Napredni algoritmi i strukture podataka

Batch i mikro batch obrada podataka, Skip list, SimHash

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Batch obrada podataka

- Batch obrada podataka je metod obrade (uglavnom) veceg obima podataka (mogu se ponavljati), gde su podaci prvobitno skladišteni
- Ovaj metod omogućava korisnicima da procesuiraju podatke kada su računarski resursi dostupni sa malo ili bez interakcije korisnika
- Batch obrada uglavom traje duže zbog masivnosti podatka koji treba da se obrade, ili repeticije posla
- Primer za ovaj tip obrade bi bili, indeksiranje interneta i *search engines* (Google, Bing, DuckDuckGo, ...), obrada DNK, genoma, treniranje velikih modela sistema mašinskog učenja, obrada podataka sudara čestica, snimanje galaksija, crnih rupa itd.

Mikro Batch

- *Stream* i *Batch* procesuiranje su dva ekstrema
- U nekim situacijam se kombinuju zarad boljih rezultata (npr. Big Data - Lambda arhitekture)
- Mikro batch obrada je mogućnost kombinacije ova dva ekstrema u jedan unificiran način obrade
- Ovaj način obrade podataka je koristan za prikupljanja podataka u malim grupama (batch) u svrhu obrade tih podataka kako dolaze (stream)
- Često se koristi zarad uštede resursa
- Podaci mogu biti grupisani na razne načine: vremenski (1s, 1min, itd.), memorijski (1MB, 10MB, itd).

Problem

Zaposlili ste se u uzbuljivom startapu koji pravi novi sistem za skladištenje podataka. Jedan deo ovog sistema čuva podatke u memoriji i od vas se očekuje da implementirate sistem sa nekim ograničenjima i zahtevima:

- Dodavanje i brisanje treba da bude relativno brzo, i algoritam za to treba da bude jednostavan
- Pretraga bi trebala da bude relativno brza
- Prostorno treba da budemo efikasni što više možemo
- Podaci su u memoriji ali ih može biti dosta

Predlozi :) ?

Skip list - uvod

- Skip list je probabilistička struktura podataka koja je napravljena na opštoj ideji linked lista
- Ova lista koristi verovatnoću za izgradnju novih (viših) slojeva linked lista, na originalnoj listi.
- Skip list je struktura podataka koja se može koristiti umesto balansiranih stabala

- Skip list koriste verovatnoću, a ne striktno prinudno balansiranje
- Dodavanje i brisanje su znatno jednostavniji i brži od ekvivalentnih algoritama balansiranih stabala
- Koristite nasumično bacanje novčića da bi napravili strukturu podataka
- Svaki dodatni sloj veza sadrži manje elemenata, ali nema novih elemenata

Doubly-linked list vs skip list

- Dobre osobine doubly-linked list:
  - Lako za dodavanje i brisanje za $\mathcal{O}(1)$ vreme
  - Nema potrebe za procenom ukupne potrebne memorije
- Loš osobine doubly-linked list:
  - Teško je pretragu izvesti ispod $\mathcal{O}(n)$ vremena — binary search ne radi
  - Teško je skočiti na sredinu
- Skip list rešava ove probleme
- Očekivano vreme pretrage je $\mathcal{O}(\log n)$

Skip list - ideja

- Možete zamisliti ovu strukturu kao sistem metroa
- Postoje vozovi koji staje na svakoj stanici
- Ali, postoji i ekspresni voz koji staje na manje stanica
- Ovo čini ekspresni voz atraktivnom opcijom ako znate gde staje

