  Ha $r \geq A + d$, akkor

  Legyen $y = \min_j Y_j$, legyen minden j-re $Y_j = Y_j - y$ és legyen $A = A + j$, valamint $Y_j = r - A$.

- A kimenet legyen $(A + Y_1, \ldots, A + Y_m)$.

Legyen továbbá

$$q'_\mu = \frac{1}{\mu} \sum_{i=1}^{\mu} A + Y_i^*.$$

5.3. megjegyzés. A fönti algoritmus *nem* számláló algoritmus a 3.2. definíció értelmében, mivel a bejövő adatok sorrendjére érzékeny, hiszen egy Y_i számláló túlcsordulhat egy olyan adattól, amelytől később, ha A már nagyobb, nem csordul túl. így az ismét érkező adattól esetleg ismét nőhet a becslés.Momentumai azonban aszimptotikusan egyenlők az ideális LogLogéval, ami viszont má számláló algoritmus, így őra vonatkozik a 4.1. tétel alsó becslése.

Figyeljük meg, hogy amennyiben nem történt túlcsordulás, úgy a kimenet ugyanaz, mint amit az ideális LogLog adna. Ha ellenben volt túlcsordulás, a kiadott eredmény esetleg kisebb, mint amit az ideális algoritmuból kapnánk. Vizsgáljuk meg most a túlcsordulások előfordulási lehetőségeit.

5.5. állítás. *Legyen adott az algoritmus valamely $(A, Y_1, \ldots, Y_m)$ állapota. Annak valószínűsége, hogy valamelyik Y_j túlcsordul, mielőtt A legalább eggyel megnőne, legfeljebb $4m^2/2^d$.*

Bizonyítás. Jelölje R azt a rossz eseményt, hogy olyan adat érkezik, amelyik valamelyik Y_j-t túlcsordítaná, azaz a $k + 1$. bitjétől kezdve legalább $A + d$ darab 0 van benne. Ennek valószínűsége

$$P(R) = 1/2^{A+d}.$$

Jelölje most J_j azt a jó eseményt, hogy az Y_j értéke pozitív lesz, de nem kell még az A-t változtatnunk. Ez számunkra azért jó, mert amint minden Y_j pozitív, úgy már mielőtt valamelyik túlcsordulna, előbb megnő az A. Ennek valószínűsége

$$P(J_j) = \frac{1/2^{A+1} - 1/2^{A+d}}{m}.$$

Jelölje most a diszjunkt és pozitív valószínűségű A és B eseményekre $A \prec B$ azt, hogy A előbb következik be, mint B. Ha addig veszünk véletlen eseményeket, míg az egyik bekövetkezik, akkor

$$P(A \prec B) = \frac{P(A)}{P(A) + P(B)} \leq \frac{P(A)}{P(B)}.$$

Ha most előbb következik be minden jó esemény mint a rossz, akkor megúsztuk túlcsordulás nélkül, azaz a túlcsordulás valószínűsége legfeljebb

$$1 - P(J_1 \prec R \wedge \cdots \wedge J_m \prec R) = P(R \prec J_1 \vee R \prec J_j) \leq$$
$$\leq \sum_{j=1}^{m} P(R \prec J_j) \leq m \frac{2^{-A-d}}{(2^{-A-1} - 2^{-A-d})/m} = m^2 \frac{1}{2^{d-1} - 1} \leq 4m^2 2^{-d},$$

□

5.6. állítás. *Amennyiben az algoritmus teljes lefutása alatt A nem csordul túl, úgy annak valószínűsége, hogy bármikor föllép túlcsordulás, legfeljebb $8m^2 \log_2 N/2^d$.*

Bizonyítás. Legyen R_a az a rossz esemény, hogy amikor az algoritmus során $A = a$, valamelyik Y_j túlcsordult. Az előző állítás szerint $P(R_a) \leq 4m^2/2^d$, így annak valószínűsége, hogy bármikor hiba lépjen föl, mivel $A < 2\log_2 N$ végig, legfeljebb

$$P(R_0 \vee \cdots \vee R_{2\log_2 N - 1}) \leq \sum_{0 \leq a < 2\log_2 N} P(R_a) \leq 8m^2 \log_2 N/2^d.$$

□

Legyen most $\varepsilon > 0$ fix szám, ő lesz a kívánt hibavalószínűség. Ha most

$$d = 4 + 2\log_2 m + \log_2 \frac{1}{\varepsilon} + \log_2 \log_2 N,$$

akkor $\varepsilon/2$-nél kisebb valószínűséggel fordul elő túlcsordulás az Y_j-kben, ha A nem csordul túl. A-ról pedig a következő igaz:

5.7. állítás. *Ha $N > 2/\varepsilon$, akkor A túlcsordulásának valószínűsége kisebb $\varepsilon/2$-nél.*

Bizonyítás. A akkor csordul túl, ha valamelyik adat elején legalább $2\log_2 N$ darab 0 van, egy ilyen adat előfordulásának valószínűsége $1/N^2$, így annak valószínűsége, hogy a legfeljebb N különböző adat közül legalább az egyik ilyen, legfeljebb $N/N^2 = 1/N$. □

Mindezeket összevetve ha $\varepsilon > 0$-t és N-et veszünk, ahol $N > 2/\varepsilon$ (tipikusan a választott N ennél jóval nagyobb), akkor legalább $1-\varepsilon$ valószínűséggel megkapjuk az ideális LogLog algoritmus eredményét a legfeljebb N számosságú adatfolyamokra

$$1 + \log_2 \log_2 N + m \log_2\big(\log_2 \log_2 N + 4 + 2\log_2 m + |\log_2 \varepsilon|\big) = \\ = (1 + o(1)) \log_2 \log_2 N \quad (5.1)$$

bitnyi memóriában.

Nézzük meg még részletesebben az implementált algoritmusból kapható becslés várható értékét és szórását, hisz ez bármennyire is különbözhetne akár az ideális algoritmus kimenetének momentumaitól még annak ellenére is, hogy e két érték $1-\varepsilon$ valószínűséggel megegyezik.

5.8. állítás. *$E(q_\mu - q'_\mu) < 3\varepsilon \log_2 N$ és $E(q_\mu^2 - {q'_\mu}^2)$-re hasonlóan $O(\varepsilon \log_2^2 N)$ felső becslés igaz.*

Bizonyítás. Tudjuk, hogy ha nem lép fel túlcsordulás, akkor $q'_\mu = q_\mu$, ellenkező esetben pedig $0 \le q'_\mu \le q_\mu$, így mindenesetre

$$E(q_\mu - q'_\mu) \le E(q_\mu | \text{van túlcsordulás}) \cdot P(\text{van túlcsordulás}) \le \\ \le E(q_\mu | A \text{ túlcsordul}) \cdot P(A \text{ túlcsordul}) + (2\log_2 N + d)\varepsilon/2.$$

Legyen most M_i az ideális LogLog-algoritmusban számított lenyomat i-edik koordinátája. Mivel $A \le M_i$ minden i-re, ezért A csak úgy csordulhat túl, ha minden $M_i \ge 2\log_2 N$, emiatt

$$E(q_\mu | A \ge 2\log_2 N) = \\ = \frac{1}{P(A \ge 2\log_2 N)} \sum_{x_i \ge 2\log_2 N} (x_1 + \cdots + x_m) P(M_1 = x_1, \ldots, M_m = x_m) \le \\ \le \frac{1}{P(A \ge 2\log_2 N)} \left(\sum_{x_1 \ge 2log_2 N} x_1 P(M_1 = x_1) \right)^m,$$

ahol útközben használtuk, hogy $P(A_1 \wedge \cdots \wedge A_m) \le P(A_i)$, valamint hogy az egyes M_i-k azonos eloszlásúak (bár nem függetlenek). A szumma becsléséhez:

$$\sum_{x_1 \ge c} x_1 (P(M_1 > x_1 - 1) - P(M_1 > x_1)) = \\ = cP(M_1 > c-1) + \sum_{M_1 \ge c} P(M_1 > x_1) = \frac{2c+4}{2^c},$$

ebből

$$E(q_\mu | A \geq 2\log_2 N) \cdot P(A \geq 2\log_2 N) \leq \left(\frac{4\log_2 N + 4}{N^2}\right)^m \leq \frac{1}{N^m},$$

legalábbis elég nagy N-ekre. Összesítve tehát elég nagy N-ekre

$$E(q_\mu - q'_\mu) < 3\varepsilon \log_2 N.$$

□

Hogyha a föntebb írtakban $\varepsilon = 1/\log_2^3 N$-et veszünk, akkor elég nagy N-ekre azt kapjuk, hogy $(1+o(1))\log_2\log_2 N$ memóriában ki tudunk számítani egy értéket, mely 1-hez tartó valószínűséggel megegyezik az ideális algoritmus kimenetével, sőt várható értékük és szórásuk is aszimptotikusan egyenlő.

