Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Analiza velikih skupova podataka

Autori: Goran Delač, Marin Šilić
Ak. god. 2019/2020

2. Laboratorijska vježba

U drugoj laboratorijskoj vježbi zadatak je programski ostvariti algoritam za pronalaženje čestih skupova predmeta PCY (Park Chen Yu). U sklopu ovog problema, razmatramo skup podataka koji se sastoji od odjeljaka (košara). Svaka košara sastoji se od predmeta koji se mogu pojaviti u više košara, ali se u pojedinoj košari pojavljuju najviše jednom. Cilj algoritma je pronaći podskupove predmeta koji se pojavljuju u najvećem broju košara.

Ovaj specifičan problem u analizi velikih skupova podataka ima inherentan problem s potrošnjom memorije. Nepraktično je brojati sve skupove podskupa predmeta u memoriji jer broj pojedinačnih predmeta može biti značajan. Kako bi se omogućilo učinkovito prebrojavanje, algoritmi nastoje sažeti prostor stanja. Tradicionalan pristup primjenom algoritma A-priori to radi pretraživanjem u širinu uz korištenje svojstva monotonosti čestih predmeta. Preciznije, ako predmet nije čest, neće biti čest ni par predmeta u kojemu se navedeni predmet nalazi.

1. Algoritam PCY (Park-Chen-Yu)

Algoritam PCY je nadogradnja algoritma A-priori koja koristi još jedan dodatan prolaz kroz skup podataka kako bi se dodatno smanjio prostor riješena prilikom brojanja skupova predmeta. Konkretno, algoritam sažima skupove predmeta u pretince. Na kraju se broje samo oni skupovi koji su se našli u “čestom” pretincu, dok se ostali skupovi zanemaruju jer sigurno nisu česti. Na taj je način moguće ostvariti dodatnu uštedu pri korištenju memorije u odnosu na osnovni A-priori algoritam.

Detalji vezani za ostvarenje algoritma PCY opisani su u predavanju Finding Frequent Itemsets, a dodatni detalji mogu se naći u literaturi citiranoj na kraju predavanja.

Agoritam PCY posebno je pogodan za korak postupka u kojemu se broje parovi predmeta s obzirom na to da će čestih parova biti više nego čestih trojki. Za prebrojavanje trojki i ostalih nadskupova čestih predmeta, u većini slučajeva dovoljno je koristiti algoritam A-priori, koji je u određenim slučajima i memorijski učinkovitiji od algoritma PCY (pogledati predavanje za obrazloženje).

Stoga, u ovoj vježbi ograničava se na brojanje **parova** predmeta. U nastavku je prikazana skica mogućeg rješenja.

```
PCY(ulaz):

  //učitaj broj košara
  var brKošara = učitaj iz ulaza;

  //učitaj podatke o pragu iz ulaznih podataka
  var s = učitaj iz ulaza;  //s ima vrijednos 0-1
  var prag = s * brKošara; //cjelobrojna vrijednost (floor)

  //učitaj broj pretinaca
  var brPretinaca = učitaj iz ulaza;

  //brojač predmeta
  var brPredmeta = [];

  //prvi prolaz
  za svaku košaru {
    za svaki predmet i unutar košare {
      brPredmeta[i]++;
    }
  }

  //pretinci za funkciju sažimanja - polje veličine brPretinaca
  var pretinci = []();

  //drugi prolaz - sažimanje
  za svaku košaru {
    za svaki par predmeta i,j unutar košare {
      //sažmi par predmeta u pretinac
      //oba predmeta moraju biti česta
      ako (brPredmeta[i] >= prag) && (brPredmeta[j] >= prag) {
        var k = ((i * brPredmeta.velicina) + j) % brPretinaca;
        pretinci[k]++;
      }
    }
  }

  //treći prolaz - brojanje parova
  //mapa - ključ par predmeta [i,j], vrijednost broj ponavljanja
```

```
var parovi = [([i,j], zbroj)]
za svaku košaru {
  za svaki par predmeta i,j unutar košare {
    //oba predmeta moraju biti česta i u  čestom pretincu
    ako (brPredmeta[i] >= prag) && (brPredmeta[j] >= prag) {
      var k = ((i * brPredmeta.velicina) + j) % brPretinaca;
      ako (pretinci[k] >= prag) {
        parovi([i,j], zbroj++);
      }
    }
  }
}
```

