Μάθημα: Διαχείριση μη Παραδοσιακών Δεδομένων

Σετ ασκήσεων: 1ο

Ονοματεπώνυμο: Κωνσταντίνος Βαρδάκας

ΑΜ: 522

Email: pcs0522@uoi.gr

Μέρος 1 (Merge-Join)

Αρχικά, στο αρχείο *merge_join.py*, το διάβασμα των αρχείων στα οποία θα πραγματοποιηθεί η συνένωση (join), γίνεται με τη χρήση μιας συνάρτησης generator (χρήση εντολής yield), *read_tsv_line* (στο αρχείο *utils.py*), η οποία επιστρέφει μια σειρά τη φορά, με αποτέλεσμα να μη απαιτείται η αποθήκευση ολόκληρων αρχείων στη μνήμη. Η συνάρτηση αυτή θεωρεί ότι η μορφή των αρχείων είναι TSV (Tab-Separated Values), με μορφή των σειρών "A\tB", όπου type(A) = string και type(B) = int.

Στο κύριο σώμα της συνάρτησης *merge_join*, αρχικοποιούνται δύο generators *R* και *S* (ένας για κάθε αρχείο που πρόκειται να συνενωθούν) και δημιουργείται ένα άδειο αρχείο TSV (*RjoinS.tsv*) για την αποθήκευση των αποτελεσμάτων του join. Πριν ξεκινήσει η διαδικασία της συνένωσης, οι μεταβλητές *r* και *s* αρχικοποιούνται με την πρώτη σειρά των αρχείων *R_sorted.tsv* και *S_sorted.tsv* αντίστοιχα (αν κάποια από αυτές δεν υπάρχει, η τιμή τους είναι *None*). Επιπλέον, η μεταβλητή *max_buffer_size* αρχικοποιείται με τιμή 0, και θα χρησιμοποιηθεί για να παρακολουθεί το μέγιστο μέγεθος του buffer, ο οποίος αποθηκεύει τις γραμμές του *S* που ταιριάζουν με την τρέχουσα γραμμή του *R*.

Η διαδικασία της συνένωσης ξεκινά και επαναλαμβάνεται όσο υπάρχουν γραμμές και στα δύο αρχεία (*r* και *s*) δηλαδή όσο οι τιμές των *r* και *s* δεν είναι *None*. Όταν τα πρώτα πεδία των γραμμών *r* και *s* είναι ίσα (*r[0] == s[0]*), δημιουργείται ο πίνακας *buffer*, στον οποίο αποθηκεύονται όλες οι γραμμές του *S* που έχουν το ίδιο πρώτο πεδίο με το *r[0]*. Η διαδικασία αυτή συνεχίζεται μέχρι το πρώτο πεδίο της γραμμής του *S* να είναι μεγαλύτερο από το *r[0]* (δηλαδή το *s[0] > r[0]*).

Μετά την αποθήκευση των γραμμών στο buffer, η μεταβλητή *max_buffer_size* ενημερώνεται με το μέγιστο μέγεθος του buffer μέσω της εντολής *max_buffer_size = max(max_buffer_size, len(buffer))*.

Στη συνέχεια, ακολουθεί η διαδικασία εγγραφής των αποτελεσμάτων στο αρχείο εξόδου, κατά την οποία οι γραμμές του *R* συνενώνονται με τις αντίστοιχες γραμμές του *S* που βρίσκονται στο *buffer*. Αυτή η διαδικασία επαναλαμβάνεται για όλες τις γραμμές του *R* που έχουν το ίδιο πρώτο πεδίο με το *key*. Όταν το πρώτο πεδίο της τρέχουσας γραμμής του *R* είναι μεγαλύτερο από το *key*, η διαδικασία του join ολοκληρώνεται για τη συγκεκριμένη ομάδα και προχωράμε στη νέα γραμμή του *R*.

Ανάλογα με το πώς συγκρίνονται τα πρώτα πεδία των γραμμών *r* και *s*, η διαδικασία συνεχίζεται ως εξής:

- Αν το πρώτο πεδίο του *r* είναι μικρότερο από το *s*, προχωράμε στην επόμενη γραμμή του *R*.
- Αν το πρώτο πεδίο του *s* είναι μικρότερο από το *r*, προχωράμε στην επόμενη γραμμή του *S*.
- Αν τα πρώτα πεδία είναι ίσα, συνεχίζουμε με τη διαδικασία του join.

Αυτή η διαδικασία ολοκληρώνεται όταν κάποιος από τους generators φτάσει στο τέλος του αρχείου (δηλαδή όταν *r = None* ή *s = None*) καθώς η συνένωση απαιτεί να υπάρχουν αντίστοιχες γραμμές με το ίδιο πρώτο πεδίο και στα δύο αρχεία.