5.3. Implementálás tömörítéssel

Lehetne egy kicsit még nyerni a helyen úgy, ha az Y_j-ket tömörítve tárolnánk, úgy valószínűleg (5.1)-ben $\log_2\log_2 N + m\log_2|\log_2\varepsilon| + O(1)$ volna mondható volna, hiszen ekkor egy Y_j-re nem $\log_2\log_2\log_2 N$ memória jutna, hanem csupán konstans méretű, mely az Y-ok közös eloszlásának entrópiájával aszimptotikusan egyenlő. Ezen becslés részleteit azonban nem dolgozzuk ki, mivel az Y_j-k tömörített tárolása esetén egyetlen Y_j megváltoztatása a fönti algoritmusban szükségesnél sokkal több számítást és memóriaelérést igényelne.

5.4. Kapcsolat az alsó becsléssel

Korábban láttuk, hogy legalább $(1-o(1))\log_2\log_2 N$ memória kell ahhoz, hogy fix pozitív százaléknyi relatív hibán belül becsülni tudjuk egy adatfolyam számosságát, így a fönt írt módszer aszimptotikusan már nem javítható. A fönti módszer a [4]-ben adottól annyiban különbözik, hogy nem az egyes M_i értékeket tárolja $O(\log_2\log_2 N)$ biten, hanem csak az A számot, a többi különbséget ennél aszimptotikusan kevesebb helyen tudja kezelni, így $O(m\log_2\log_2 N)$ helyett csak $(1+o(1))\log_2\log_2 N$ bitet használ, aminél kevesebb pedig nem is lehet elegendő.

5.5. A LogLog lenyomat egy koordinátájának viselkedése

Most megvizsgáljuk, hogy milyen eloszlása van a LogLog lenyomat egy koordinátájának. A LogLog-algoritmus működésének alapja az 5.9 állítás, a teljes algoritmus erre az egy érdekes megfigyelésre épül. Maga az állítás geometria valószínűségi változók maximumának eloszlásáról szól és már régen ismert volt, azonban Flajolet vette észre, hogy ez alapján lehet számláló algoritmust készíteni.

5.5. definíció. $R : \Omega \to \mathbb{N} \quad R(\{\omega_1, \ldots, \omega_n\}) = \max_{1 \le i \le n} \rho(\omega_i)$, azaz egy $\boldsymbol{\omega} \in \Omega_n$ sorozatra $R(\boldsymbol{\omega})$ az $\boldsymbol{\omega}$ elemeinek ρ-értékeinek maximuma. Legyen $R_n = R|_{\Omega_n}$, R_n az (Ω_n, P_n) téren egy valószínűségi változó.

Számoljuk ki ezen R_n eloszlását. Legyen minden k természetes számra $A_i \subset \widetilde{\Omega}^n$ az az esemény, hogy

$$A_i = \{\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \widetilde{\Omega}^n : \rho(\omega_i) \le k.$$

Legyen továbbá

$$A = \bigcap_{i=1}^{n} A_i \quad \text{és} \quad \bar{A} = A \setminus \{\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \widetilde{\Omega}^n : |\iota(\boldsymbol{\omega})| < n\}.$$

Itt az A-ból kivont halmaz $\widetilde{P}_n$-re nullmértékű, ezért $\widetilde{P}_n(A) = \widetilde{P}_n(\bar{A})$, és mert A a permutációra zárt, azért $P_n(R_n \le k) = P_n(A) = \bar{P}_n(A) = \widetilde{P}_n(A)$. Másrészt A a független és azonos valószínűségű $\rho(\omega_i) \le k$ események szorzata és mert $\widetilde{P}_1(\rho(\omega) \le k) = 1 - \frac{1}{2^k}$, úgy a föntiek szerint

$$P_n(R_n \le k) = \left(1 - \frac{1}{2^k}\right)^n. \tag{5.2}$$

Lássuk az ígért állítást, miszerint a LogLog-lenyomat koordinátái valóban közel esnek az adott koordinátához kerülő elemek számának logaritmusához. Ennek bizonyítása során egyúttal meglátjuk azt a módszert, amit a tovább elemzésekhez is használni fogunk, a Poisson és Mellin transzformáltak együttes használatát.

5.9. állítás.

$$E_n R_n = \log_2 n + \frac{\gamma}{\log 2} + \frac{1}{2} + f(\log n) + o(1)$$

valamint

$$D_n^2 R_n = \frac{\pi^2}{6\log^2 2} + \frac{1}{12} + g(\log n) + o(1),$$

ahol $f(x)$ és $g(x)$ 1 szerint periodikus függvények, $|f(x)| < 1{,}5\cdot 10^{-4}$, $|g(x)| < 5{,}5\cdot 10^{-4}$.

Bizonyítás. Legyen $\varepsilon_n = E_n R_n$ és $\varphi_n = E_n R_n^2$, a definíció szerint

$$\varepsilon_n = \sum_{k=1}^{\infty} k P_n(R_n = k) = \sum_{k=1}^{\infty} k(P_n(R_n \geq k) - P_n(R_n \geq k+1)),$$

ebből Abel-átrendezéssel, beírva még a $0P_n(R_n \geq 0) = 0$ tagot és figyelmbevéve, hogy $P_n(R_n \leq 0) = 0$, kapjuk, hogy

$$\begin{aligned}\varepsilon_n &= \sum_{k=1}^{\infty}(k-(k-1))P_n(R_n \geq k) = \sum_{k=1}^{\infty} 1 - P_n(R_n < k) = \\ &= \sum_{k=0}^{\infty} 1 - P_n(R_n \leq k).\end{aligned}$$

Hasonlóan kapjuk, hogy

$$\varphi_n = \sum_{k=1}^{\infty}(k^2-(k-1)^2)P_n(R_n \geq k) = \sum_{k=0}^{\infty}(2k+1)(1-P_n(R_n \leq k)).$$

Legyen most ε_n és φ_n Poisson-transzformáltja $\widetilde{\varepsilon}(z)$ és $\widetilde{\varphi}(z)$, ezekre

$$\begin{aligned}\widetilde{\varepsilon}(z) &= \sum_{n=0}^{\infty}\frac{z^n e^{-z}}{n!}\varepsilon_n = \sum_{n=0}^{\infty}\frac{z^n e^{-z}}{n!}\sum_{k=0}^{\infty}\left(1-\left(1-\frac{1}{2^k}\right)^n\right) = \\ &= e^{-z}\sum_{k=0}^{\infty}\sum_{n=0}^{\infty}\frac{z^n}{n!} - \frac{(z(1-2^{-k}))^n}{n!} = e^{-z}\sum_{k=0}^{\infty} e^z - e^{z-z/2^k} = \\ &= \sum_{k=0}^{\infty} 1 - e^{-z/2^k},\end{aligned}$$

valamint hasonlóan

$$\widetilde{\varphi}(z) = \sum_{k=0}^{\infty}(2k+1)\left(1-e^{-z/2^k}\right).$$

Itt az átrendezések jogosak voltak, mivel csupa pozitív tagok fordulnak elő. Vezessük be a $\tau(z) = 1 - e^{-z}$ függvényt, ezzel

$$\widetilde{\varepsilon}(z) = \sum_{k=0}^{\infty} \tau(z/2^k) \quad \text{és} \quad \widetilde{\varphi}(z) = \sum_{k=0}^{\infty} (2k+1)\tau(z/2^k).$$