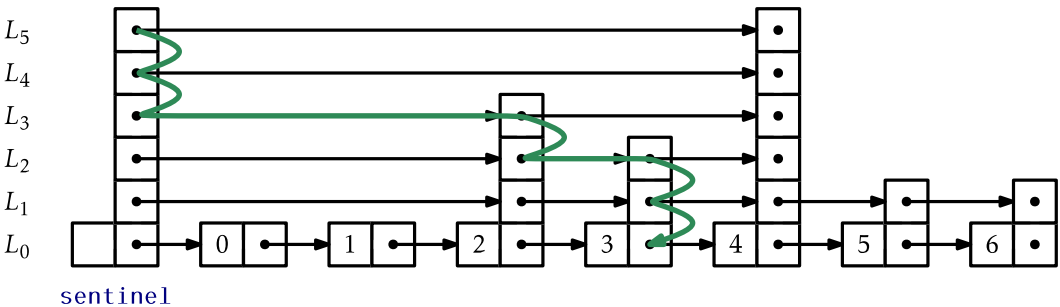
- Početak (glava) i kraj (rep) imaju pokazivaḳa svakom nivou
- Zove se Skip lista, jer omogućava preskakanje čvorova
- Čvorovi su promenljive visine od 1 do $\mathcal{O}(\log n)$ pokazivača

Skip list - pretraga

Pretraga elementa **k** se vrši po sledećem algoritmu

- Ako je k = key, kraj
- Ako je k < next key, prelazimo na nivo ispod
- Ako je k >= next key, idemo desno

Pretraga element **4**

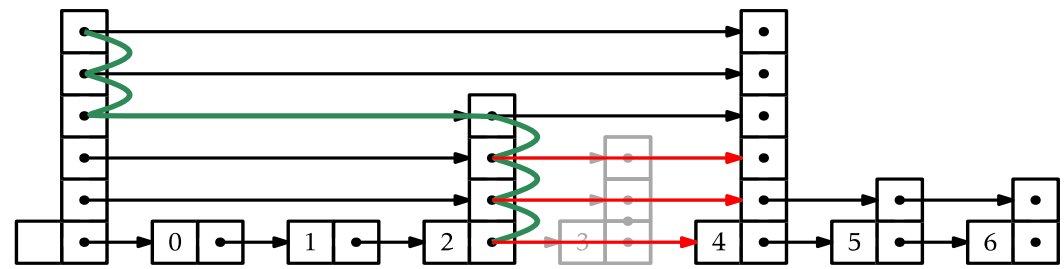


Skip list - brisanje

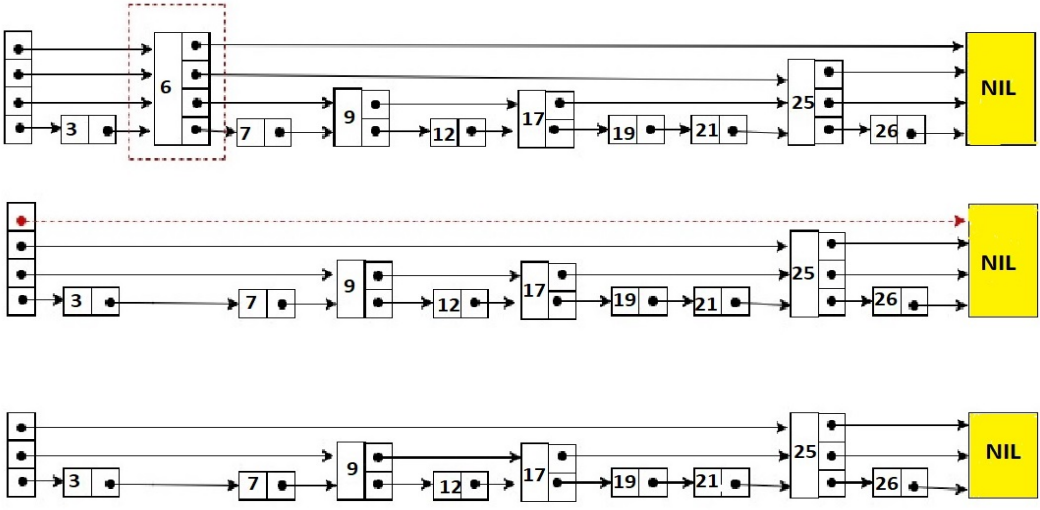
Brisanje elementa **k** se vrši po sledećim koracima:

- Lociramo koji element trebalo da se obriše, na osnovu prethodnog algoritma — **pretraga**
- Kada je element lociran, prevezujemo pokazivače da bi se element uklonio iz liste, baš kao što radimo u linked listi.
- Brisanje počinjemo od najnižeg nivoa i vršimo prevezivanje pokazivača sve dok ne stignemo do elementa
- Nakon brisanja elementa može postojati nivo bez elemenata, tako da ćemo i ove nivoe ukloniti, smanjivši nivo Skip liste.