Μέρος 2 (Union)

Η υλοποίηση της ένωσης δύο αρχείων (union) έχει υλοποιηθεί στο αρχείο *union_sorted.py* πραγματοποιείται με τον ίδιο τρόπο διαβάσματος αρχείων, χρησιμοποιώντας τη συνάρτηση generator *read_tsv_line*. Η αρχικοποίηση των generators για τα αρχεία *R* και *S*, καθώς και η ανάγνωση της πρώτης γραμμής για κάθε αρχείο (*r* και *s*), ακολουθεί την ίδια διαδικασία με την προηγούμενη υλοποίηση. Όμοια, δημιουργείται ένα άδειο αρχείο εξόδου (*RunionS.tsv*), το οποίο θα αποθηκεύσει τα αποτελέσματα της ένωσης.

Για την αποφυγή διπλότυπων εγγραφών στο αρχείο εξόδου, εισάγεται η μεταβλητή *last_written*, η οποία αρχικοποιείται ως *None*. Αυτή η μεταβλητή χρησιμοποιείται για να παρακολουθεί αν η τρέχουσα σειρά έχει ήδη καταχωρηθεί στο αρχείο εξόδου, αποφεύγοντας έτσι την εγγραφή των ίδιων πλειάδων πολλές φορές.

Η διαδικασία της ένωσης ξεκινά και επαναλαμβάνεται όσο υπάρχουν σειρές είτε στο *r* είτε στο *s*, δηλαδή όσο τουλάχιστον ένα από τα αρχεία περιέχει ακόμα δεδομένα. Αν η τρέχουσα σειρά από το αρχείο *R* (*r*) είναι μικρότερη από την τρέχουσα σειρά από το αρχείο *S* (*s*), η σειρά *r* καταχωρείται στο αρχείο εξόδου, εφόσον δεν έχει καταχωρηθεί ήδη (δηλαδή αν δεν είναι ίση με την τιμή της μεταβλητής *last_written*). Στη συνέχεια, προχωράμε στην επόμενη σειρά του αρχείου *R*. Αν, αντιθέτως, η τρέχουσα σειρά του *S* (*s*) είναι μικρότερη από την σειρά του *R* (*r*), η σειρά *s* καταχωρείται στο αρχείο, εφόσον δεν έχει καταχωρηθεί ξανά και θέτουμε ως *s* την επόμενη γραμμή του *S*.

Όταν οι σειρές *r* και *s* είναι ίδιες (δηλαδή *r[0] == s[0]*), καταχωρείται η σειρά μόνο αν δεν έχει καταχωρηθεί προηγουμένως, και στη συνέχεια προχωράμε στην επόμενη σειρά και από τα δύο αρχεία. Η διαδικασία συνεχίζεται μέχρι να εξαντληθούν οι σειρές και από τα δύο αρχεία.

Αυτή η μέθοδος εξασφαλίζει την απουσία διπλότυπων αποτελεσμάτων, χωρίς να απαιτεί την προσωρινή αποθήκευση δεδομένων σε buffer.

Μέρος 3 (Intersection)

Για την υλοποίηση της τομής δύο αρχείων (intersection) που πραγματοποιείται στο αρχείο *intersection_sorted.py*, χρησιμοποιείται και πάλι η συνάρτηση generator *read_tsv_line* για το διάβασμα των αρχείων, επιτρέποντας την ανάγνωση μίας σειράς τη φορά, χωρίς να απαιτείται η φόρτωση ολόκληρων των αρχείων στη μνήμη. Όμοια, πραγματοποιούνται οι αρχικοποιήσεις και η δημιουργία του αρχείου όπως και στις προηγούμενες περιπτώσεις.

Η διαδικασία της τομής ξεκινά και επαναλαμβάνεται όσο υπάρχουν σειρές και στα δύο αρχεία (δηλαδή όσο και το *r* και το *s* δεν είναι *None*). Συγκεκριμένα:

- Όταν οι σειρές *r* και *s* είναι ίδιες (*r* == *s*), τότε αυτή η κοινή σειρά αποτελεί μέρος της τομής. Για να αποφύγουμε την εγγραφή διπλότυπων, συγκρίνουμε τη σειρά με τη μεταβλητή *last_written* και την καταχωρούμε μόνο αν δεν έχει γραφτεί ήδη. Μετά την εγγραφή, προχωρούμε στην επόμενη σειρά και από τα δύο αρχεία.
- Αν η σειρά r είναι μικρότερη από τη σειρά s (r < s), προχωράμε στην επόμενη σειρά του R.
- Αντιστρόφως, αν η σειρά s είναι μικρότερη από τη σειρά r (s < r), προχωράμε στην επόμενη σειρά του S.