A $\tau(z)$ függvény nagyságrendje a 0-ban $O(z)$, a végtelenben pedig $O(1)$, ezért az ő alapsávja $S_\tau = \langle -1, 0\rangle$. A Mellin transzformáció alkalmazásához még határozzuk meg a $\sum_{k=0}^{\infty} 2^{-ks}$ és a $\sum_{k=0}^{\infty}(2k+1)2^{-ks}$ Dirichlet-sorok konvergenciafélsíkját. Ez az a félsík, ahol $|2^{-s}| = 2^{-\Re(s)} < 1$, azaz a $\langle 0, \infty\rangle$ sáv, így mondhatjuk, hogy a $\langle -1, 0\rangle$ sávban

$$\widetilde{\varepsilon}^*(s) = \left(\sum_{k=0}^{\infty} 2^{ks}\right) \tau^*(s) \quad \text{és} \quad \widetilde{\varphi}^*(s) = \left(\sum_{k=0}^{\infty} (2k+1)2^{ks}\right) \tau^*(s),$$

ahol $\tau^*(s) = -\Gamma(s)$, azaz az összegzéseket elvégezve

$$\widetilde{\varepsilon}^*(s) = \frac{1}{2^s - 1}\Gamma(s) \quad \text{és} \quad \widetilde{\varphi}^*(s) = -\frac{2^s + 1}{(2^s - 1)^2}\Gamma(s).$$

Ezen képletekből látható, hogy a transzformáltak meromorfan kiterjednek a teljes síkra, szingularitásokkal a negatív egész helyeken és a $2k\pi i$ helyeken minden k egészre.

Az A.3 alkalmazásával azt kapjuk, hogy

$$\widetilde{\varepsilon}(x) = \frac{\log x}{\log 2} + \frac{\gamma}{\log 2} + \frac{1}{2} - \sum_{k\neq 0} \frac{\Gamma(s_k)}{\log 2} e^{(-2k\pi \log x)i} + O(x^{-\delta}).$$

Itt az utolsó előtti szummában $e^{(-2k\pi \log z)i}$ $\log z$-ben 1 szerint periodikus, azaz ha

$$f(x) = \sum_{k\neq 0} \frac{\Gamma(s_k)}{\log 2} e^{(-2k\pi x)i},$$

akkor ez a szumma éppen $f(\log x)$, ez lesz a feladatban írt f függvény, becslésére numerikusan adódik

$$|f(x)| \leq \frac{1}{\log 2} \sum_{k\neq 0} |\Gamma(2k\pi i)| < 1{,}5 \cdot 10^4.$$

Végül az $\varepsilon_n = \widetilde{\varepsilon}(n) + o(1/\log n)$ becslést depoissonizálással kapjuk meg, ehhez meg kell becsülnünk $\widetilde{\varepsilon}(z)$ nagyságát egy, a valós tengelyt tartalmazó szögtartományban, valamint azon kívül is. Legyen ez a szögtartomány a $\theta = \pi/4$-hez tartozó, a pozitív valós tengelyre szimmetrikus negyedsík. A szögtartományon belül a következő becslés mondható:

5.10. állítás. *Ha* $|\arg z| \le \pi/2$, *akkor* $|\widetilde{\varepsilon}(z)| \le 2\log_2|z| + 10$.

Bizonyítás. Legyen $z = x + iy$, a feltétel ekkor azt jelenti, hogy $x \ge 0$. A háromszög-egyenlőtlenség alapján így kezdhetjük a becslésünket:

$$|\widetilde{\varepsilon}(z)| \le \sum_{k=0}^{\infty} |1 - e^{-z/2^k}| = \sum_{x/2^k \le 1} |1 - e^{-z/2^k}| + \sum_{x/2^k > 1} |1 - e^{-z/2^k}|.$$

A második szumma tagjait a triviális $|1-e^{-z/2^k}| \le 1+e^{-x/2^k} \le 2$-vel becsülve kapjuk, hogy

$$|\widetilde{\varepsilon}(z)| \le \sum_{x/2^k \le 1} |1 - e^{-z/2^k}| + 2(\log_2 x + 1) \le \sum_{x/2^k \le 1} |1 - e^{-z/2^k}| + 2\log_2|z| + 2,$$

mivel legfeljebb $\log_2 x + 1$ darab nemnegatív k-ra állhat $x/2^k > 1$. Az első szumma tagjainak becsléséhez pedig legyen $0 < a = e^{-x/2^k} \le 1$ és $\varphi = y/2^k$, ekkor a becslendő mennyiség $|1-ae^{i\varphi}| \le |1-a|+|a-ae^{i\varphi}| \le 1-a+|1-e^{i\varphi}|$. Mivel 1 és $e^{i\varphi}$ az egységsugarú körön egymástól φ távolságra fekszik, ezért $|1 - e^{i\varphi}| \le |\varphi| = |y|/2^k$, azaz végül

$$|1 - e^{-z/2^k}| \le 1 - e^{-x/2^k} + x/2^k \le (x + |y|)/2^k \le 2|z|/2^k,$$

és így

$$|\widetilde{\varepsilon}(z)| \le \sum_{x/2^k \le 1} 2|z|/2^k + 2\log_2|z| + 2 = 2\log_2|z| + 2 + 4|z|/x \le 2\log_2|z| + 10.$$

□

A szögtartományon kívül a következő teljesül:

5.11. állítás. *Ha* $|\arg z| > \pi/4$ *és* $|z|$ *elég nagy, akkor* $|e^z\widetilde{\varepsilon}(z)| < e^{7|z|/8}$.

Bizonyítás. Legyen ismét $z = x + iy$, és legyen $|z|$ olyan nagy, hogy $4|z| < e^{|z|/8}$ teljesüljön. Először az $x \ge 0$ esetet vizsgáljuk. Ekkor $x > |y|$ igaz, mivel a szögtartományon kívül vagyunk, vagyis $x < |z|/\sqrt{2} < 3|z|/4$. Az előző bizonyításban írtak szerint ekkor $|\tau(z/2^k)| \le (x + |y|)/2^k \le 2|z|/2^k$, így

$$|e^z\widetilde{\varepsilon}(z)| \le e^x \cdot 4|z| < e^{7|z|/8}.$$

Ha pedig $x < 0$, akkor

$$|e^z\widetilde{\varepsilon}(z)| \le \sum_{k=0}^{\infty} |e^z(1 - e^{-z/2^k})| = \sum_{k=0}^{\infty} e^{x(1-1/2^k)}|e^{z/2^k} - 1| \le$$

$$\le \sum_{k=0}^{\infty} |e^{z/2^k} - 1| = O(\log|z|) < e^{7|z|/8},$$

elég nagy $|z|$-re, itt használtuk az 5.10 állítást z helyében $-z$-vel. □

E két állítás együtt az A.5. tétel szerint azt adja, hogy minden $\beta > 0$-ra

$$\varepsilon_n = \widetilde{\varepsilon}(n) + O(n^{\beta-1}),$$

ahogy azt vártuk.

Nézzük most $\widetilde{\varphi}^*(s)$-et. Az A.4 szerint

$$\widetilde{\varphi}(z) = \frac{\log^2 z}{\log^2 2} + \left(\frac{2\gamma}{\log^2 2} + \frac{1}{\log 2}\right)\log z + \left(\frac{\pi^2/6+\gamma^2}{\log^2 2} + \frac{\gamma}{\log 2} + \frac{1}{3}\right) - \\ - \sum_{k\neq 0}\left(\frac{2\Gamma(s_k)}{\log^2 2}\log z - \frac{\Gamma(s_k)}{\log 2}\right) z^{-s_k} + O(z^{-\delta}),$$

itt $\widetilde{\varphi}(z)$ kifejtésének utolsó előtti tagja $(1-2\log_2 z)f(\log z)$.