Brisanje elementa **3**



Brisanje elementa **6**, i smanjenje nivao liste



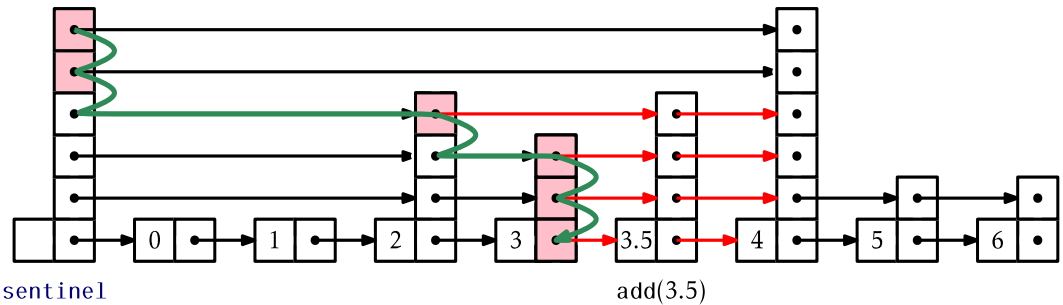
Skip list - dodavanje

- Lociramo gde bi element trebalo da se doda, na osnovu prethodnog algoritma — **pretraga**
- Povežemo pokavivač prethodnog elementa na novokreiranim elementom
- Pokazivač novokreiranog elementa pokazuje na naredni element
- Ove operacije su identične kao i kod linked liste
- **ALI**, treba i da odredimo koliko nivao naš element ima

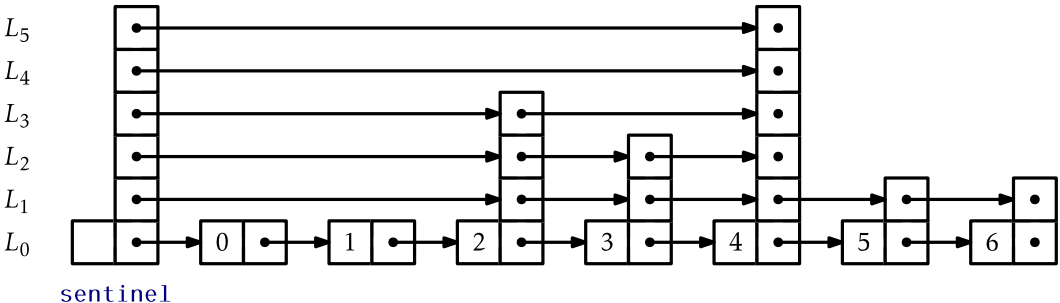
- Za proračun nivao svakog elementa, koristimo verovatnoću
- Koristimo ideju bacanje novčića (Koje su šanse za bacanje jedne glave? 50 %. Dva zaredom? 25 %. Tri zaredom? 12,5 %.)
- Inicijalna visina svakog elementa je **0**
- Bacamo novčić, i sve dok dobijamo **1** (glava) povećavamo visinu elementa

- Pronalazimo **k**
- Dodajemo nod u nivo **0**
- **while** $\mathrm{FLIP}() == \mathit{'GLAVA'}$
  - Dodajemo novi nivo
  - Povećavamo nivo elementa

Dodavanje element **3.5**



Skip list - Pitanja?



Pitanja :) ?

Skip list - Dodatni materijali

- Skip list paper
- MIT Skip list
- Open data tructures Skip list
- Building a Skip list

Problem 1

Zaposlili ste se u Google-u (bravo), i dobili ste zadatak da poboljšate njihov *crawler* mehanizam. Vaš zadatak je da utvrdite koje stranice su slične. Pred vama su sledeći zahtevi i ograničenja:

- Relativno laka paralelizacija posla
- Zauzimamo što manje resursa moguće

Predlozi :) ?

