ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων

(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2020-21)

ΕΡΓΑΣΙΑ 4 – Συνολοδεδομένα

Προθεσμία: 28 Ιουνίου 2021, 9μμ

Στο ecourse θα βρείτε το αρχείο συνολοδεδομένων transactions.txt, το οποίο θα χρησιμοποιήσετε. Η κάθε γραμμή του αρχείου περιέχει ένα (αταξινόμητο) σύνολο από item identifiers. Το κάθε σύνολο (δηλ. η κάθε γραμμή) θεωρείται ως μία συναλλαγή κάποιου πελάτη σε ένα κατάστημα. Σε μία συναλλαγή είναι δυνατόν το ίδιο αντικείμενο να υπάρχει πολλές φορές (bag semantics). Για παράδειγμα, παρατηρείστε ότι στην 5η γραμμή το αντικείμενο με κωδικό 4 υπάρχει τρεις φορές. Κάθε συναλλαγή χαρακτηρίζεται από την αντίστοιχη γραμμή. Για παράδειγμα, η συναλλαγή στην πρώτη γραμμή του αρχείου έχει identifier 0, αυτή στη δεύτερη γραμμή έχει identifier 1, κλπ.

Μέρος 1: Containment queries

Γράψτε ένα πρόγραμμα, το οποίο διαβάζει τα δεδομένα από το αρχείο και κατασκευάζει δομές, οι οποίες μπορούν να χρησιμοποιηθούν για την αποτίμηση containment queries. Σε αυτά τα ερωτήματα, οι συναλλαγές θεωρούνται ότι είναι σύνολα (set semantics), η ερώτηση είναι κι αυτή ένα σύνολο και το ζητούμενο είναι να βρεθούν οι συναλλαγές που περιέχουν όλα τα αντικείμενα της ερώτησης. Για παράδειγμα, η συναλλαγή με id = 4 (δηλαδή η 5η γραμμή) είναι αποτέλεσμα στην ερώτηση q = {2,4,16} γιατί περιέχει και τα 3 αυτά αντικείμενα.

Στο πρόγραμμά σας, αρχικά διαβάστε τα περιεχόμενα του αρχείου δεδομένων και αποθηκεύστε όλες τις συναλλαγές σε ένα πίνακα (ή λίστα) *transactions* στη μνήμη, όπου η κάθε εγγραφή του πίνακα περιέχει το σύνολο των αντίστοιχων items. Κατόπιν, υλοποιήστε τις εξής μεθόδους:

α) Απλή μέθοδος αναφοράς (**naïve**). Για κάθε ερώτηση, διαβάζει τις συναλλαγές από τον πίνακα και αν κάποια συναλλαγή περιέχει όλα τα αντικείμενα της ερώτησης, τότε το id της συναλλαγής προστίθεται στα αποτελέσματα. Στο τέλος επιστρέφονται τα αποτελέσματα.

β) **Exact signature file**. Πριν τη χρήση της μεθόδου αυτής για αποτίμηση ερωτημάτων, διαβάστε τις συναλλαγές και δημιουργήστε έναν πίνακα *sigfile*, ο οποίος έχει το ίδιο μέγεθος με τον πίνακα *transactions*. Για κάθε συναλλαγή στον πίνακα *transactions*, δημιουργήστε ένα bitmap, του οποίου το least significant bit αντιστοιχεί στο αντικείμενο 0, το 2nd least significant bit αντιστοιχεί στο αντικείμενο 1, κλπ. Για κάθε αντικείμενο στη συναλλαγή, «ενεργοποιήστε» το αντίστοιχο bit (δηλαδή κάντε το 1) και αφήστε τα bits που αντιστοιχούν στα αντικείμενα που δεν υπάρχουν στη συναλλαγή να είναι 0. Για παράδειγμα, για μία συναλλαγή που

περιλαμβάνει μόνο τα αντικείμενα 0, 1 και 5, το bitmap που δημιουργείται είναι 100011, δηλαδή ο αριθμός 35. Για κάθε συναλλαγή λοιπόν στον πίνακα *transactions*, αποθηκεύστε στην αντίστοιχη θέση στον πίνακα *sigfile* το bitmap που προκύπτει από αυτήν. Κατόπιν, γράψτε μία συνάρτηση αποτίμησης ερωτημάτων containment, όπου η ερώτηση μετατρέπεται αρχικά σε bitmap με τον τρόπο που περιγράφεται παραπάνω και κατόπιν διατρέχουμε τον πίνακα *sigfile* και, για κάθε συναλλαγή όπου όλα τα 1-bits της ερώτησης είναι ενεργοποιημένα, εισάγουμε το id της συναλλαγής στο αποτέλεσμα. Η συνάρτηση θα πρέπει να χρησιμοποιεί bitwise operations για τους παραπάνω ελέγχους. **Το πρόγραμμά σας θα πρέπει να δημιουργεί ένα αρχείο *sigfile.txt*** όπου η 1η γραμμή είναι το signature της συναλλαγής με id=0, η 2η γραμμή είναι το signature της συναλλαγής με id=1, κλπ., δηλαδή:

```
2201170739200
4160
1078105154
...
```

