Διαχείριση Σύνθετων Δεδομένων

3ο Σετ ασκήσεων

Μπακάλης Δημήτριος Α.Μ. 3033

Η άσκηση αποτελείται από 3 μέρη:
Στο πρώτο μέρος μας ζητείται να υλοποιήσουμε τον Αλγόριθμο A Top-K join (HRJN) πάνω σε δύο αρχεία (males_sorted και females_sorted) τον οποίο υλοποιώ στο αρχείο main1.pie Για να το τρέξουμε, αρκεί να έχουμε τα δύο αρχεία δεδομένων στον ίδιο φάκελο με το πρόγραμμα και να δώσουμε την εντολή στο τερματικό "**python3 main1.pie K_value**" όπου το K-value είναι ένας θετικός ακέραιος αριθμός ο οποίος μας δείχνει πόσα ζευγάρια θα τυπωθούν.
Για το δεύτερο μέρος μας ζητείται να υλοποιήσουμε, στα ίδια αρχεία, μια διαφορετική έκδοση του πρώτου αλγορίθμου B Top-K join τον υλοποιώ στο αρχείο main2.pie Για να το τρέξουμε, αρκεί να έχουμε τα δύο αρχεία δεδομένων στον ίδιο φάκελο με το πρόγραμμα και να δώσουμε την εντολή στο τερματικό "**python3 main2.py K_value**". Το K_value είναι της ίδιας φύσης με το πρώτο πρόγραμμα.
Τέλος στο τρίτο μέρος μας ζητείται να δώσουμε την γραφική παράσταση των χρόνων εκτέλεσης για διαφορετικές τιμές του K_value (1,2,5,10,20,50,100), τις πλειάδες που προσπελάστηκαν στην διάρκεια εκτέλεσης του πρώτου αλγορίθμου (των males_sorted και females_sorted) καθώς και τα πλεονεκτήματα/μειονεκτήματα του πρώτου αλγορίθμου σε σύγκριση με τον δεύτερο.

Βασική λειτουργία προγραμμάτων:

1ος Αλγόριθμος:

Χρησιμοποιώ 3 συναρτήσεις (f, dummy και Top_k) καθώς και μια main. Για να τρέξει ο αλγόριθμος απλά καλώ την main(), μέσα στην οποία καλείται η Top_K. Η Top_K συνάρτηση λειτουργεί ως generator και μου επιστρέφει τα K_value ζευγάρια τα οποία και τυπώνω. Επίσης τυπώνω τον χρόνο λειτουργίας καθώς και τις πλειάδες που προσπελάστηκαν (θα τα χρειαστώ στο τρίτο μέρος της άσκησης) .

Η συνάρτηση **Top_k()** υλοποιεί ουσιαστικά τον αλγόριθμο A Top-K join (HRJN). Ξεκινάει κάνοντας μια infinite while. Αυτό χρειάζεται διότι κοιτάω τα τις πλειάδες των αρχείων εν αλλαξ. Πιο συγκεκριμένα, έχω την συνάρτηση **dummy(given_file)** η οποία θα χρησιμοποιηθεί ως generator μέσα στην Top_k για να επιστρέφει μια προς μια τις πλειάδες από τα αρχεία δεδομένων. Θέτω σε δυο global μεταβλητές το πρώτο αποτέλεσμα του dummy, δηλαδή μια φορά για το males_sorted και μια φορά για το females_sorted.

