

ΕΡΓΑΣΙΑ 1 – Υλοποίηση Τελεστών

Μέρος 1 (merge-join):

Input: 2 αρχεία tsv ίδιου format

Output: Πλειάδες συνένωσης των 2 αρχείων tsv

Το πρόγραμμα αυτό ακολουθεί τον αλγόριθμο merge-join.

Ουσιαστικά ορίζουμε “δεικτες” (R_line & S_line) σε κάθε λίστα, όπου αρχικοποιούνται στο πρώτο στοιχείο της κάθε λίστας.

*Διαβάζουμε κάθε γραμμή μια μόνο μια φορά σε όλο το πρόγραμμα

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S -> ο “δείκτης” της λίστας R μετακινείται μια θέση πιο κάτω

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας R -> ο “δείκτης” της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S -> διαβάζονται τα κοινά στοιχεία της λίστας S (αυξάνοντας τον δείκτη της λίστας S) και εισάγονται σε έναν πίνακα “buffer”. Όταν το buffer σταματήσει την εισαγωγή στοιχείων, τσεκάρουμε αν ο αριθμός στοιχείων του buffer είναι μεγαλύτερος από το length του προηγούμενου buffer. Αν ισχύει τότε θέτουμε το maxBuffer με το length του current buffer, έπειτα τυπώνεται ο συνδιασμός:

Στοιχείο δείκτη R - Στοιχεία Buffer

Και αυξάνουμε τον δείκτη του R

Και συνεχίζεται ολη αυτή η διαδικασία σε μια λούπα μέχρι να φτάσουμε στο τελευταίο line της λίστας R

Τέλος, τυπώνεται το maxBuffer size.

Μέρος 2 (union):

Input: 2 αρχεία tsv ίδιου format

Output: Πλειάδες ένωσης των 2 αρχείων tsv

Το πρόγραμμα ακολουθεί μια παραλλαγή της μεθόδου merge-join.

Όπως και στο πρώτο μέρος, χρησιμοποιούμε πάλι “δεικτες” (R_line & S_line) όπου αρχικοποιούνται στο πρώτο στοιχείο της κάθε λίστας.

*Διαβάζουμε κάθε γραμμή μια μόνο μια φορά σε ολο το πρόγραμμα

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S -> printαρουμε το στοιχείο R και ο “δείκτης” της λίστας R μετακινείται μια θέση πιο κάτω

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας R -> printαρουμε το στοιχείο S και ο “δείκτης” της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S, συγκρινουμε τους αριθμούς της πλειάδας και τους τυπώνουμε με τη σειρά από τον μικρότερο στον μεγαλύτερο

(-Αν αριθμοςR < αριθμοςS -> print R & αυξηση δεικτη R

-Αν αριθμοςS < αριθμοςR -> print S & αυξηση δεικτη S

-Αν αριθμοςR = αριθμοςS -> print R ή print S & αυξηση δεικτη R & S)

Συνέχιση της διαδικασίας σε λούπα μέχρι να φτάσουμε στο τέλος μιας λίστας. Έπειτα printαρουμε τα υπολοιπόμενα στοιχεία της άλλης λίστας

*υπάρχει ένα temp variable στο οποίο αποθηκεύεται κάθε φορά το τελευταίο στοιχείο που “δεχόμαστε”. Αν το επόμενο στοιχείο που θέλουμε να βαλουμε στο output είναι ίσο με το temp στοιχείο, τότε το απορρίπτουμε διότι είναι διπλότυπο.

Μέρος 3 (intersection):

Input: 2 αρχεία tsv ίδιου format

Output: Πλειάδες τομής των 2 αρχείων tsv

Αυτό το πρόγραμμα είναι σχεδόν ίδιο με το δεύτερο πρόγραμμα, αλλά με κάποιες αλλαγές.

*Διαβάζουμε κάθε γραμμή μια μόνο μια φορά σε όλο το πρόγραμμα

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S -> ο “δείκτης” της λίστας R μετακινείται μια θέση πιο κάτω

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας R -> ο “δείκτης” της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S, συγκρινουμε τους αριθμούς της πλειάδας.