γ) **Exact bitslice signature file**. Πριν τη χρήση της μεθόδου αυτής για αποτίμηση ερωτημάτων, διαβάστε τις συναλλαγές και δημιουργήστε μία δομή *bitslice*, όπου για κάθε αντικείμενο που εμφανίζεται σε συναλλαγές, δημιουργήστε ένα bitmap, του οποίου το least significant bit αντιστοιχεί στη συναλλαγή 0, το 2nd least significant bit αντιστοιχεί στη συναλλαγή 1, κλπ. Διαβάστε τις συναλλαγές και για κάθε αντικείμενο που βρίσκεται σε αυτές «ενεργοποιήστε» το αντίστοιχο bit (δηλαδή κάντε το 1). Για παράδειγμα, το αντικείμενο 165 εμφανίζεται στις συναλλαγές 7379 και 8930, άρα το αντίστοιχο bitmap είναι ο αριθμός $2^{7379} + 2^{8930}$. Κατόπιν, γράψτε μία συνάρτηση αποτίμησης ερωτημάτων containment, η οποία υπολογίζει το λογικό AND των bitmaps που αντιστοιχούν στα αντικείμενα που περιλαμβάνονται στην ερώτηση και μετά βρίσκει τις συναλλαγές που αντιστοιχούν στις θέσεις όπου το bitmap που προκύπτει έχει 1 και τις προσθέτει στο αποτέλεσμα. **Το πρόγραμμά σας θα πρέπει να δημιουργεί ένα αρχείο *bitslice.txt*** όπου κάθε γραμμή έχει το bitslice signature του κάθε αντικειμένου (τα αντικείμενα στη σειρά) . Παράδειγμα:

```
0: 13781899616355442865576460209192669...
1: 13775907167687575125877227393633422...
...
```

δ) **Inverted file**. Ακολουθήστε την ίδια διαδικασία με αυτήν της κατασκευής του *bitslice index*, με τη διαφορά ότι για κάθε αντικείμενο, φτιάξτε μία λίστα με τα ids των συναλλαγών που περιέχουν το αντικείμενο. Η δομή του *inverted file* θα έχει λοιπόν για κάθε αντικείμενο που εμφανίζεται στα δεδομένα μία *inverted list* με ταξινομημένα τα ids των συναλλαγών που το περιέχουν. Κατόπιν, γράψτε μία συνάρτηση αποτίμησης ερωτημάτων containment, η οποία υπολογίζει την τομή των λιστών που αντιστοιχούν στα αντικείμενα που περιλαμβάνονται στην ερώτηση (*με χρήση merge αλγορίθμου για sorted lists*) και προσθέτει τις συναλλαγές που συμπεριλαμβάνονται στην τομή στο αποτέλεσμα. **Το πρόγραμμά σας θα πρέπει να γράφει ένα αρχείο *invfile.txt*** όπου κάθε γραμμή έχει την ανεστραμμένη λίστα του κάθε αντικειμένου (τα αντικείμενα στη σειρά). Παράδειγμα:

```
0: [4, 6, 7, 8, 13, 16, 18, ..., 9999]
1: [2, 5, 6, 12, 13, 18, 20, ..., 9999]
...
```

Παράμετροι προγράμματος: Το πρόγραμμά σας πρέπει να παίρνει σαν command-line παραμέτρους τα εξής:

<transactions file> : το αρχείο δεδομένων (δίνεται σαν δείγμα στο ecourse το transactions.txt)

<queries file> : ένα αρχείο ερωτήσεων στην ίδια μορφή με το αρχείο δεδομένων (δίνεται σαν δείγμα στο ecourse το queries.txt)

<qnum> : το id της ερώτησης που θέλουμε να εκτελεστεί (αν θέλουμε να εκτελέσουμε όλες τις ερωτήσεις δίνουμε -1, αλλιώς το id είναι ο αριθμός γραμμής στο queries file ξεκινώντας από το 0)

<method> : η μέθοδος που θέλουμε να τρέξουμε, όπου -1 = όλες, 0 = naïve, 1 = exact signature file, 2 = exact bitslice signature file, 3 = inverted file

Για κάθε μέθοδο που τρέχετε να μετράτε το χρόνο εκτέλεσής της και τον δείχνετε στην έξοδο. Επίσης να δείχνετε τα αποτελέσματα αν η μέθοδος τρέχει για μία συγκεκριμένη ερώτηση. Αν τρέχει για όλες τις ερωτήσεις, μην βγάζετε αποτελέσματα.

