Algoritmi in podatkovne strukture – 2

Slovar

izvedba z Bloomovim filtrom

Nenatančni slovar

Imamo slovar S nekakšnih elementov. S tem slovarjem želimo početi vsaj naslednje operacije:

iskanje: `Find(`S`, x) --> y` – v slovarju S poiščemo element `x`. Rezultat `y` je Boolova vrednost `true` ali `false`. Želimo si, da je odgovor praviloma pravilen – dovolimo občasno lažne pozitivne odgovor *false positive*.

Morda želimo še početi:

dodajanje: `Insert(`S`, x)` – v slovar S dodamo nov element `x`.

Odpovemo pa se (za sedaj):

izločanje: `Delete(`S`, x) --> y` – iz slovarja S izločimo element `x`. Rezultat `y` je lahko Boolova vrednost `true` ali `false`, ki sporoči ali je bil element uspešno izločen ali ne, ali pa operacija ničesar ne vrne.

Nenatančni slovar – s štetjem

Tokrat imamo operaciji:

dodajanje: `Insert(`S`, x)` – v slovar S dodamo element `x` – ni nujno nov!

iskanje: `Find(`S`, x) --> y` – v slovarju S poiščemo element `x`. Rezultat `y` je število pojavitev elementa v slovarju.

Še vedno se odpovemo (za sedaj):

izločanje: `Delete(`S`, x) --> y` – iz slovarja S izločimo element `x`. Rezultat `y` je lahko Boolova vrednost `true` ali `false`, ki sporoči ali je bil element uspešno izločen ali ne, ali pa operacija ničesar ne vrne.

Literatura

Primer:

Andrei Broder, Michael Mitzenmacher: *Network Applications of Bloom Filters: A Survey*, Internet Mathematics, Vol. 1, No. 4, 485-509.

Primer

Imamo slovar n-teric v DNK in nas zanima, koliko je katerih n-teric v določenem DNK.

Imejmo naslednji DNK:

`TAACCCT ...`

Potem imamo naslednje število pojavitev 3-teric v njej:

`AAC` : 1

`ACC` : 1

`CCC` : 2

`CCT` : 1

`TAA` : 1

Prostorska in časovna zahtevnost čim manjša.

Nenatančni slovar – naivna izvedba

Uporabimo dosedanje znanje.

	prostor	`Find`	`Insert`	`Delete`
seznam	$n + rn$	$O(n)$	$O(1)$	$O(n)$
urejen seznam	$n + rn$	$O(n)$	$O(n)$	$O(n)$
binarno drevo	$n + 2rn$	$O(n)$	$O(n)$	$O(n)$
AVL drevo	$n + 2rn$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
B drevo	$n + brn$	$O(\log_b n)$	$O(\log_b n)$	$O(\log_b n)$
RB-drevo	$n + 2rn$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
preskočna vrsta	$n{+}?rn$	$O(\log n)$	$O(\log n)$	$O(\log n)$
razpršena tabela	$n{+}?rn$	$O(1)$	$O(1)$	$O(1)$

Ali lahko naredimo operacije v času $O(1)$ in prostoru $O(n) = cn$ bitov prostora za nek majhen c (na primer $c = 5$)?

Primerjava: binarno drevo potrebuje $(n + 2rn) \lg n$ (ali celo $64(n + 2rn)$) bitov prostora – delo z velikimi količinami podatkov *big data*.

Bloomov filter – izvedba 1

Podatkovna struktura:

- imamo bitno polje `BF[0..m-1]`, kjer je $m = cn$;
- imamo k različnih razpršilnih funkcij $h_1(), h_2(), ..., h_k()$, ki slikajo v domeno $0..m-1$;

in potem operaciji: Podatkovna struktura:

dodajanje: `Insert(`S`, x)`:

```
Insert(S, x):
  za vsak i= 1..k:
    naračunaj li= hi(x);
    BF[li] = 1
```

iskanje: `Find(`S`, x)`:

```
Find(S, x):
  za vsak i= 1..k naračunaj li= hi(x);
  če so vsi BF[li] == 1 ==> return true
  sicer return false
```

Časovna zahtevnost: $O(k)$, prostorska zahtevnost: $O(m) = O(n)$ bitov.

Bloomov filter – izvedba 2

Podatkovna struktura:

- imamo k bitno polj `BFi[0..m-1]`, kjer je $m = \frac{cn}{k}$;
- imamo k različnih razpršilnih funkcij $h_1(), h_2(), \ldots, h_k()$, ki slikajo v domeno $0..m-1$;

in potem operaciji: Podatkovna struktura:

dodajanje: `Insert(`S`, x)`: podobno kot prej
iskanje: `Find(`S`, x)`: podobno kot prej

Časovna zahtevnost: $O(k)$, prostorska zahtevnost: $O(km) = O(n)$ bitov.