Μέσα στην Top_k() και συγκεκριμένα μέσα στην infinite while κάνω for loop στον generator για το αρχείο males_sorted και παίρνω την πρώτη πλειάδα. Αποθηκεύω τα απαραίτητα πεδία σε τοπικές μεταβλητές καθώς επίσης, φτιάχνω μια τοπική λίστα που περιέχει το id, το βάρος και την ηλικία του άντρα. Ελέγχω να δω αν η συγκεκριμένη πλειάδα είναι έγκυρη και ενημερώνω τις μεταβλητές p1_max και p1_cur κατάλληλα (σύμφωνα με τον αλγόριθμο HRJN). Επίσης αν δεν μιλάμε για την πρώτη πλειάδα του males_sorted, υπολογίζω το T (threshold) με την βοήθεια της **f(value_1,value_2)** συνάρτησης. Επιπλέον ενημερώνω το λεξικό των αντρών (L1) με την πλειάδα που μόλις προσπέλασα. Το κλειδί μιας εγγραφής στο λεξικό L1 είναι το age πεδίο του και η τιμή του είναι μια λίστα από λίστες. Αν μιλάμε για την πρώτη εγγραφή του συγκεκριμένου κλειδιού τότε απλά βάζω την τοπική λίστα που δημιούργησα πιο πάνω. Αν όμως στο συγκεκριμένο κλειδί έχω ήδη κάποια τιμή τότε, επεκτείνω την τιμή με την τοπική λίστα που δημιούργησα καθώς διάβασα την πλειάδα από το αρχείο.
Αφού τελειώσαμε με τα δεδομένα στην συνέχεια, κάνω τους κατάλληλους ελέγχους για να μπορέσω να κάνω το Join και να ενημερώσω σωστά την ουρά(Q) μου.
Συγκεκριμένα, ελέγχω την τωρινή πλειάδα του males_sorted με το λεξικό (L2) των γυναικών στην θέση L2[male_age]. Έτσι είμαι σίγουρος ότι όλες οι συγκεκριμένες πλειάδες κάνουν join με την τωρινή πλειάδα των αντρών. Πριν κάνω push στην ουρά μου (Q) κάνω έναν επιπλέον έλεγχο. Ο έλεγχος αυτός ουσιαστικά βλέπει μέσα σε μια λίστα (itterated_list) ποια ζευγάρια επιστράφηκαν και σε περίπτωση που βρω το ίδιο ζευγάρι, τότε δεν το περνάω στην ουρά μου και προχωράω στο επόμενο «έγκυρο join».
Όταν τελειώσει η διαδικασία του join αυτό που κάνω είναι, να δω την κεφαλή της ουράς και όσο το αποτέλεσμα είναι μεγαλύτερο ή του T ίσο (>=Threshold) κάνω pop το συγκεκριμένο ζευγάρι από την ουρά και το επιστρέφω στην main, μέσω yield (η Top_K λειτουργεί ως generator). Επιπλέον ενημερώνω την λίστα με τα ζευγάρια που επιστράφηκαν (itterated_list)

Ακριβώς την ίδια διαδικασία ακολουθώ και στην επόμενη for loop, *μετά την for loop του male_sorted*, που αφορά το αρχείο females_sorted. Δηλαδή ενημερώνω το L2 λεξικό με την κάθε έγκυρη πλειάδα από τις γυναίκες και ελέγχω να δω με ποιες πλειάδες από το λεξικό L1 (των αντρών) κάνει join. Έπειτα με τον ίδιο τρόπο ενημερώνω την ουρά μου, κάνω το κατάλληλο pop από την ουρά(Q), ενημερώνω και την λίστα με τα ζευγάρια που επιστράφηκαν και κάνω yield προς την main.

Τέλος τυπώνω τα αποτελέσματα του Top_K στην main, τον χρόνο εκτέλεσης του προγράμματος και τις πλειάδες που προσπελάστηκαν.

2ος Αλγόριθμος:

Για τον δεύτερο αλγόριθμο έχω 1 συνάρτηση dummy(given_file) και μια main() στην οποία υλοποιώ τον αλγόριθμο B Top-K join.

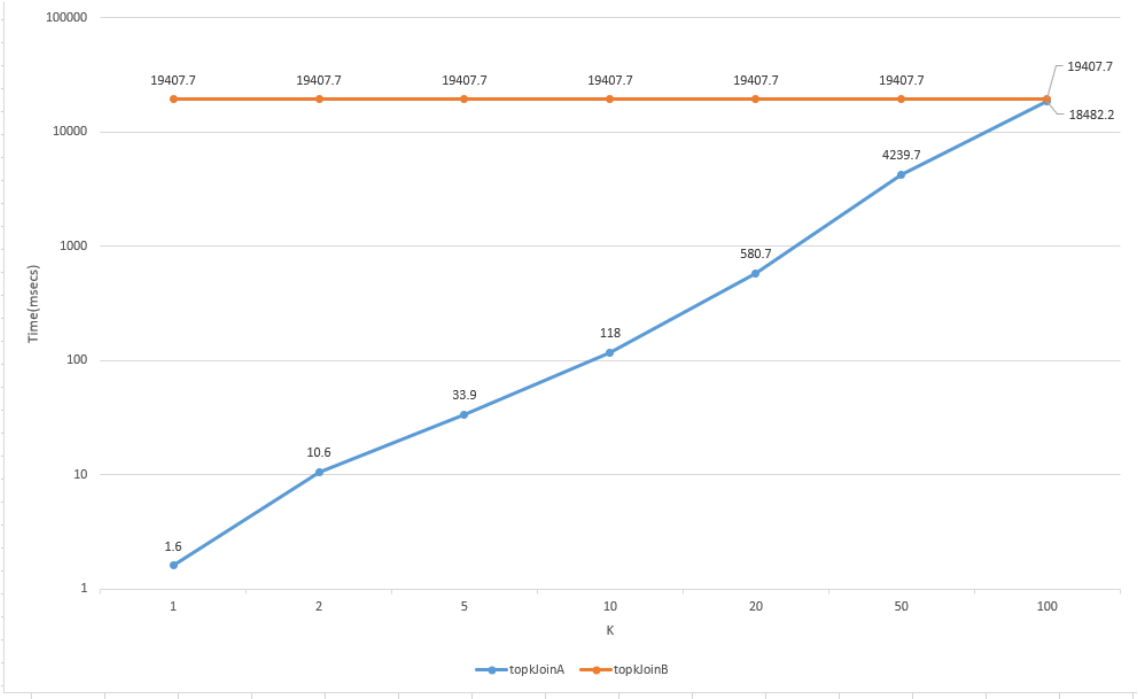
Η συνάρτηση **dummy(given_file)** χρησιμοποιείται σαν generator και απλά μου επιστρέφει την επόμενη πλειάδα από το αρχείο females_sorted.
Η πρώτη δουλειά μέσα στην main() είναι να κάνω μια for loop σε όλο το αρχείο males_sorted και να κρατήσω όλες τις «έγκυρες πλειάδες» σε ένα λεξικό L1. Δημιουργώ μια τοπική λίστα που έχει 3 εγγραφές: το id, το βάρος και την ηλικία του άντρα. Το λεξικό έχει για κλειδί το age πεδίο του και η τιμή του είναι μια λίστα από λίστες. Αν μιλάμε για την πρώτη εγγραφή του συγκεκριμένου κλειδιού τότε απλά βάζω την τοπική λίστα που δημιούργησα πιο πάνω. Αν όμως στο συγκεκριμένο κλειδί έχω ήδη κάποια τιμή τότε, επεκτείνω την τιμή με την τοπική λίστα που δημιούργησα καθώς διάβασα την πλειάδα από το αρχείο.