Ez az eredmény a fönti az 5.10 és az 5.11 állításokban írtakhoz nagyon hasonlóan depoissonizálható. Ennek alkalmazásával tehát írhatjuk, hogy ezekre jutottunk eddig

$$\varepsilon_n = \log_2 n + \frac{\gamma}{\log 2} + \frac{1}{2} - f(\log n) + o(1/\log n)$$
$$\varphi_n = \log_2^2 n + \left(\frac{2\gamma}{\log 2} + 1\right)\log_2 n + \left(\frac{\pi^2/6+\gamma^2}{\log^2 2} + \frac{\gamma}{\log 2} + \frac{1}{3}\right) - \\ + (1-2\log_2 n)f(\log n) + o(1/\log n).$$

Ebből kapjuk a szórásnégyzetre $D_n^2 R_n = \varphi_n - \varepsilon_n^2$ alapján, hogy

$$D_n^2 R_n = \frac{\pi^2}{6\log^2 2} + \frac{1}{12} + \left(2 + \frac{2\gamma}{\log 2}\right) f(\log n) - f(\log n)^2 + o(1),$$

ami az állítás volt, ha $g(x) = \left(2 + \frac{2\gamma}{\log 2}\right) f(x) - f(x)^2$-et választunk. □

5.4. megjegyzés. A jelenlevő pici, de elkerülhetetlen f és g fluktuációk a feladatot alapvetően nem elemivé teszik. Megjegyzendő továbbá, hogy ha ebből gyártunk algoritmust, akkor a kimenet nem torzítatlan becslése $\log_2 n$-nek, hanem a teljes precízség kedvéért korrigálnunk kell az f hatását is. Ehhez legyen $l(y)$ az $y = x + f(x) + \gamma/\log 2 + 1/2$ függvény inverze, ekkor az $l(R_n)$ már legalább aszimptotikusan torzítatlan $\log_2 n$-re, mivel f periodikus torzítását kompenzáltuk. Mivel azonban f nagyon kicsi, általában elég, ha l-et pusztán lineáris függvénynek tekintjük, azaz f hatását elhanyagoljuk.

Az l inverz függvény létezik, mivel $E_n R_n$ szigorúan növő, hiszen $P_n(R_n \leq k)$ szigorúan csökken.

5.6. A LogLog algoritmus elemzése

Megadjuk most, hogy hogyan becsülje a LogLog-algoritmus az adatfolyam nagyságát a lenyomatból kapott q_μ szám alapján. Maga a q_μ szám a számosság logaritmusához áll közel, így vagy őt (vagy ha különösen precízek akarunk lenni, akkor az apró fluktuációkra is javított változatát) használjuk $\log n$ becslésére, vagy 2^{q_μ}-t n becslésére. Az előbbi mellett szól az, hogy nagy μ-kre q_μ aszimptotikusan normális, így az ő szórása n nagyságrendjének pontatlanságát jelzi, éppen amilyen jellegű információt mi n-ről használni akarunk. A másik mellett szólna, hogy Flajolet azt elemezte, és az magára n-re ad becslést. Mi most mégis az első mellett maradunk. Megvizsgáljuk tehát q_μ eloszlását nagy n, m, μ-kre.

Megjegyezzük, hogy általában q_μ és 2^{q_μ} eloszlása között nem tudunk közvetlen kapcsolatot mondani, hiszen például föntebb láttuk, hogy $ER_n \sim \log n$, ezzel szemben $E2^{R_n} = \infty$.

Először poissonizáljuk a LogLog-lenyomatot, hogy a lenyomat koordinátái egymástól függetlenné váljanak.

5.6. definíció. Legyen m fix, legyen $\lambda > 0$ paraméter és legyen N λ paraméterű Poisson eloszlású, valamint legyen $p^{(n)}$ a LogLog-lenyomat vektorvalószínűségiváltozó az Ω_n téren, végül legyen $p_{(\lambda)} = p^{(N)}$. p-t ekkor a λ paraméterű poissonizált LogLog-lenyomatnak nevezzük.

5.12. állítás. *$p_{(\lambda)}$ koordinátái független azonos eloszlású valószínűségi változók.*

Bizonyítás. Először meghatározzuk $p^{(n)}$ eloszlását.

Ehhez még előbb meghatározzuk a $\boldsymbol{\omega}_i = \{\omega \in \boldsymbol{\omega} : I(\omega) = i\}$ halmazok méretének együttest eloszlását, azaz a $p^{(n)}$ egyes koordinátáiban szereplő definiáló maximumok argumentumszámainak együtteseloszlását. Mivel az n darab bejövő adat függetlenül egyenlő valószínűséggel kerül az egyes koordinátákhoz, így ez az eloszlás m-edrendű polinomiális, n paraméterrel. Egy koordináta eloszlását pedig már ismerjük annak függvényében, hogy mennyi elem került hozzá, ha ν_i elemet kapott, akkor ez az eloszlás R_{ν_i} eloszlása, amelyet (5.2)-ban írtunk föl. Ezen eloszlások különböző i-kre ráadásul egymástól függetlenek, mivel $I(\omega)$ és $v(\omega)$ függetlenek. Így tehát ismerjük $p^{(n)}$ koordinátáinak együttes eloszlását, ez

$$P_n(p^{(n)} \le (t_1, \dots, t_m)) = \sum_{\substack{\nu_1,\dots,\nu_m \ge 0 \\ \nu_1+\dots+\nu_m=n}} \binom{n}{\nu_1, \dots, \nu_n} \frac{1}{m^n} \prod_{i=1}^{m} \left(1 - \frac{1}{2^{t_i}}\right)^{\nu_i},$$

ahol $\leq$ alatt koordinátánkénti kisebb vagy egyenlőséget értünk. Rövíditésnek legyen $h(t,\nu) = (1-1/2^t)^\nu - (1-1/2^{t-1})^\nu$, ezzel

$$P_n(p^{(n)} = (t_1,\dots,t_m)) = \sum_{\substack{\nu_1,\dots,\nu_m\geq 0\\ \nu_1+\dots+\nu_m=n}} \binom{n}{\nu_1,\dots,\nu_n} \frac{1}{m^n} \prod_{i=1}^m h(t_i,\nu_i).$$

Most ebből számoljuk ki $p_{(\lambda)}$ eloszlását.

$$P(p_{(\lambda)} = (t_1,\dots,t_m)) = \sum_{n=0}^\infty \frac{e^{-\lambda}\lambda^n}{n!} P_n(p^{(n)} = (t_1,\dots,t_m)) =$$

$$= \sum_{\nu_1,\dots,\nu_m\geq 0} \frac{e^{-\lambda}\lambda^{\nu_1+\dots+\nu_m}}{(\nu_1+\dots+\nu_m)!} \frac{(\nu_1+\dots+\nu_m)!}{\nu_1!\dots\nu_m!} \frac{1}{m^{\nu_1+\dots+\nu_m}} \prod_{i=1}^m h(t_i,\nu_i) =$$

$$= \prod_{i=1}^m \sum_{\nu_i\geq 0} \frac{e^{-\lambda/m}(\lambda/m)^{\nu_i}}{\nu_i!} h(t_i,\nu_i) = \prod_{i=1}^m \left(e^{-\frac{\lambda/m}{2^{t_i}}} - e^{-\frac{\lambda/m}{2^{t_i-1}}}\right),$$

ami azt mutatja, hogy $p_{(\lambda)}$ koordinátái valóban függetlenek és azonos eloszlásuk. □

Jelölje $R_{(\lambda/m)}$ az $R_{N/m}$ valószínűségi változót, az ő várható értéke a föntebbi számolások szerint $\widetilde{\varepsilon}(\lambda/m)$, szórásnégyzete $\widetilde{\varphi}(\lambda/m) - \widetilde{\varepsilon}^2(\lambda/m)$, ezek aszimptotikáját pedig már ismerjük.

Legyen a továbbiakban $0 < \beta \leq 1$ fix szám, legyen

$$\mu = \lfloor \beta m \rfloor,$$

legyen $p_{(\lambda)} = (M_1,\dots,M_m)$ és legyen

$$q_{(\lambda)} = \frac{1}{\beta} \sum_{i=1}^\mu M_i^*.$$

q-t a β-csonkolt becslésnek fogjuk nevezni, mivel csupán a β legkisebb elem alapján számolunk becslést, a nagyobb elemek hozzájárulásának hiányát pedig az $1/\beta$ tényezővel kompenzáljuk. Ha még ezen túl $q^{(n)}$ jelöli a $p^{(n)}$ koordinátáiból kapott hasonló csonkolt becslést, akkor igaz, hogy $q_{(\lambda)}$ és $q^{(N)}$ eloszlása megegyezik, ebből kifolyólag azonosak a momentumaik is, mivel pedig $q^{(N)}$ várható értéke és szórásnégyzete $q^{(n)}$ várható értékének és szórásának Poisson-transzformáltja, ha $q_{(\lambda)}$ momentumait ismerjük, abból depoissonizációval megkaphatjuk $q^{(n)}$ momentumait is.