Preprostješa izvedba, ki ima enake lastnosti in zato pogosteje uporabljana. Omogoča preprosto povzporejanje.

Kako je pa z lažnimi pozitivnimi odgovori?

Verjetnost lažnega pozitivnega odgovora

Recimo, da imamo prvo izvedbo, ker jo bo lažje analizirati. Rezultati so podobni za drugo.

Ker imamo dobre razpršilne funkcije, je verjetnost, da je nek bit 0 po vstavljanju n elementov

$$p' = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{\frac{-kn}{m}} = p$$

potem je *pričakovana* vrednost, da dobimo lažen pozitiven odgovor, kar pomeni, da dobimo k enic, približno enaka

$$f \approx (1-p)^k = \left(1 - e^{\frac{-kn}{m}}\right)^k \quad .$$

Izziv: Analizirajte zgornje vrednosti za različne m, c, n in k.

Razmisleki

Smo pri prvi izvedbi.

- če povečamo k, se bo (intuitivno) zmanjšala verjetnost, da bomo našli bit 0 pri razprševanju;
- če zmanjšamo k, se bo zmanjšalo število bitov 0, kar pomeni, da bo f manjši.

Optimizacija? Matematika na pomoč!!

Naj bo $g = \ln f = k \ln(1 - p)$ potem je optimum tedaj, ko je parcialni odvod $\frac{dg}{dk} = 0$.

Ostalo prepuščeno za *izziv*.

Bloomov filter – s štetjem

Podatkovna struktura:

- imamo k d-bitno polj `BFi[0..m-1]`, kjer je $m = \frac{cn}{k}$;
- imamo k različnih razpršilnih funkcij $h_1(), h_2(), \ldots, h_k()$, ki slikajo v domeno $0..m-1$;

in potem operaciji: Podatkovna struktura:

dodajanje: `Insert(`S`, x)`:

```
Insert(S, x):
  za vsak i= 1..k:
    naračunaj li= hi(x);
    BF[li]++
```

iskanje: `Find(`S`, x)`: *izziv*: kaj vrniti?

Časovna zahtevnost: $O(k)$, prostorska zahtevnost: $O(dm) = O(n)$ bitov.

Izkaže se, da je $d = \log \log n$ z zelo veliko verjetnosto dovolj.

Operacije nad BF

Recimo, da imamo slovarja, ki sta predstavljena z Bloomovima filtroma S_1 in S_2 in naj bodo vse $h_{1,i}() = h_{2,i}()$.

- unija neštevnih slovarjev $S_1 \cup S_2$:

```
Union(S1, S2):
  za vsak i= 0..m-1:
    result.BF[i]= S1.BF[i] or S2.BF[i]
```

- unija števnih slovarjev $S_1 \cup S_2$: *izziv*
- krčenje velikosti neštevnega slovarja z m na $\frac{m}{2}$ bitov: naredimo unijo zgornje in spodnje polovice bitne tabele; ohranimo vse $h_i()$ a odslej ne uporabljamo najbolj pomembnega bita
- krčenje velikosti števnega slovarja z m na $\frac{m}{2}$ bitov: *izziv*

Izziv: zakaj je zgornje res?

Primeri uporabe

- slovar za angleški črkovalnik
- porazdeljene baze podatkov: izmenjava samo BF in ne zelotnih zapisov
- P2P prekrivna omrežja
- meritve tokov podatkov (paketi ali sporočila v omrežjih, borzni podatki ipd.)

Zapletenost

	prostor	`Find`	`Insert`	`Delete`
seznam	$n + rn$	$O(n)$	$O(1)$	$O(n)$
urejen seznam	$n + rn$	$O(n)$	$O(n)$	$O(n)$
binarno drevo	$n + 2rn$	$O(n)$	$O(n)$	$O(n)$
AVL drevo	$n + 2rn$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
B drevo	$n + brn$	$O(\log_b n)$	$O(\log_b n)$	$O(\log_b n)$
RB-drevo	$n + 2rn$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
preskočna vrsta	$n + ?rn$	$O(\log n)$	$O(\log n)$	$O(\log n)$
razpršena tabela	$n + ?rn$	$O(1)$	$O(1)$	$O(1)$
Bloomov filter	$\frac{cn}{\lg n}$	$O(1)$	$O(1)$	???