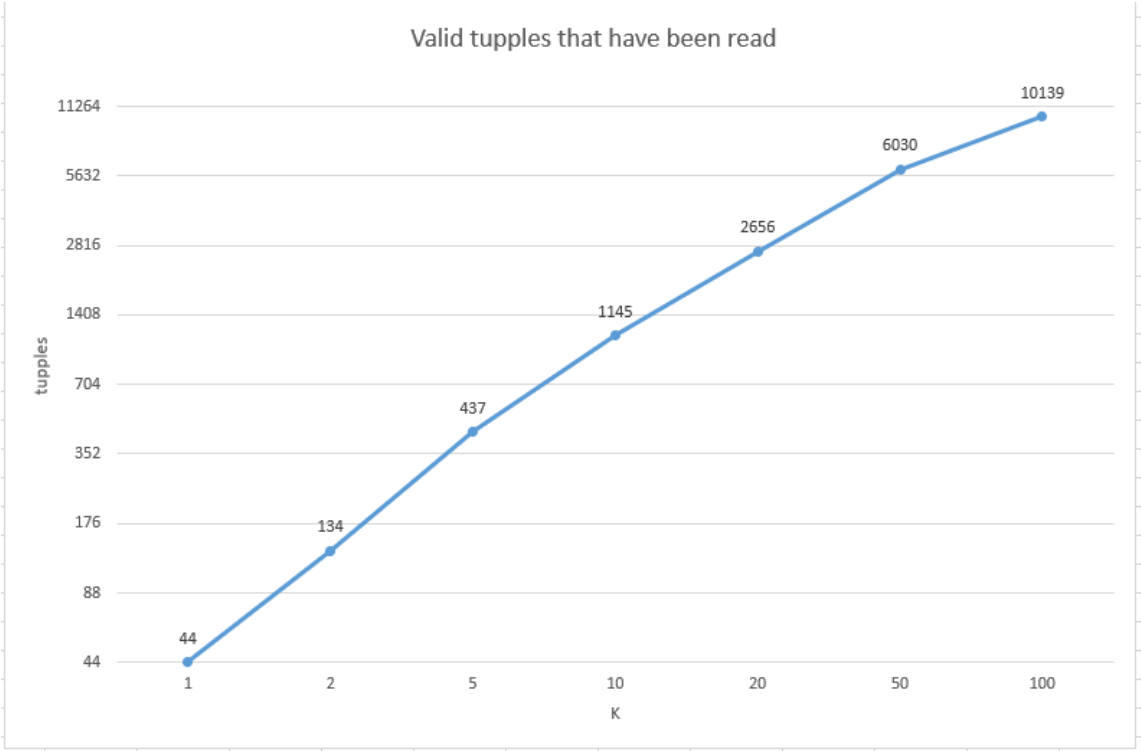
Έπειτα κάνω μια for loop στον generator(για το αρχείο females_sorted) και αποθηκεύω σε τοπικές μεταβλητές τα απαραίτητα πεδία. Αγνοώ τις μη έγκυρες πλειάδες και προχωρώ στον έλεγχο για τα join.
Για να κάνω τον έλεγχο, απλά κάνω μια for loop στο λεξικό των αντρών και συγκεκριμένα στην θέση L1[female_age]. Έτσι εξασφαλίζω ότι έχω σίγουρα join με τις συγκεκριμένες πλειάδες. Μέσα στην for μετράω πόσες πλειάδες προσπέλασα. Αν είναι ο αριθμός μικρότερος ή ίσος από το K_value απλώς κάνω push το ζευγάρι στην ουρά μου(Q), αλλιώς για κάθε επόμενη πλειάδα ελέγχω την κεφαλή της ουράς. Αν η τιμή του «πιθανού» join είναι μεγαλύτερη από την κεφαλή τότε κάνω pop την κεφαλή και push το νέο ζευγάρι, αλλιώς συνεχίζω τον έλεγχο για το επόμενο ζευγάρι.