Παραδείγματα εκτέλεσης:

```
<όνομα προγράμματος> transactions.txt queries.txt 0 0
Naive Method result:
{322, 5923, 6596, 8131, 8838, 1258, 77, 7182, 2063, 2289, 9650, 2227,
5523, 2454, 4854, 9752, 7641}
Naive Method computation time = ***
```

```
<όνομα προγράμματος> transactions.txt queries.txt 0 -1
Naive Method result:
{322, 5923, 6596, 8131, 8838, 1258, 77, 7182, 2063, 2289, 9650, 2227,
5523, 2454, 4854, 9752, 7641}
Naive Method computation time = ***
Signature File result:
{322, 5923, 6596, 8131, 8838, 1258, 77, 7182, 2063, 2289, 9650, 2227,
5523, 2454, 4854, 9752, 7641}
Signature File computation time = ***
Bitsliced Signature File result:
{322, 5923, 6596, 8131, 8838, 1258, 77, 7182, 2063, 2289, 9650, 2227,
5523, 2454, 4854, 9752, 7641}
Bitsliced Signature File computation time = ***
Inverted File result:
{322, 5923, 6596, 8131, 8838, 1258, 77, 7182, 2063, 2289, 9650, 5523,
2227, 2454, 4854, 9752, 7641}
Inverted File Computation time = ***
```

```
<όνομα προγράμματος> transactions.txt queries.txt -1 -1
Naive Method computation time = ***
Signature File computation time = ***
Bitsliced Signature File computation time = ***
Inverted File Computation time = ***
```

Μέρος 2: Relevance queries

Γράψτε ένα πρόγραμμα, το οποίο διαβάζει τα δεδομένα από το αρχείο και κατασκευάζει μία δομή *inverted file* η οποία εξυπηρετεί *ερωτήσεις σχετικότητας* (relevance queries). Αυτή τη φορά, πρέπει να λάβετε υπόψη (α) το γεγονός ότι ένα αντικείμενο μπορεί να εμφανίζεται πολλές φορές σε μία συναλλαγή και (β) τη σπανιότητα εμφάνισης των αντικειμένων. Έστω ότι T είναι το σύνολο των συναλλαγών και |T| είναι ο πληθάριθμός τους. Για μία συναλλαγή $\tau \in T$ και μία ερώτηση q που αποτελείται από ένα σύνολο αντικειμένων, η σχετικότητα της τ με την q ορίζεται ως $rel(\tau,q)=\Sigma_{i \in q}(occ(i,\tau)\cdot|T|/trf(i,T))$, όπου occ(i,τ) είναι το πόσες φορές εμφανίζεται το αντικείμενο i στη συναλλαγή τ, και $trf(i,T)=|\{\tau : \tau \in T \wedge i \in \tau\}|$ είναι το πόσες συναλλαγές στο σύνολο T περιλαμβάνουν το αντικείμενο i. Στόχος είναι να υπολογιστούν οι συναλλαγές τ με rel(τ,q)>0 σε φθίνουσα σειρά με βάση το rel(τ,q).

Στο inverted file σας, σε κάθε inverted list η οποία π.χ. αντιστοιχεί στο αντικείμενο i, θα πρέπει να είναι μία λίστα με ζευγάρια (τ, occ(i,τ)), ταξινομημένη με βάση το τ, όπου τ είναι τα ids των transactions που περιέχουν το i και occ(i,τ)>0 είναι το πόσες φορές εμφανίζεται το i στο τ. Εξάλλου, θα πρέπει να φτιάξετε μία δεύτερη δομή, όπου για κάθε αντικείμενο i, κρατάτε το |T|/trf(i,T). Κατόπιν, γράψτε μία συνάρτηση αποτίμησης ερωτημάτων relevance, η οποία υπολογίζει την ένωση των λιστών που αντιστοιχούν στα αντικείμενα που περιλαμβάνονται στην ερώτηση (με χρήση merge αλγορίθμου για sorted lists) και προσθέτει τις συναλλαγές που συμπεριλαμβάνονται στην ένωση στο αποτέλεσμα μαζί με το rel(τ,q) τους. Στο τέλος, ταξινομεί τα αποτελέσματα με βάση το rel(τ,q) τους και επιστρέφει τα k με το μεγαλύτερο rel(τ,q) ή όλα αν ο αριθμός τους δεν ξεπερνά το k. **Το πρόγραμμά σας θα πρέπει να γράφει ένα αρχείο *invfileocc.txt*** όπου κάθε γραμμή έχει το |T|/trf(i,T) του κάθε αντικειμένου i και την ανεστραμμένη λίστα του, εμπλουτισμένη με το occ(i,τ) για κάθε τ που περιέχει το i (τα αντικείμενα στη σειρά). Παράδειγμα:

```
0: 2.515090543259557, [[4, 2], [6, 3], [7, 1], [8, 2], ..., [9999, 1]]
1: 2.6602819898909282, [[2, 1], [5, 1], [6, 1], [12, 3], ..., [9999, 1]]
...
```