Za one koji žele znati više

Programsko rješenje u ovoj vježbi zaustavlja se na brojanju parova. U općenitom slučaju prilikom stvaranja pravila asocijacije, potrebno je pronaći sve česte podskupove predmeta.
Rješenje se može poopćiti na način da algoritam napravi više prolaza i pri tome radi provjere za nadskupove predmeta. Nakon parova (i, j) slijedile bi trojke (i, j, k) i postupak bi se ponavljao sve dok bi se pronalazili česti podskupovi predmeta.
Postoji više načina na koji se mogu generirati kandati čestih nadskupova predmeta, a uobičajeni postupak možete naći ovdje.

2. Format ulazne datoteke i izlaza

Format ulazne datoteke u program koji predajete u ovoj laboratorijskoj vježbi je:

```
N
s
b
košara 1
košara 2
…
košara N

N je ukupan broj košara u datoteci
```

`s` je prag podrške zapisan kao omjer (vrijednost mi je 0 - 1)

`b` je broj pretinaca na raspolaganju funkciji sažimanja. Ako skup podataka sadrži **k** različitih predmeta, par predmeta **i**, **j** sažima se u pretinac ovom formulom:

$$h = ((i \cdot k) + j) \% b$$

U nastavku datoteke nalaze se košare. Svaka košara se nalazi u svojoj liniji datoteke. Predmeti unutar košare odvojeni su razmakom. Svaki predmet zapisan je kao cijeli broj [1, k], gdje je k broj različitih predmeta.

Primjer nekoliko košara u ulaznoj datoteci:

```
1 4 7 10 24
2 4 6 13
8 11 25
```

Na izlazu program mora redom ispisati sljedeće podatke:

```
A
P
X1
X2
X3
…
Xn
```

`A` je ukupan broj kandidata čestih parova koje bi brojao algoritam A-priori. Ako je algoritam u prvom prolazu odredio da je **m** predmeta često, onda taj broj iznosi:

$$\frac{m \cdot (m-1)}{2}$$

`P` je ukupan broj parova koje prebrojava algoritam PCY. Radi se samo o parovima koji se sažimaju u česti pretinac.

`X1, X2, …, Xn` su **silazno sortirani** brojevi ponavljanja čestih parova. Navodi se samo ukupan broj ponavljanja za svaki par.

Primjetite da je $A \leq P \leq n$

Primjer izlazne datoteke:

```
4576
323
1320
1034
```

```
1034
978
914
914
836
```

Napomene

1. Vremensko ograničenje za izvođenje programa za sve testne ulaze iznosit će 200s.

2. Ulazna točka za Java rješenja treba biti u **razredu PCY**, a ulazna točka u Python rješenja treba biti u datoteci **PCY.py**

3. Ulazni skup podataka potrebno je prilikom učitavanja sačuvati u memoriji - program će ga na ulaz dobiti samo jednom. Bit će osigurano dovoljno memorije za izvršavanje algoritma i pohranu ulaznog skupa podataka.

4. Nije potrebno oslobađati memoriju, odnosno dereferencirati strukture podataka i pozivati garbage collector u višim programskim jezicima (Java, Python). Bit će osigurano dovoljno memorije za sve korake algoritma.

5. Na stranicama predmeta postavljen je primjer ulazne datoteke s pripadajućim očekivanim izlazom (*lab2_primjer.zip*). Preporučamo provjeru ispravnosti na temelju zadanog primjera prije predaje vježbe na sustav *sprut*.

6. Preporuča se ostvarenje vježbi u programskim jezicima Java i Python, asistenti su ostvarili vježbe u tim jezicima i mogu vam pružiti eventualno potrebnu pomoć. Pitanja možete poslati na službeni mail predmeta avsp@zemris.fer.hr.