Lássuk tehát $q_{(\lambda)}$ vizsgálatát. Előbb külön megvizsgáljuk a $\beta = 1$ esetet, amit Flajolet LogLog-algoritmusnak nevez, majd később térünk a $\beta < 1$ esetre, mely Flajoletnál a super-LogLog nevet viseli. Ekkor $q_{(\lambda)}$ m darab független $R_{(\lambda/m)}$ eloszlású valószínűségi változó átlaga, így a föntebb írt tétel alapján megállapíthatjuk, hogy

5.1. tétel. *Ha $\beta = 1$, akkor*

$$Eq_{(\lambda)} = \log_2 \frac{\lambda}{m} + \frac{\gamma}{\log 2} + \frac{1}{2} + f\left(\log \frac{\lambda}{m}\right) + o(1),$$

valamint

$$D^2 q_{(\lambda)} = \frac{1}{m}\left(\frac{\pi^2}{6\log^2 2} + \frac{1}{12} + g\left(\log \frac{\lambda}{m}\right) + o(1)\right),$$

ahol f és g ugyanazok, mint a föntebbi tételben, mindezeken túl pedig $q_{(\lambda)}$ eloszlása aszimptotikusan normális, amint $m \to \infty$.

Bizonyítás. Független, azonos eloszlású valószínűségi változókra a tétel jól ismert. □

A fönti tételből megállapíthatjuk, hogyha igazán precízen akarjuk egy adatfolyam számosságának logaritmusát becsülni, akkor őt $q - \gamma/\log 2 - 1/2$ helyett $l(q) - \gamma/\log 2 - 1/2$-del becsüljük, ahol $l(y)$ az $y = x + f(x)$ függvény inverze. Ezen inverz biztosan létezik, mivel $x + f(x)$ szigorúan növő függvény, mert n növekedtével R_n várható értéke szigorúan nő, hiszen az eloszlásfüggvénye szigorúan csökken. Ha az adatfolyam számosságára kívánunk becslést kapni, akkor precíz becslést az $L(q) = 2^{l(q)-\gamma/\log 2-1/2}$ valószínűségi változó nyújt, melynek eloszlása aszimptotikusan lognormális, amint $m \to \infty$. Valójában a $2^{l(q)-\gamma/\log 2-1/2}$ változó momentumai kiszámíthatóak a már látott módszerekkel teljesen pontosan is, ennek részletei [4]-ben megtalálhatók.

A fönti eredmény depoissonizálása már nem igényel újabb meggondolásokat, hiszen éppen ezen függvényekre mutattuk meg korábban, hogy a depoissonizációs tétel alkalmazható rájuk.

Nézzük most a $\beta < 1$ esetet, ezt Stigler [13]-ban írt tétele alapján tudjuk kezelni. Az ottani tétel a mi esetünkre specializálva a következőt mondja:

5.2. tétel. *Legyen b a legkisebb olyan egész, amelyre $P(R_{(\lambda/m)} \leq b) \geq \beta$, tegyük föl, hogy itt szigorú egyenlőtlenség áll. Legyen*

$$e = \frac{1}{\beta}\left(\sum_{k=0}^{b-1} kP(R_{(\lambda/m)} = k) + b\left(\beta - P(R_{(\lambda/m)} \leq b-1)\right)\right) \tag{5.3}$$

és

$$d^2 = \frac{1}{\beta}\left(\sum_{k=0}^{b-1} k^2 P(R_{(\lambda/m)} = k) + b^2\left(\beta m - P(R_{(\lambda/m)} \leq b-1)\right)\right) - e^2. \quad (5.4)$$

Ekkor $(q_{(\lambda)} - e)\sqrt{m}$ *eloszlásban tart* Y*-hoz amint* m *tart a végtelenbe, ahol* Y *eloszlása* 0 *várható értékű,* d^2/β^2 *szórásnégyzetű normális.*

5.5. megjegyzés. Az (5.3) és (5.4)-beli e és d^2 annak a valószínűségi változónak a várható értéke és szórásnégyzete, amelynek eloszlásfüggvénye az $R_{(\lambda/m)}$ eloszlásfüggvényének $1/\beta$-szorosának és 1-nek a minimuma.

A tételben szereplő b paraméter értékét ki tudjuk számolni.

$$b = \left\lceil \log_2 \frac{\lambda}{m} - \log_2 \log \frac{1}{\beta} \right\rceil = \log_2 \frac{\lambda}{m} - c_\beta + \delta(\beta, \lambda),$$

Ahol $c_\beta = \log_2(-\log \beta)$ és $0 \leq \delta < 1$ olyan, hogy b egész legyen. Legyen továbbá

$$a_k = e^{-\frac{\lambda/m}{2^k}},$$

ezzel $P(R_{(\lambda/m)} = k) = a_k - a_{k-1}$, $P(R_{(\lambda/m)} \leq k) = a_k$. Figyeljük még meg, hogy

$$a_b = \beta^{w(\beta,\lambda)},$$

ahol $w(\beta, \lambda) = 2^{-\delta(\beta,\lambda)}$, $1/2 < w(\beta, \lambda) \leq 1$, valamint hogy

$$a_k^{2^s} = a_{k-s}.$$

e és d^2 számolásához úgy kezdhetünk, mint azt tettük $\widetilde{\varepsilon}$ és $\widetilde{\varphi}$ esetében, Abel-átrendezést alkalmazunk. Vezessük be

$$u : (0,1) \to \mathbb{R} \quad u(x) = \sum_{k=1}^{\infty} x^{2^k}$$

függvényt, ezzel írható, hogy

$$e = b - \frac{1}{\beta}\sum_{k=1}^{b} a_b^{2^b} = b - \frac{1}{\beta}u(a_b) + o(1),$$

valamint a

$$v : (0,1) \to \mathbb{R} \quad v(x) = \sum_{k=1}^{\infty} (2k-1)x^{2^k}$$

függvénnyel

$$d^2 + e^2 = b^2 - \frac{1}{\beta}(2bu(a_b) - v(a_b)) + o(1),$$

amiből végül azt kapjuk, hogy

$$d^2 = \frac{1}{\beta}v(a_b) - \frac{1}{\beta^2}u^2(a_b).$$

Azt kaptuk, tehát, hogy w befutja $(1/2, 1]$-et, amint λ változik, és

$$e = b - \frac{1}{\beta}u(\beta^{-w}) + o(1) \tag{5.5}$$

$$d^2 = \frac{1}{\beta}v(\beta^{-w}) - \frac{1}{\beta^2}u^2(\beta^{-w}) + o(1). \tag{5.6}$$

Hasonlóan a korábbi megfigyelésünkhöz e egy monoton növő függvénytől $o(1)$ additív hibával különbözik, hiszen $Eq^{(n)}$ monoton növő. Emiatt tehát létezik az

$$y = \log_2 x - \log_2 m - \log_2(-\log\beta) + \delta(\beta, x) - \beta u\left(\beta^{-2^{-\delta(\beta,x)}}\right)$$

függvénynek inverze, legyen az $l_\beta(y)$, így $l_\beta(q_{(\lambda)})$ aszimptotikusan torzítatlan becslése λ-nak.

Az optimális β pedig az 5.2. tétel és (5.6) alapján az lesz, melyre

$$\frac{d^2}{\beta^2} = \sup_{w\in(1/2,1]} \frac{1}{\beta^3}v(\beta^{-w}) - \frac{1}{\beta^4}u^2(\beta^{-w})$$

minimális. Numerikusan kiszámolva ez a minimum $\beta^* \approx 0{,}762$ esetében vétetik föl.

