Otokset tietokannoista

Lähteet

- Jeffrey Scott Vitter: Faster Methods for Random Sampling, *Communications of ACM* 27, 7, 703–718, 1984.
- Numerical Recipes in C, luku 7.3

Luennon runko

- käydään läpi neljä algoritmia joilla voidaan tehdä otantaa tietokannoista
- näistä viimeinen on tehokkain
- se hyödyntää edellisellä luennolla käsiteltyjä transformaatio- ja hylkäämismenetelmiä
- näin saadaan kuvaa näistä menetelmistä myös realistisemmassa ympäristössä kuin viime luennon alkeis/leikkiesimerkeissä

Johdanto

- usein tarpeellinen tehtävä on valita n tietuetta satunnaisesti tietokannasta, jossa N tietuetta, $n < N$
- talletettuna toissijaiseen muistiin tms. hitaalle välineelle
- kriittinen seikka on kuinka hyvin voidaan välttää turhat luku/kirjoitusoperaaatiot
- halutaan saada tuloksia, jotka pätevät koko aineistoon, joutumatta käymään läpi koko aineistoa
- ei samaa tietuetta kahdesti
- myös tietuekoko voi olla suuri
- esim. laaduntarkkailu, ostoskorianalyysi

Perusmenetelmä

1. **do** {
2. generoi satunnainen kokonaisluku $1 \leq k \leq N$
3. valitse k:s tietue, ellei sitä ole jo valittu
4. } **until** n tietuetta valittu

Perusmenetelmä (2)

- vaatii $O(n)$ satunnaisluvun generoimista
- generointiaika $O(n)$, mikäli 3. askelen vaatima tarkistus tapahtuu vakioajassa
- tämä vaatii tilaa $O(N)$ jos tarkistukseen käytetään binääristä taulukkoa, tai $O(n)$ osoitinta hajauttamalla

Sijaintijärjestys

- algoritmi valitsee tietueet ottamatta huomioon niiden sijaintijärjestystä tiedostossa
- esim. 84. tietue voi tulla valituksi ennen 16. tietuetta
- jos otoksen tietueet halutaan poimia järjestyksessä, tietuenumerot täytyy lajitella ennen poimintaa
- lajitteluaika $O(n \log n)$, esim. quicksort, heapsort
- poimintaa ei voida aloittaa, ennen kuin kaikki tietuenumerot on generoitu
- myöskään tulostusta ei voi tehdä on-line

Otanta järjestyksessä

- generoidaan otokseen tulevien tietueiden numerot suoraan järjestyksessä (on-line)
- ratkaisuperiaate:
- kun N tietueesta on valittava n kpl, kunkin yksittäisen tietueen valinnan todennäköisyys on n/N
  - N on vielä käsittelemättömien tietueiden määrä
  - n on otokseen vielä tarvittavien tietueiden määrä
  - jokaisen tietueen kohdalla valinta todennäköisyydellä n/N (generoidaan satunnaisluku k väliltä 1,…,N ja tietue otetaan otokseen jos $k < n$)
  - kun $n = 0$, algoritmi päättyy

Algoritmi S

1. **do** {
2. generoi satunnaisluku $u \sim Tas(0,1)$
3. **if** $NU > n$ aseta $N \leftarrow N-1$ ja siirry kohtaan 5.
4. valitse järjestyksessä seuraava tietue otokseen
5. aseta $n \leftarrow n-1, N \leftarrow N-1$
6. } **while** $n > 0$

Algoritmin S analyysia

- jos jossakin vaiheessa $n = N$ kaikki jäljellä olevat tietueet valitaan otokseen
- koska jokaisen tietueen kohdalla on tehtävä päätös, otetaanko se mukaan vai ei $\rightarrow O(N)$-aikavaativuus

