Σωτήριος Παναγιώτου, 4456

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 “pointers” σε καθε λίστα, όπου αρχικοποιούνται στο πρώτο στοιχείο της κάθε λίστας.

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

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

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

*Αν το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S -> διαβάζονται τα κοινά στοιχεία της λίστας S (αυξάνοντας τον pointer της λιστας S) και εισάγονται σε έναν πίνακα “buffer”. Οταν 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 pointer R - Στοιχεία Buffer
Και αυξάνουμε τον pointer του R

Και συνεχίζεται ολη αυτή η διαδικασία σε μια λούπα μέχρι να φτάσουμε στο τελευταίο line της λίστας R
Τελος, 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 “pointers” όπου αρχικοποιούνται στο πρώτο στοιχείο της κάθε λίστας.

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

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

*Οσο το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας R -> printαρουμε το στοιχείο S και ο "pointer" της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S, συγκρινουμε τους αριθμούς της πλειάδας και τους τυπώνουμε με τη σειρά από τον μικρότερο στον μεγαλυτερο
(-Αν αριθμοςR < αριθμοςS -> print R & αυξηση pointer R
-Αν αριθμοςS < αριθμοςR -> print S & αυξηση pointer S
-Αν αριθμοςR = αριθμοςS -> print R ή print S & αυξηση pointer R & S)

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

*υπάρχει ενα temp variable στο οποιο αποθηκεύεται κάθε φορά το τελεύταιο στοιχείο που "δεχόμαστε". Αν το επόμενο στοιχείο που θέλουμε να βαλουμε στο output είναι ίσο με το temp στοιχείο, τοτε το απορρίπτουμε διότι ειναι 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

*Οσο το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S -> ο "pointer" της λίστας R μετακινείται μια θέση πιο κάτω

*Οσο το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας R -> ο "pointer" της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S, συγκρινουμε τους αριθμούς της πλειάδας.
(-Αν αριθμοςR < αριθμοςS -> αυξηση pointer R
-Αν αριθμοςS < αριθμοςR -> αυξηση pointer S
-Αν αριθμοςR = αριθμοςS -> print R ή print S & αυξηση pointer R & S)

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

*υπάρχει ενα temp variable στο οποιο αποθηκεύεται κάθε φορά το τελεύταιο στοιχείο που "δεχόμαστε". Αν το επόμενο στοιχείο που θέλουμε να βαλουμε στο output είναι ίσο με το temp στοιχείο, τοτε το απορρίπτουμε διότι ειναι 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

*Οσο το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S -> printαρουμε το στοιχείο R και ο "pointer" της λίστας R μετακινείται μια θέση πιο κάτω

*Οσο το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας S είναι μικρότερο από το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας R -> printαρουμε το στοιχείο S και ο "pointer" της λίστας S μετακινείται μια θέση πιο κάτω

*Αν το στοιχείο στο οποίο βρίσκεται ο pointer της πλειάδας R είναι ίσο με το στοιχείο στο οποίο βρίσκεται ο pointer πλείαδας S, συγκρινουμε τους αριθμούς της πλειάδας.
(-Αν αριθμοςR < αριθμοςS -> print R & αυξηση pointer R
-Αν αριθμοςS < αριθμοςR -> αυξηση pointer S
-Αν αριθμοςR == αριθμοςS -> αυξηση pointer R & S)

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

*υπάρχει ενα temp variable στο οποιο αποθηκεύεται κάθε φορά το τελεύταιο στοιχείο που "δεχόμαστε". Αν το επόμενο στοιχείο που θέλουμε να βαλουμε στο output είναι ίσο με το temp στοιχείο, τοτε το απορρίπτουμε διότι ειναι 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 λιστα,
το προγραμμα υλοποιεί τον recursive αλγοριθμο merge-sort, με καποιες αλλαγες.

Εκει που συγκρίνει τα στοιχεια των πινάκων τσεκάρει αν 1ο πεδίο πλειαδας στον πινακα left == 1ο πεδίο πλειαδας στον πινακα right. Αν ισχυει αυτο τότε αντι να γράψει στην λίστα μας την πλειάδα R[j], γράφει την πλειάδα (R[j][0], L[I][0]+R[j][0]) ΚΑΙ σβήνει απο τις λίστες τα L[i] και R[j] (ουσιαστικα τα θετω με ("", 0)) για να μη ξαναγινει συγκριση μεταξυ τους και να μη μπουν στην τελικη μας λιστα

Οταν τελειώσει το mergeSort function, η λιστα θα εχει γινει sorted ακριβως με τον τροπο που ζηταει η εκφωνηση. **Αλλά** υπάρχουν ακόμα οι "κενές" πλειάδες ("", 0). Για αυτο στη main φιλτραρω την λιστα αλλη 1 φορα ετσι ωστε να διωξω τις "κενες" πλειαδες

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