Μέρος 4 (Set-Difference)

Για την υλοποίηση της διαφοράς δύο ταξινομημένων αρχείων (*set difference,* του αρχείου *difference_sorted.py*), χρησιμοποιήθηκε η ίδια προσέγγιση με τη συνάρτηση read_tsv_line και με τις αντίστοιχες αρχικοποιήσεις. Στη συνέχεια, ο αλγόριθμος εκτελείται όσο υπάρχει επόμενη σειρά στο αρχείο *R*, με τις πιθανές περιπτώσεις να είναι τρεις:

- Αν το στοιχείο του *R* είναι μεγαλύτερο από το στοιχείο του *S* (*r* > *s*), αυξάνεται ο δείκτης του *S* (δεν μπορούμε να γνωρίσουμε σε αυτό το σημείο αν το συγκεκριμένο r ανήκει στην διαφορά R - S ή στην τομή του R και του S, γνωρίζουμε μόνο ότι το s ανήκει στην διαφορά του S - R το οποίο δεν είναι το ζητούμενο της συνάρτησης)
- Αν το στοιχείο του *R* είναι ίδιο με το στοιχείο του *S* (*r* == *s*), τότε αυτό ανήκει στην τομή τους, συνεπώς δεν καταγράφεται και προχωρούν και οι δύο δείκτες.
- Αν το στοιχείο του *R* είναι μικρότερο από το στοιχείο του *S* (*s* > *r*), τότε το στοιχείο αυτό ανήκει στη διαφορά τους και καταγράφεται, εφόσον δεν έχει ήδη καταγραφεί. Στη συνέχεια, προχωράμε στον επόμενο δείκτη του *R*.

Μέρος 5 (Grouping and aggregation)

Για την υλοποίηση της ομαδοποίησης και συνάθροισης των πλειάδων του R σύμφωνα με το πρώτο πεδίο (αρχείο *group_merge_sort.py*), χρησιμοποιήθηκε ο αλγόριθμος sort-merge, προσαρμοσμένος ώστε να επιτελεί τη συνάθροιση κατά τη διαδικασία της συγχώνευσης.

Αρχικά, το αρχείο "*R.tsv*" διαβάζεται στη μνήμη με τη συνάρτηση "*read_tsv*" (στο αρχείο *utils.py*), η οποία επιστρέφει μια λίστα από πλειάδες (key, value), όπου το πρώτο πεδίο (key) παραμένει ως συμβολοσειρά, ενώ το δεύτερο πεδίο (value) μετατρέπεται σε ακέραιο αριθμό. Όμοια με την συνάρτηση *read_tsv_line* του αρχείου *utils.py*, η συνάρτηση αυτή θεωρεί ότι η μορφή των αρχείων είναι TSV (Tab-Separated Values), με μορφή των σειρών “A\tB”, όπου type(A) = string και type(B) = int.

Στη συνέχεια, ο αλγόριθμος merge sort εφαρμόζεται μέσω της αντίστοιχης συνάρτησης, διασπώντας τη λίστα αναδρομικά μέχρι να προκύψουν υπολίστες μεμονωμένων στοιχείων. Η βασική διαφορά σε αυτή την υλοποίηση είναι η προσαρμογή της διαδικασίας συγχώνευσης, η οποία γίνεται στη συνάρτηση "merge". Κατά τη συγχώνευση πραγματοποιείται η ακόλουθη διαδικασία:

- Αν το πρώτο πεδίο της πλειάδας από τον αριστερό πίνακα είναι μικρότερο από το αντίστοιχο πεδίο της πλειάδας του δεξιού πίνακα, προστίθεται η αριστερή πλειάδα στο αποτέλεσμα.
- Αντίστοιχα, αν το πρώτο πεδίο της δεξιάς πλειάδας είναι μικρότερο, προστίθεται η δεξιά πλειάδα στο αποτέλεσμα.
- Αν τα πρώτα πεδία είναι ίσα, πραγματοποιείται άθροιση των δεύτερων πεδίων και δημιουργείται μια νέα πλειάδα με αυτό το άθροισμα.

Το αποτέλεσμα της συγχώνευσης επιστρέφεται ως λίστα πλειάδων (key, value), η οποία μπορεί να χρησιμοποιηθεί για εγγραφή στο αρχείο εξόδου.

Τέλος, η συνάρτηση "group_merge_sort" καλεί τη "merge_sort" και γράφει το αποτέλεσμα σε αρχείο "Rgroupby.tsv". Με τον τρόπο αυτό επιτυγχάνεται η επιθυμητή ομαδοποίηση και συνάθροιση των πλειάδων, ενώ η διαδικασία εκτελείται εξολοκλήρου στην κύρια μνήμη.

Οδηγίες εκτέλεσης

Για την εκτέλεση του εκάστοτε αρχείου, ακολουθείται η παρακάτω συλλογιστική:

- Ανοίγουμε το τερματικό στο φάκελο των αρχείων .py
- Προσθέτουμε τα αρχεία των δεδομένων στον ίδιο φάκελο
- Η γενική εντολή είναι: python {scriptname.py} args

Συγκεκριμένα όλα τα αρχεία πλην του group_merge_sort.py, τα arguments είναι <R_file.tsv> <S_file.tsv> [output_file.tsv] (με <> τα υποχρεωτικά και με [] τα προαιρετικά) π.χ.:
>>> python merge_join.py R_sorted.tsv S_sorted.tsv RjoinS.tsv

Αναφορικά με το αρχείο group_merge_sort.py, τα arguments είναι <R_file.tsv> [output_file.tsv]