A. Függelék

Segédeszközök

A.1. Mellin transzformáció

A Mellin transzformáció a Fourier és Laplace transzoformációhoz hasonló operáció, segítségével harmonikus összegek kezelhetők analitikusan, azaz $\sum_k \lambda_k f(\mu_k x)$ alakú összegek. Nem teljes általánosságban foglalkozunk a Mellin transzformációval, mert mi csak speciális eseteit használjuk, azonban a használt speciális esetekben az általános tételeknél erősebb tételekre van szükségünk, ezért úgyis külön bizonyítást kellen írnunk. Az általános tételek [6]-ban találhatók meg.

A.1.1. A Mellin transzformáció

A.1. definíció. Legyen $f : (0, \infty) \to \mathbb{R}_\oplus$ folytonos nemnegatív függvény, ekkor az ő *Mellin-transzformáltja* a komplex s helyen az

$$f^*(s) = \int_0^\infty f(x)x^{s-1}dx \tag{A.1}$$

improprius integrál.

Mivel kompakt intervallumon folytonos függvény mindig integrálható, ezért a Mellin transzformált létezése és végessége pontosan azzal ekvivalens, hogy mind a $\lim_{t\to 0} \int_t^1 f(x)x^{s-1}dx$, mind a $\lim_{t\to\infty} \int_1^t f(x)x^{s-1}dx$ határértékek léteznek és végesek, ehhez pedig az kell, hogy ezen sorozatok Cauchy-sorozatok legyenek, azaz mind

$$\lim_{\substack{t\to 0\\ t'\to 0}} \int_t^{t'} f(x)x^{s-1}dx = 0 \quad \text{mind} \quad \lim_{\substack{t\to\infty\\ t'\to\infty}} \int_t^{t'} f(x)x^{s-1}dx = 0 \tag{A.2}$$

igaz legyen.

A.1. állítás. *Ha $f : (0,\infty) \to \mathbb{R}_\oplus$ folytonos függvény, $s = \sigma + it$ és $f^*(\sigma)$ létezik és véges, akkor $f^*(s)$ is létezik és véges.*

Bizonyítás. A fönt írtak miatt föltevésünk azt jelenti, hogy $0 \leq \int_t^{t'} f(x)x^{\sigma-1}dx \to 0$, mivel pedig

$$0 \leq \left| \int_t^{t'} f(x)x^{s-1}dx \right| \leq \int_t^{t'} |f(x)||x^{s-1}|dx = \int_t^{t'} f(x)x^{\sigma-1}dx \to 0,$$

kapjuk az állítást. □

Megjegyezzük, hogy valós s-ekre $\int_0^1 f(x)x^{s-1}dx$ és $\int_1^\infty f(x)x^{s-1}dx$ mindig léteznek és nemnegatívok, esetleg azonban végtelenek. Mivel azonban $x \in (0,1)$ esetén $f(x)x^{s-1}$ s-ben monoton csökkenő, $x \in (1,\infty)$-ben pedig monoton növő, így ha

$$\begin{aligned} \alpha &= \inf\left\{ s \in \mathbb{R} : \int_0^1 f(x)x^{s-1}dx < \infty \right\}, \\ \beta &= \sup\left\{ s \in \mathbb{R} : \int_1^\infty f(x)x^{s-1}dx < \infty \right\}, \end{aligned} \tag{A.3}$$

akkor az $s \in (\alpha,\beta)$ valósokra mindkettő véges, azaz a Mellin transzformált ezen pontokban létezik és véges. Az előző állítással együtt így bebizonyítottuk, hogy

A.2. állítás. *Ha $f : (0,\infty) \to \mathbb{R}_\oplus$ folytonos függvény, α és β* (A.3) *szerintiek, akkor az*

$$\langle \alpha, \beta \rangle = \{ s \in \mathbb{C} : \Re(s) \in (\alpha,\beta) \}$$

(esetleg üres) sáv minden pontjában létezik és véges f Mellin transzformáltja. E sávot f alapsávjának *hívjuk és S_f-fel jelöljük.*

Polinomiális nagyságrendű függvényekre ki tudjuk számolni az alapsávot.

A.3. állítás. *Ha $f(x) = O(x^a)$ 0-körül és $f(x) = O(x^b)$ ∞ körül, akkor $\langle -a, -n \rangle$ része az alapsávnak.*

Bizonyítás. A feltételek szerint valamely pozitív c_1, c_2 számokra $f(x) \leq c_1 x^a$ $0 < x < 1$-ben és $f(x) \leq c_2 x^b$ $x > 1$ esetén, ebből pedig

$$\begin{aligned} &\int_0^1 f(x)x^{s-1}dx \leq c_1 \int_0^1 x^{a+s-1}dx < \infty, && \text{ha } a+s-1 > -1 \text{ és} \\ &\int_1^\infty f(x)x^{s-1}dx \leq c_2 \int_1^\infty x^{b+s-1}dx < \infty, && \text{ha } b+s-1 < -1, \end{aligned}$$

azaz a (A.3)-beli α-ra és β-ra $\alpha \leq -a$ és $\beta \geq -b$. □

A.1. példa. Az $f(x) = e^{-x}$ függvény nemnegatív, folytonos $(0, \infty)$-n és a 0 körül $f(x) = O(x^0)$, míg a végtelen körül minden $\gamma > 0$-ra $f(x) = O(x^{-\gamma})$[1], ezért minden $\langle 0, \gamma \rangle$ része az alapsávjának, tehát alapsávja $\langle 0, \infty \rangle$, azaz ezen a féltéren Mellin transzformáltja létezik, ez a $\Gamma(s)$ függvény.

A Mellin-transzformált hasznos tulajdonsága, hogy a lineáris kombináció átmegy rajta, azaz igaz a következő

A.4. állítás. *Ha f és g a $(0, \infty)$-n értelmezett folytonos nemnegatív függvény, λ és μ tetszőleges komplex számok, valamint az S sáv része alapsávjaik metszetének, akkor $S \subseteq S_{f+g}$ és minden $s \in S$-re*

$$(\lambda f + \mu g)^*(s) = \lambda f^*(s) + \mu g^*(s).$$

Bizonyítás. S tulajdonsága miatt a transzformáltat definiáló integrálok léteznek, az integrálás pedig lineáris. □

A függvények átskálázása is követhetően változtatja a Mellin-transzformáltat, nevezetesen igaz a következő:

A.5. állítás. *Ha $f : (0, \infty) \to \mathbb{R}_\oplus$ folytonos nemnegatív függvény és $\lambda > 0$ valós, $g(x) = f(\lambda x)$, akkor $S_g = S_f$ és $s \in S$-re*

$$g^*(s) = \lambda^{-s} f^*(s).$$

Bizonyítás. A definiáló integrálok ismét mind léteznek, és az $y = \lambda x$ helyettesítéssel adódik, hogy

$$\int_0^\infty f(\lambda x) x^{s-1} dx = \lambda^{-s} \int_0^\infty f(y) y^{s-1} dy.$$

□

Legyen most $f : (0, \infty) \to \mathbb{R}_\oplus$ egy folytonos nemnegatív függvény és legyen

$$h(s) = \sum_{k=1}^{\infty} c_k k^{-s}$$

egy Dirichlet-sor, melyben $c_k \geq 0$ valósok. Tudjuk, hogy a Dirichelt-sorok abszolút konvergenciájának tartománya egy félsík, jelölje ezt esetünkben $S_h = \langle -c, \infty \rangle$. Az f függvényből a h szerint képzett harmonikus sornak nevezzük az

$$F(x) = \sum_{k=1}^{\infty} c_k f(kx)$$

sort. Tegyük fel, hogy ez egy nemnegatív folytonos függvénye x-nek a pozitív félegyenesen. Ekkor ennek Mellin-transzformáltjáról szól a következő tétel.

[1]Ezt jelöljük $f(x) = O(x^{-\infty})$-nel.