Problem 2

Zaposlili ste se u istraživačkom centru. Vaš prvi posao kao inženjera jeste da nad dva skupa gena, ustanovite sličnosti, tako da treba da uporedimo dve sekvence gena. Pred vama su sldeći zahtevi i ograničenja:

- Relativno laka paralelizacija posla
- Zauzimamo što manje resursa moguće
- Jasan i jednostavan algoritam za razumevanje

Predlozi :) ?

SimHash

- SimHash je tehnika za brzu procenu koliko su dva skupa slična
- Snaga SimHash-a je u konverziji podataka u *hash* vrednost, i izračunavanje *Hemingovog rastojanja*
- Hemingovo rastojanje je metrika koja se koristi za pronalaženje sličnosti dva skupa podataka
- Hemingovo rastojanje dobijamo koristeći **XOR** $(a \oplus b)$, a zatim računamo ukupan broj 1 u rezultujućem nizu

- Fingerprint seta podataka je *hash* vrednost njegovih karakteristika
- Slični setovi podataka imaju slične *hash* vrednosti
- Što su setovi sličniji, to je Hemingova udaljenost manja
- Izuzetno brz i efikasan algoritam u pogeldu skladišnog prostora

SimHash - algoritam

- Set podataka (npr. tekst) podelimo na delove i uklonite zaustavne reči (ako ih ima)
- Dodelimo težine dobijenim vrednostima (npr. broj ponavljanja reči)
- Izračunamo **b**-bitni *hash* za svaki element iz dobijenog skupa, propuštajući element kroz *hash* funkciju
- Za svaku dobijenu vrednost uradimo konverziju $\mathbf{0} \rightarrow \mathbf{-1}$
- Formiramo tabelu, tako što vrednosti stavimo jedne ispod drugih
- Sumiramo kolone, množeći težine sa vrednošću
- Ponovo izvršimo konverziju, ali sada za svaku vrednost u dobijenom rezultatu:
  - if $el > 0$, $el \leftarrow \mathbf{1}$
  - if $el < 0$, $el \leftarrow \mathbf{0}$
- Dobijamo **b**-bit fingerprint za ceo ulazni set
- Uradimo XOR operaciju sa drugim setom podataka i dobijamo Hemingovu udaljenost

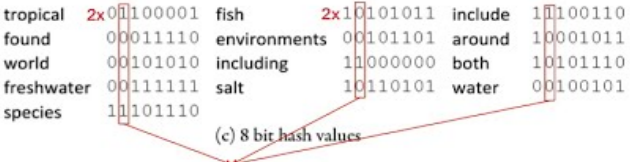
SimHash - primer

Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

(a) Original text

tropical 2 fish 2 include 1 found 1 environments 1 around 1 world 1
including 1 both 1 freshwater 1 salt 1 water 1 species 1

(b) Words with weights



(c) 8 bit hash values

1 -5 9 -9 3 1 3 3

(d) Vector V formed by summing weights

1 0 1 0 1 1 1 1

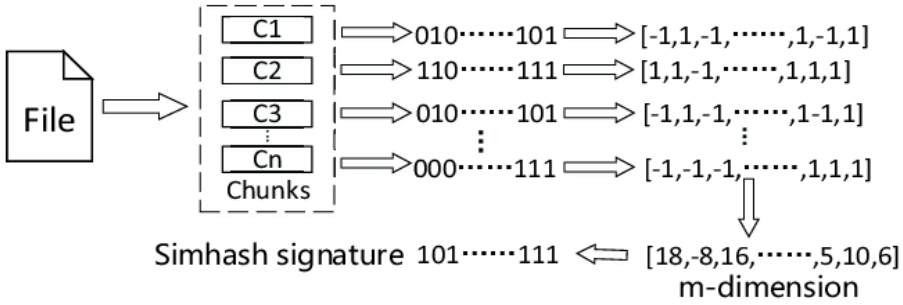
(e) 8-bit fingerprint formed from V

(Victor Lavrenko, SimHash)

SimHash - dodatni materijali

- Similarity Estimation Techniques from Rounding Algorithms
- Detecting Near-Duplicates for Web Crawling
- Computing Text Similarity by Simhash+Hamming Distance

SimHash - Pitanja



(EPAS: A Sampling Based Similarity Identification Algorithm for the Cloud)

Pitanja :) ?