Μόλις τελειώσω τον έλεγχο για όλα τα females, κάνω pop όλα τα στοιχεία της ουράς (Q) σε μια νέα λίστα. Ξέρω ότι η ουρά περιέχει ακριβώς K_value στοιχεία και ότι τα στοιχεία στην λίστα θα μπούνε σε αύξουσα σειρά, μιας και η ουρά μου είναι min-heap. Επομένως κάνω reverse την λίστα μου και τυπώνω: τα ζευγάρια και τους χρόνους εκτέλεσης

3ο Μέρος:

Παρακάτω παρουσιάζω τους χρόνους εκτέλεσης (A Top-K join, B Top-K join)καθώς και τις έγκυρες πλειάδες που προσπελάστηκαν κατά την διάρκεια εκτέλεσης του A Top-K join (HRJN). Να σημειωθεί ότι οι χρόνοι που εκτυπώνονται στο τερματικό είναι σε seconds, ενώ στα γραφήματα έχουν μετατραπεί σε mseconds(*10^-3). Επιπλέον να σημειωθεί ότι οι χρόνοι εκτέλεσης μπορεί να διαφέρουν κατά +/- 1.5 seconds από μηχάνημα σε μηχάνημα.





Στο δεύτερο γράφημα βλέπουμε τις πλειάδες που προσπελάστηκαν από το males_sorted και από το females_sorted. Ουσιαστικά είναι ίδια τιμή και για τα δύο. Επίσης τα τυπώνω και στο τερματικό.

Από το πρώτο γράφημα καταλαβαίνουμε ότι ο Αλγόριθμος Α είναι μέχρι ένα σημείο, και πιο συγκεκριμένα μέχρι μια τιμή του Κ, πιο βελτιωμένος σε σύγκριση με τον Αλγόριθμο Β. Αυτό γίνεται διότι για σχετικά μικρές τιμές του Κ(1,2,5,10,20,50,100) ο Α δημιουργεί δυναμικά τα topK ζευγάρια σε αντίθεση με τον Β που ο αριθμός του Κ δεν επηρεάζει τον χρόνο εκτέλεσης του. Ο χρόνος εκτέλεσης του Β επηρεάζεται αποκλειστικά από το μέγεθος των αρχείων δεδομένων.

Ο Α αλγόριθμος είναι πιο γρήγορος, μέχρι την τιμή Κ = 100, διότι διαβάζει εναλλάξ τις πλειάδες από τα αρχεία δεδομένων και έτσι «προλαβαίνει» να δημιουργήσει τα topK ζευγάρια δυναμικά. Όταν όμως προχωράμε σε μεγαλύτερες τιμές, η ουρά (max-heap) του Α γίνεται υπερβολικά μεγάλη σε μέγεθος και απαιτούνται πολλοί έλεγχοι και υπολογισμοί.

Ο Β αλγόριθμος έχει σταθερό χρόνο εκτέλεσης, για οποιαδήποτε τιμή του Κ, και αυτό γίνεται γιατί διαβάζει εξ ολοκλήρου το αρχείο των αντρών και οι συγκρίσεις γίνονται με τις πλειάδες των γυναικών. Επίσης ο Β αλγόριθμος χρησιμοποιεί μια ουρά (min-heap) όπου το μέγεθος της δεν ξεπερνάει την τιμή του K-value καθώς ελέγχουμε πάντα το στοιχείο της κεφαλής και ανάλογα την περίπτωση το αλλάζουμε.

Συμπέρασμα:

Το πλεονέκτημα του Α έναντι του Β είναι ότι, για μικρές τιμές του Κ ο Α αλγόριθμος είναι γρηγορότερος.

Το μειονέκτημα του Α έναντι του Β είναι ότι η εκτέλεση του Α, για μεγάλες τιμές, αργεί σημαντικά σε σύγκριση με τον Β που τρέχει σε σταθερό χρόνο.