A.1. tétel. *A fönti feltételek mellett $s \in S_f \cap S_h$-ra*

$$F^*(s) = h(s)f^*(s).$$

Bizonyítás. Mivel az $S_f \cap S_h$ sávban minden előforduló tag nemnegatív, így jogos az

$$\int_0^\infty \sum_{k=0}^\infty c_k f(kx)x^{s-1}ds = \sum_{k=0}^\infty c_k \int_0^\infty f(kx)x^{s-1}dx$$

átalakítás, amiből pedig a tétel azonnal adódik. □

A.1.2. Az inverz Mellin transzformáció

Az inverz Mellin-transzformációról szóló klasszikus tételt csak idézzük:

A.2. tétel. *Ha $f : (0,\infty) \to \mathbb{R}_\oplus$ folytonos függvény, alapsávja $\langle \alpha, \beta \rangle$, $c \in (\alpha, \beta)$, akkor minden $x > 0$-ra*

$$f(x) = \lim_{T\to\infty} \frac{1}{2\pi i} \int_{c-iT}^{c+iT} f^*(s)x^{-s}ds.$$

A fönti tétel alapján kapcsolatot fogunk találni $f(x)$ aszimptotikus viselkedése és $f^*(s)$ szingularitásai között. Ezt csak az alkalmazott speciális esetre írjuk le az alábbi tételben, ami a [6]-ben írt technikát használja, de mégis néhány ponton változtatni kell a bizonyításon és kihasználni a transzformált speciális tulajdonságait.

A.3. tétel. *Ha az $f : (0,\infty) \to \mathbb{R}_\oplus$ folytonos függvény Mellin-transzformáltja*

$$f^*(s) = \frac{1}{2^s - 1}\Gamma(s),$$

akkor minden pozitív delta számra igaz, hogy

$$f(x) = \frac{\log x}{\log 2} + \frac{\gamma}{\log 2} + \frac{1}{2} - \sum_{k\neq 0} \frac{\Gamma(s_k)}{\log 2} e^{(-2k\pi \log x)i} + O(x^{-\delta}),$$

amint $x \to \infty$.

Bizonyítás. Minden n nemnegatív számra legyen T_n az $(1/2, -(2n+1)\pi i) \to (1/2, (2n+1)\pi i) \to (\delta, (2n+1)\pi i) \to (\delta, -(2n+1)\pi i)$ zárt út, ekkor Cauchy tétele szerint

$$\frac{1}{2\pi i} \int_{T_n} f^*(s)x^{-s}ds = \frac{1}{2\pi i} \sum_{|k|\leq n} \mathrm{Res}_{s=2k\pi i}\, f^*(s)x^{-s},$$

ami pedig megegyezik az állításban szereplő főtaggal. Másrészt pedig

$$\left|\frac{1}{2\pi i}\int_{T_n} f^*(s)x^{-s}ds - \frac{1}{2\pi i}\int_{1/2-(2n+1)\pi i}^{1/2+(2n+1)\pi i} f^*(s)x^{-s}ds\right| \leq$$

$$\leq \int_{1/2}^{\delta}(|f^*(s+(2n+1)\pi i)| + |f^*(s-(2n+1)\pi i)|)ds + \\ + \int_{\delta-(2n+1)\pi i}^{\delta+(2n+1)\pi i} |f^*(s)x^{-s}|ds \leq (*).$$

Tudjuk, hogy ha $\Re\sigma \in [1/2, \delta]$, akkor létezik olyan $c > 1$ szám, hogy $\Gamma(\sigma + it) = O(c^{-|t|})$, valamint ha $t \equiv \pi \pmod{2\pi}$, akkor $1/(2^{\sigma+it}-1) = O(1)$, így a fönti egyenlőtlenség így folytatható:

$$(*) \leq (\delta + 1/2)c^{-(2n+1)\pi} + x^{-\delta}\int_{-\infty}^{\infty} c^{-|t|}ds = O(c^{-(2n+1)\pi}) + O(x^{-\delta}).$$

Amint n-nel tartunk a végtelenbe, úgy kapjuk a tétel állítását. □

A fönti tételhez teljesen hasonlóan lehet az alábbit is belátni.

A.4. tétel. *Ha az $f : (0,\infty) \to \mathbb{R}_{\oplus}$ folytonos függvény Mellin-transzformáltja*

$$f^*(s) = -\frac{2^s+1}{(2^s-1)^2}\Gamma(s),$$

akkor minden pozitív delta számra igaz, hogy

$$f(x) = \frac{\log^2 z}{\log^2 2} + \left(\frac{2\gamma}{\log^2 2} + \frac{1}{\log 2}\right)\log z + \left(\frac{\pi^2/6+\gamma^2}{\log^2 2} + \frac{\gamma}{\log 2} + \frac{1}{3}\right) - \\ - \sum_{k\neq 0}\left(\frac{2\Gamma(s_k)}{\log^2 2}\log z - \frac{\Gamma(s_k)}{\log 2}\right)z^{-s_k} + O(z^{-\delta}),$$

amint $x \to \infty$.

A.2. Poisson transzformáció

A.2. definíció. Egy g_n számsorozat *Poisson-transzformáltja* a

$$\widetilde{g}(z) = e^{-z}\sum_{n=0}^{\infty}\frac{g_n}{n!}z^n$$

függvény.

Megfelelő feltételek mellett egy sorozat n-edik tagja és a sorozat Poisson-transzformáltja az n helyen aszimptotikusan megegyezik. Ennek mögöttes oka az, hogy az n paraméterű Poisson eloszlás az n számnak $n^{1/2+\varepsilon}$ sugarú környezetében koncentrálódik, így ha a sorzat tagjai nem változnak gyorsan, akkor a sorozat tagjainak az ilyen súlyokkal vett átlaga közel lesz a sorozat n-edik tagjához. Ezt a következő tétel írja le:

A.5. tétel. *Ha a g_n sorozat $\widetilde{g}(z)$ Poisson-transzformáltja egész függvény és léteznek olyan $R > 0$, $\beta > 0$, $0 < \alpha < 1$, $0 < \theta < \pi/2$ számok, hogy a következő két feltétel teljesül:*

(I) $|z| > R$, $-\theta \leq \arg z \geq \theta$ *esetén* $\widetilde{g}(z) = O(|z|^{\beta})$,

(O) $|z| > R$, $|\arg z| > \theta$ *esetén* $e^{z}\widetilde{g}(z) = O(e^{\alpha|z|})$,

akkor $g_n - \widetilde{g}(n) = O(n^{\beta-1})$.

Ennek bizonyítása megtalálható [8]-ban.

A.3. Hashelés

A [3] alapján közöljük, hogyan lehet expliciten megadni nagyon jó hash-családokat.

A.3. definíció. $(N; n, m)$ *hash-családnak* hívunk egy (F, X, Y) hármast, ahol X és Y n ill. m elemű halmazok és $F = (f_1, \ldots, f_N)$ N darab $f_i : X \to Y$ függvény sorozata.

A.4. definíció. Egy $(N; n, m)$ hash-családot *ε-univerzálisnak* nevezünk, ha bármely két különböző $x, x' \in X$ elemre azon

$$|\{f \in F : f(x) = f(x')\}| \leq \varepsilon N.$$

A.5. definíció. Egy $(N; n, m)$ hash-családot *ε-erősen univerzálisnak* nevezünk, ha

- Minden $x \in X$ és $y \in Y$ elemekre

$$|\{f \in F : f(x) = y\}| = N/m.$$

- Minden különböző $x, x' \in X$ és nem feltétlenül különböző $y, y' \in Y$ elemekre

$$|\{f \in F : f(x) = y, f(x') = y'\}| \leq \varepsilon N/m.$$

A.6. állítás. *Ha egy $(N; n, m)$ család ε-erősen univerzális, akkor ő ε-univerzális is.*

Bizonyítás.

$$\{f \in F : f(x) = f(x')\} = \bigcup_{y \in Y}^{*} \{f \in F : f(x) = y, f(x') = y\},$$

ahol $\bigcup^*$ a diszjunkt unió, ezért

$$|\{f \in F : f(x) = f(x')\}| = \sum_{y \in Y} |\{f \in F : f(x) = y, f(x') = y\}| \leq$$
$$\leq \sum_{y \in Y} \varepsilon N/m \leq \varepsilon N,$$

lévén Y m elemű. □

A.7. állítás. *Egy $(N; n, m)$ ε-univerzális családból véletlenül vett f függvényre fix $x \neq x' \in X$ mellett*

$$P(f(x) = f(x')) \leq \varepsilon.$$

A.8. állítás. *Egy $(N; n, m)$ ε-erősen univerzális családból véletlenül vett f függvényre fix $x \neq x' \in X$, $y, y' \in Y$ mellett*

$$P(f(x') = y' | f(x) = y) \leq \varepsilon.$$

Bizonyítás. Azon f-ek száma, melyekre $f(x) = y$, pontosan N/m, azoké pedig, melyekre még emellett $f(x') = y'$, legfeljebb $\varepsilon N/m$, így a feltételes valószínűség definíciója rögtön adja az állítást. □

A fönti állítások szerint tehát egy erősen univerzális hash-családból ha véletlenül veszünk egy függvényt, akkor az olyan véletlenül szórja szét X-et Y-ba, hogy „valószínűtlen”, hogy két elemet ugyanoda vigyen, másrészt „szinte független” az, hogy különböző elemeket hova visz.