Φτιάξτε επίσης μία απλή μέθοδο αναφοράς (**naïve**), η οποία, για κάθε ερώτηση, διαβάζει τις συναλλαγές από τον πίνακα και υπολογίζει το rel(τ,q) για κάθε συναλλαγή τ. Η μέθοδος αυτή χρησιμοποιεί τη δεύτερη δομή, η οποία για κάθε αντικείμενο i, κρατάει το |T|/trf(i,T), αλλά δεν χρησιμοποιεί το inverted file.

Παράμετροι προγράμματος: Το πρόγραμμά σας πρέπει να παίρνει σαν command-line παραμέτρους τα εξής:

<transactions file> : το αρχείο δεδομένων (δίνεται σαν δείγμα στο ecourse το transactions.txt)

<queries file> : ένα αρχείο ερωτήσεων στην ίδια μορφή με το αρχείο δεδομένων (δίνεται σαν δείγμα στο ecourse το queries.txt)

<qnum> : το id της ερώτησης που θέλουμε να εκτελεστεί (αν θέλουμε όλες τις ερωτήσεις δίνουμε -1, αλλιώς το id είναι ο αριθμός γραμμής στο queries file ξεκινώντας από το 0)

<method> : η μέθοδος που θέλουμε να τρέξουμε, όπου -1 = όλες, 0 = naïve, 1 = inverted file

<k> : αριθμός των σχετικότερων αποτελεσμάτων που θέλουμε

Για κάθε μέθοδο που τρέχετε να μετράτε το χρόνο εκτέλεσής της και τον δείχνετε στην έξοδο. Επίσης να δείχνετε τα αποτελέσματα σαν μία λίστα με εγγραφές του τύπου [rel(τ,q), id of τ], ταξινομημένες με βάση το rel(τ,q), αν η μέθοδος τρέχει για μία συγκεκριμένη ερώτηση. Αν τρέχει για όλες τις ερωτήσεις, μην βγάζετε αποτελέσματα.

Παραδείγματα εκτέλεσης:

```
<όνομα προγράμματος> transactions.txt queries.txt 0 -1 2
Naive Method result:
[[263.1578947368421, 8346], [145.0651240373354, 7641]]
Naive Method computation time = ***
Inverted File result:
[[263.1578947368421, 8346], [145.0651240373354, 7641]]
Inverted File computation time = ***

<όνομα προγράμματος> transactions.txt queries.txt -1 -1 10
Naive Method computation time = ***
Inverted File computation time = ***
```

Οδηγίες ως προς τις υποβολές:

1) Μπορείτε να χρησιμοποιήσετε δομές όπως priority queue ή heap από τις βιβλιοθήκες της γλώσσας προγραμματισμού (π.χ. το module heapq της Python) εάν αυτό απαιτείται.
2) Αν χρησιμοποιήσετε Java, το πρόγραμμά σας θα πρέπει να γίνεται compile και να τρέχει και εκτός Eclipse στους υπολογιστές του εργαστηρίου. Μην χρησιμοποιείτε packages.
3) Αν χρησιμοποιήσετε Python, μην χρησιμοποιήστε τη βιβλιοθήκη pandas και μην υποβάλετε κώδικα για interactive programming (π.χ. ipython)
4) Υποβάλετε τις εργασίες σας σε ένα zip αρχείο (όχι rar) το οποίο πρέπει να περιλαμβάνει όλους τους κώδικες καθώς και ένα PDF αρχείο τεκμηρίωσης το οποίο να περιγράφει τη μεθοδολογία σας. Μην υποβάλετε αρχεία δεδομένων.
5) Μην ξεχνάτε να βάζετε το όνομά σας (σε greeklish) και το ΑΜ σε κάθε αρχείο που υποβάλετε.
6) Ο έλεγχος των προγραμμάτων σας μπορεί να γίνει σε άλλα αρχεία εισόδου από αυτά που σας δίνονται, άρα θα πρέπει ο κώδικάς σας να μην εξαρτάται από τα συγκεκριμένα αρχεία εισόδου που σας δίνονται.
7) Κάντε turnin την εργασία στο assignment4@mye041