Etäisyyksien generointi

- nopeammissa algoritmeissa generoidaan suoraan peräkkäisten mukaan otettavien tietueiden etäisyyksiä:
  - merkitään $S(n,N)$:llä satunnaismuuttujaa, joka kertoo, kuinka monta seuraavaa tietuetta hylätään
  - ratkaisuperiaate: toistetaan seuraavia askelia, kunnes $n = 0$
  1. generoidaan satunnaisluku $S(n,N)$
  2. hylätään seuraavat $S(n,N)$ tietuetta
  3. $N \leftarrow N - S(n,N) - 1, n \leftarrow n-1$

Satunnaismuuttuja $S(n,N)$

- diskreetti satunnaismuuttuja, joka saa arvoja väliltä $0,\ldots,N-n$
- merkitään seuraavassa lyhyesti S
- pistetodennäköisyysfunktio $P(S=s) = f(s)$ ilmaisee, mikä on todennäköisyys, että sivuutetaan s arvoa
- jakaumafunktio $P(S(n,N) \leq s) = \sum_0^{N-n} f(s) = F(s)$ ilmaisee, mikä on todennäköisyys että sivuutetaan enintään s arvoa

Algoritmi A

- seuraava algoritmi perustuu siihen, että $f(s) = F(s) - F(s-1)$ ja $f(0) = F(0) = n/N$

1. **do** {
2. generoi satunnaisluku $u \sim Tas(0,1)$
3. **while** $u > F(s)$ sivuuta
4. valitse ensimmäinen tietueista jolle $u \leq F(s)$
5. aseta $n \leftarrow n-1, N \leftarrow N - S(n,N) - 1$
6. } **while** $n > 0$

Algoritmin A analyysiä

- do-silmukka suoritetaan n kertaa
- kohta 3 vie aikaa luokkaa $O(N)$, koska joudutaan käymään läpi tietueita järjestyksessä, kunnes löytyy se tietue jolle $u \leq F(s)$
- algoritmin aikavaativuus on siis $O(N)$ (kuten Algoritmi S)
- generoitavien satunnaislukujen määrä on n eli huomattavasti pienempi

Hylkäämismenetelmä diskreeteille jakaumille

- seuraavassa algoritmissa käytetään hyväksymis/hylkäämismenetelmää, jota käsiteltiin edellisellä luennolla
- siinä yhteydessä esiteltiin menetelmä vain generoitaessa otoksia jatkuvista jakaumista
- hieman muuntaen kyseistä menetelmää voi kuitenkin yhtä hyvin soveltaa diskreettien jakaumien tapauksessa

Hylkäämismenetelmä (2)

- muunnetaan diskreetti pistetodennäköisyysfunktio jatkuvaksi porrasfunktioksi (paloittain vakioksi funktioksi)
- portaan pituus on 1 ja funktion arvo i:nnellä portaalla on pistetodennäköisyys $p(i-1)$
- ks. kuva Numerical Recipes, s. 294
- hylkäysmenetelmässä käytetään vain kokonaislukuosaa $\lfloor x \rfloor$ reaaliarvoisesta satunnaisluvusta x

Hylkäämismenetelmä S:n valitsemiseksi

- tehtävä siis: generoi satunnaislukuja S:n jakaumasta
- olkoon X satunnaismuuttuja jonka jakauma approksimoi jakaumaa $F(s)$ "hyvin"
- etsitään $c \geq 1$ jolle $f(\lfloor x \rfloor) \leq cg(x),$ kaikille x, relevantilla arvoalueella
- generoidaan jakaumasta g kandidaatti x
- generoidaan satunnaisluku u väliltä $0, \ldots, 1$
- jos $u > f(\lfloor x \rfloor / (cg(x)))$ hylätään $\lfloor x \rfloor$
- muuten hyväksytään $x \rightarrow$ sivuutettavien tietueiden lukumäärä

Litistys (*squeeze method*)

- ehdon $u > f(\lfloor X \rfloor)/cg(X)$ testaaminen vie jonkin verran aikaa
- hylkäämisen todennäköisyys on pieni

$\Rightarrow$ aikaa voidaan säästää korvaamalla $f(s)$ funktiolla $h(s)$ joka voidaan laskea nopeammin ja jolle $h(s) \leq f(s)$