(-Αν αριθμοςR < αριθμοςS -> αυξηση δεικτη R

-Αν αριθμοςS < αριθμοςR -> αυξηση δεικτη S

-Αν αριθμοςR = αριθμοςS -> print R ή print S & αυξηση δεικτη R & S)

Συνέχιση της διαδικασίας σε λούπα μέχρι να φτάσουμε στο τέλος μιας από των 2 λιστών.

*υπάρχει ένα temp variable στο οποίο αποθηκεύεται κάθε φορά το τελευταίο στοιχείο που “δεχόμαστε”. Αν το επόμενο στοιχείο που θέλουμε να βαλουμε στο output είναι ίσο με το temp στοιχείο, τότε το απορρίπτουμε διότι είναι διπλότυπο.

Μέρος 4 (set-difference):

Input: 2 αρχεία tsv ίδιου format

Output: Πλειάδες διαφοράς R-S των 2 αρχείων tsv

Αυτό το πρόγραμμα είναι σχεδόν ίδιο με το τρίτο πρόγραμμα, αλλά με κάποιες αλλαγές.

*Διαβάζουμε κάθε γραμμή μια μόνο μια φορά σε όλο το πρόγραμμα

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S -> printαρουμε το στοιχείο R και ο “δείκτης” της λίστας R μετακινείται μια θέση πιο κάτω

*Όσο το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας R -> printαρουμε το στοιχείο S και ο “δείκτης” της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο δείκτης της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο δείκτης πλειάδας S, συγκρινουμε τους αριθμούς της πλειάδας.

(-Αν αριθμοςR < αριθμοςS -> print R & αυξηση δεικτη R

-Αν αριθμοςS < αριθμοςR -> αυξηση δεικτη S

-Αν αριθμοςR == αριθμοςS -> αυξηση δεικτη R & S)

Συνέχιση της διαδικασίας σε λούπα μέχρι να φτάσουμε στο τέλος μας από των 2 λιστών.

*υπάρχει ένα temp variable στο οποίο αποθηκεύεται κάθε φορά το τελευταίο στοιχείο που “δεχόμαστε”. Αν το επόμενο στοιχείο που θέλουμε να βάλουμε στο output είναι ίσο με το temp στοιχείο, τότε το απορρίπτουμε διότι είναι διπλότυπο.

Μέρος 5 (Ομαδοποίηση και συνάθροιση):

Input: 1 αρχείο tsv

Output: Sorted R με summed πλειάδες ίδιου πρώτου πεδίου

Φορτώνουμε το αρχείο R σε μια λίστα,

το πρόγραμμα υλοποιεί τον recursive αλγόριθμο merge-sort, με κάποιες αλλαγές.

Εκεί που συγκρίνει τα στοιχεία των πινάκων τσεκάρει αν 1ο πεδίο πλειάδας στον πίνακα left == 1ο πεδίο πλειάδας στον πίνακα right. Αν ισχύει αυτό τότε αντί να γράψει στην λίστα μας την πλειάδα R[j], γράφει την πλειάδα (R[j][0], L[i][0]+R[j][0]) ΚΑΙ σβήνει από τις λίστες τα L[i] και R[j] (ουσιαστικά τα θέτω με (“”, 0)) για να μη ξαναγίνει σύγκριση μεταξύ τους και να μη μπουν στην τελική μας λίστα

Όταν τελειώσει το mergeSort function, η λίστα θα έχει γίνει sorted ακριβώς με τον τρόπο που ζητάει η εκφώνηση. **Αλλά** υπάρχουν ακόμα οι “κενές” πλειάδες (“”, 0). Για αυτό στη main φιλτραρώ την λίστα άλλη 1 φορά έτσι ώστε να διωξω τις “κενές” πλειάδες

**Τα προγράμματα έχουν τεσταριστεί με διάφορες λίστες και φαίνεται να δουλεύουν ως πρέπει - βγάζουν σωστά outputs