A hashelés hitelesítésre is használható, azaz egy üzenetet, X egy x elemét küldjük valakinek, akivel megállapodtunk egy titkos $f \in F$ elemben, ehhez kellett nekünk $\log_2 N$ bit információt titkosan kicserélni. Ekkor az x üzenet mellé aláírásként elküldjük $f(x)$-et is. Ha a hash-család erősen univerzális volt, akkor ebben az $(x, f(x))$ párban bármi változtatás csak legfeljebb ε valószínűséggel ad továbbra is érvényes $(x, f(x))$ párt, tehát kicsi ε mellett ez jól használható hitelesítésre. Persze elengedhetetlen, hogy $\log_2 N$ bitet teljes titokban cseréljünk a partnerrel. A hitelesítésre használt hash-családokra a következőket várjuk ezért el:

- Legyen N kicsi, mondjuk $2^1 00$ körül. (100 bit teljes titokban küldendő információ)
- Legyen ε kicsi, mondjuk 2^{-20} körül. (Az ellenség bármit tesz, egymilliomod eséllyel vág át minket)
- Legyen m kicsi, $1/\varepsilon$ körül. (Ne kelljen hosszú aláírásokat küldüzgetni)
- Legyen n nagy, mondjuk gyakorlatilag végtelen. (Hosszú üzenetet se kelljen darabonként aláírni)

Ha ellenben a hasheléssel az adatok véletlenszerűsítését kívánjuk megvalósítani, akkor más követelményeink vannak. N kicsisége ekkor nem fontos, a (véletlenszerűen) kiválasztott hash-függvényt ugyanis egyszer és mindenkorra beépíthetjük az algoritmusba. A jól használhatóság szempontjából ε kicsisége lényeges m-hez képest.

A.9. állítás. *Egy $(N; n, m)$ ε-erősen univerzális hash-családban $\varepsilon \geq 1/m$.*

Bizonyítás. Mivel fix $x \neq x'$-re és y-ra

$$\{f \in F : f(x) = y\} = \bigcup_{y' \in Y}^{*} \{f \in F : f(x) = y, f(x') = y'\},$$

úgy $N/m \leq \sum_{y' \in Y} \varepsilon N/m = \varepsilon N$, amiből tényleg $1/m \leq \varepsilon$ jön. □

Most rátérünk jó hash-család gyártására.

A.6. tétel. *Legyenek β, γ pozitív egészek, legyen $\alpha = \beta + \gamma$. Létezik $(2^{3\beta+\gamma}; 2^{2^\gamma \alpha}, 2^\beta)$ $2^{1-\beta}$-erősen univerzális $H_{\beta,\gamma}$ hash-család.*

Látjuk, hogy a fönti család nagyon jó, hiszen itt $\varepsilon = 2/m$, ami az optimumnak csak kétszerese, $3\beta + \gamma$ bitnyi kulcsot használ, és körülbelül $2^{-\gamma}$ részre nyomja össze az inputot.

A.1. megjegyzés. A továbbiakban a $GF(2^t)$ testeket fogjuk használni, ezért minden t-re fixáljunk egy, lehetőleg minél kisebb súlyú $p_t(x) \in GF(2)[x]$ t-edfokú irreducibilis polinomot, és azonosítsuk $GF(2^t)$-t $GF(2)[x]/p_t(x)$-szel. Ekkor $GF(2^t)$ elemeire gondolhatunk úgy is, mint a $GF(2)$ fölötti legfeljebb $t-1$-edfokú polinomkra, és úgy is mint egy t bites számra, az $a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ polinomot pedig azonosíthatjuk az $a_0 a_1 \ldots a_{t_1}$ számmal, ekkor az összeadás a szokásos, míg a szorzás a $\mod p_t(x)$ vett polinomszorzás.

A.6. definíció. Legyenek $t' > t$ pozitív egészek, ekkor legyen $\varphi : GF(2^{t'}) \to GF(2^t)$ az a szürjekció, ami az $a_0 + a_1 x + \cdots + a_{t'-1} x^{t'-1}$ polinomhoz az $a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$ polinomot rendeli, ez a bináris számok között egy $t' - t$ bites jobbra tolásnak felel meg.

Most megadjuk $H_{\beta,\gamma}$ elemeit. Legyen minden $k_1, k_2 \in GF(2^\alpha)$, $k_3 \in GF(2^\beta)$ számhármashoz

$$f_{k_1,k_2,k_3} : GF(2^\alpha)^{2^\gamma} \to GF(2^\beta)$$
$$f_{k_1,k_2,k_3}(x_0, x_1, \ldots, x_{2^\gamma-1}) = \varphi\left(k_2\left(x_0 + x_1 k_1^{-1} + \cdots + x_{2^\gamma-1} k_1^{-(2^\gamma-1)}\right)\right) + k_3.$$

Praktikusan a polinom értékét Horner-elrendezéssel lehet számolni, az x_i bemenetdarabok úgyis sorban jönnek, ha pedig az input nincs $2^\gamma\alpha$ bit hosszú, akkor a maradékot föl lehet tölteni nullákkal, úgy, hogy a polinomot nem számoljuk tovább. A bemenetet α bites egységekben célszerű venni, így α értékét értelmes dolog 8-cal oszthatónak választani.

Az elméleti optimumhoz nagyon közeli hash-függvényt tehát a következő módon lehet gyártani. Kigondoljuk, hogy hány bitről hány bitre akarunk hashelni, veszünk olyan β, γ-t, amire $2^\gamma(\beta + \gamma)$ ill. β ezekhez közel esik, választok véletlenszerűen k_1, k_2, k_3 számokat, és veszem az f_{k_1,k_2,k_3} függvényt.

Gyakorlatban ez a következőt jelenti. Figyelem, itt $\cdot$ a $GF(2^\alpha)$ beli szorzás.

- Legyen $w = 0$.
- Legyen $i = 0$.
- Amíg $i < 2^\gamma$, addig
  Legyen $w = w \cdot k_1^{-1} + x_i$
  Legyen $i = i + 1$.
- Legyen $w = k_2 \cdot w$.
- Legyen $w = w \gg \gamma$.
- Legyen $w = w + k_3$.
- Kimenet: w.

Irodalomjegyzék

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, February 1999.

[2] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and qunatiles over sliding windows. In *Proceedings of the 23th PODS*, 2004.

[3] Jürgen Bierbauer. Introduction to codes and their use (manuscript), http://www.math.mtu.edu/~jbierbra/homezeugs/codecourse.ps.

[4] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA03)*, 2003.

[5] Cristian Estan, George Varghese, and Mike Fisk. Bitmap algorithms for counting active flows on high speed links. In *Proceedings of Internet Measurment Conference*, 2003.

[6] P. Flajolet, X. Gourdon, and P. Dumas. Mellin transforms and asymptotics : Harmonic sums. *Theoretical Computer Science*, 141(1-2):3–58, 1995.

[7] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, October 1985.

[8] Philippe Jacquet and Wojciech Szpankowski. Analytical depoissonization and its applications. *Theoretical Computer Science*, 201(1-2):1–62, 1998.

[9] Samuel Karlin and Howard M. Taylor. *Sztochasztikus folyamatok*. Gondolat, 1985.

[10] Richard M. Karp, Christos H. Papadimitriou, and Scott Shenker. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. on Database Systems*, March 2003.

[11] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, November 1982.

[12] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, October 1978.

[13] Stephen M. Stigler. The asymptotic distribution of the trimmed mean. *The Annals of Statistics*, 1(3):472–477, 1973.