- $f(\lfloor X \rfloor)$ täytyy laskea vain kun $u > h(\lfloor X \rfloor)/cg(X)$
- litistystekniikka: $h(\lfloor X \rfloor) \leq f(\lfloor X \rfloor) \leq cg(X)$
- litistys ei tietysti ole välttämätöntä algoritmin toiminnan kannalta

Algoritmi (hylkäämismenetelmä)

1. **do** {
2. generoi satunnaisluku $u \sim Tas(0, 1)$
3. generoi satunnaisluku x jonka pistetn/tiheys on $g(x)$
4. **if** $u \leq h(\lfloor x \rfloor)/(cg(x)))$ aseta $S \leftarrow \lfloor x \rfloor$
5. **else if** $u \leq f(\lfloor x \rfloor)/(cg(x)))$ aseta $S \leftarrow \lfloor x \rfloor$
6. **else goto 2**
7. sivuuta S tietuetta ja valitse sen jälkeen seuraava otokseen
8. aseta $n \leftarrow n - 1, N \leftarrow N - S(n, N) - 1$
9. } **while** $n > 0$

Sopivan funktion g valinta

- mikä olisi funktio g joka olisi sopivan lähellä S:n jakaumaa?
- S:n arvo voidaan mieltää pienimmäksi n:stä *kokonaisluvusta*, jotka on generoitu toisistaan riippumattomasti väliltä $0, \ldots, N$
- generoidaan otos "vastaavasta" jatkuvasta jakaumasta
- mikä on "vastaava" jatkuva jakauma?
- olkoon X satunnaismuuttuja joka noudattaa kyseistä jakaumaa
- X:n arvo voidaan ajatella pienimmäksi n:stä riippumattomasta *reaaliluvusta* Tas$(0, N)$-jakaumasta

Kandidaattien generointi

- kandidaatin x generoiminen jakaumasta g on helppoa ja nopeaa
- käytetään transformaatiomenetelmää
- merkitään $Z_1, Z_2, \ldots, Z_n$ riippumattomia Tas$(0, N)$-jakautuneita reaalilukuja
- näitä lukuja ei suinkaan tarvitse generoida, niitä käytetään tässä vain g:tä vastaavan jakaumafunktion $G(x)$ johtamiseen

$$G(x) = P(X \leq x) \;=\; 1 - P(X > x)$$

$$=\; 1 - \prod_{1 \leq k \leq n} P(Z_k > x) = 1 - (1 - \frac{x}{N})^n.$$

Kandidaattien generointi (2)

- haluttu x saadaan siis transformaatiomenetelmällä $x = G^{-1}(u)$
- edellyttää tietysti että sellainen on (tässä sivuutetaan sen näyttäminen intuitioon vedoten, samoin tekninen käänteisfunktion johtaminen)
- saadaan $G^{-1}(u) = N(1 - (1 - u)^{1/n})$
- koska muuttuja $1 - U$ on tasaisesti jakautunut kun U:kin on, saadaan kandidaatti x asettamalla

$$x \leftarrow N(1 - u^{1/n})$$

Muut parametrit

- myös vakio c ja litistämiseen käytettävä funktio h valittava
- h oltava sellainen että se pystytään laskemaan nopeasti
- sivuutetaan ne tässä yhteydessä

Algoritmin analyysiä

- kun n on suuri suhteessa N:ään, algoritmi saattaa olla hitaampi kuin algoritmi A
- do-silmukka suoritetaan n kertaa
- enää ei tarvitse käydä sen sisällä läpi tietueita vaan sivuutettavien tietueiden lukumäärä saadaan generoitua suoraan satunnaislukuna vastaavan satunnaismuuttujan jakaumasta
- algoritmin aikavaativuus on siis $O(n)$

Yhteenveto

- usein tarpeellinen tehtävä on valita otos satunnaisesti tietokannasta
- turhien luku-/kirjoitusoperaatioiden välttäminen tärkeää
- siksi tietueet on syytä valita järjestyksessä jossa ne on talletettu
- hyväksymis/hylkäämismenetelmään perustuva algoritmi suoriutuu tehtävästä lineaarisessa ajassa suhteessa otoksen kokoon