

# ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

---

## Project 2 – Multiple Sort Merge Join

**Όνομα:** Ελευθέριος Δημητράς **ΑΜ:** 1115201600042

**Όνομα:** Μιχαήλ Ξανθόπουλος **ΑΜ:** 1115201600119

**Εντολή Εκτέλεσης:** ./main -w ./workloads/small/ -q small.work < ./workloads/small/small.init

### Διευκρινήσεις Merge:

Η **merge** παίρνει σαν ορίσματα τους δύο ταξινομημένους πίνακες όπως αυτοί επεστράφησαν από την **TableSort()**. Χρησιμοποιείται ένα δείκτης στον πίνακα **A** και δύο δείκτες στον πίνακα **B**. Ο ένας εκ των δύο του **B** είναι **pinned** στο 1<sup>ο</sup> από μια λίστα με ίδια κλειδιά και ο 2<sup>ος</sup> κάνει το traversal στη λίστα αυτή. Μόλις ο δείκτης του **A** και ο 2<sup>ος</sup> δείκτης του **B** δείχνουν σε ίδιο κλειδί, το εισάγουμε στη λίστα. Αν το κλειδί του **A** είναι μεγαλύτερο από του **B**, μετακινούμε τον **pinned** δείκτη στη λίστα με τα αμέσως μεγαλύτερα κλειδιά του **B**. Τέλος, αν μετακινήσουμε το δείκτη του **A** στο επόμενο κλειδί και αυτό είναι ίδιο με το προηγούμενο, επαναφέρουμε το 2<sup>ο</sup> δείκτη του **B** στη θέση του **pinned** δείκτη, αλλιώς μετακινούμε τον **pinned** στη θέση του 2<sup>ου</sup>.

### Διευκρινήσεις Λίστας:

Η λίστα έχει υλοποιηθεί με **templates** και η λειτουργικότητά της είναι η εξής. Κάθε κόμβος, αποτελείται από ένα **Bucket**, το οποίο έχει ένα σταθερό μέγεθος χώρου, καθορισμένο από το χρήστη, στο οποίο θα αποθηκεύονται τα δεδομένα. Τα δεδομένα μπορούν να είναι οποιουδήποτε τύπου επιλέξει ο χρήστης και θα πρέπει να είναι σταθερού μεγέθους.

Η αποθήκευση των στοιχείων γίνεται σε έναν δυναμικά δεσμευμένο πίνακα που εξυπηρετεί στη γρήγορη προσπέλαση των στοιχείων. Θα πρέπει το μέγεθος του **Bucket** να είναι **τουλάχιστον** όσο το μέγεθος ενός από τα στοιχεία που πρόκειται να αποθηκευτούν σε αυτήν.

Για βελτίωση της χρήσης της μνήμης, δε δεσμεύεται το ακριβές μέγεθος **Bucket** που επιλέγει ο χρήστης, αλλά το μέγιστο δυνατό πολλαπλάσιο μνήμης των στοιχείων που πρόκειται να αποθηκευτούν. Για παράδειγμα, αν ο χρήστης ζητήσει **Bucket size 100** Bytes και ο χρήστης θέλει να αποθηκεύσει μια δομή με μέγεθος **51** Bytes, θα δεσμευτούν **51** Bytes και όχι **100**. Έτσι, σώζουμε **49** Bytes ανά **Bucket**.

### Διευκρινήσεις Sort:

Το sorting δουλεύει χρησιμοποιώντας δυο πίνακες οι οποίοι εναλλάσσονται μεταξύ των αναδρομικών κλήσεων της συνάρτησης **SimpleSortRec()**. Η **SimpleSortRec()** αρχικά δημιουργεί το **ιστόγραμμα** και τον πίνακα **psum** όπως αναγράφεται στην εκφώνηση. Έπειτα χρησιμοποιεί το **table1** ως **R** και το **table2** ως **R'**, δηλαδή σε κάθε αναδρομική κλήση ταξινομεί τον πίνακα **R** χρησιμοποιώντας το **psum** και γράφει τα **αποτελέσματα** στον **R'**. Αφού τελειώσει αυτή η διαδικασία αντιγράφουμε τα αποτελέσματα (που είναι ταξινομημένα) στον **R**. Όταν σε κάποια αναδρομική κλήση τα δεδομένα του δοθέντος **R** (ο οποίος αποτελεί ένα από τα buckets της αρχικής κλήσης), γίνουν στο πλήθος μικρότερα από **8.192** δηλαδή **64KB** τότε ο **R** ταξινομείται με **quicksort**. Έτσι τελικά, μόλις τελειώσει η **SimpleSortRec()**, έχουμε ταξινομημένο τον πίνακα που δόθηκε στην αρχική κλήση της συνάρτησης.  
(Είναι ταξινομημένοι και ο **R** και ο **R'**)

### Διευκρινήσεις ComparisonPredicate:

Το **comparison predicate** δουλεύει χρησιμοποιώντας ένα *struct compPred* το οποίο περιέχει το *comparison query*. Αν ο πίνακας υπάρχει στην ενδιάμεση μνήμη με κάποια μορφή τότε χρησιμοποιείται αυτή η μορφή για το predicate, αλλιώς χρησιμοποιείται ο αρχικός πίνακας.

### Διευκρινήσεις JoinPredicate:

Το **join predicate** δουλεύει χρησιμοποιώντας τις συναρτήσεις της προηγούμενης άσκησης. Αν οι δύο πίνακες που δίνονται υπάρχουν στην ενδιάμεση μνήμη με κάποια μορφή τότε χρησιμοποιείται αυτή η μορφή για το predicate, αλλιώς χρησιμοποιούνται οι αρχικοί πίνακες.

### Διευκρινήσεις JoinSelf:

Το **join self** χρησιμοποιείται όταν πρέπει να γίνει *join* μεταξύ δύο στηλών ενός πίνακα. Το *join* γίνεται συγκρίνοντας ανα γραμμή οι δοθείσες στείλες και κρατώντας μόνο αυτές που έχουν ίδιο κλειδί. Αν αυτός ο πίνακας υπάρχει στην ενδιάμεση μνήμη με κάποια μορφή τότε χρησιμοποιείται αυτή η μορφή για το predicate, αλλιώς χρησιμοποιείται ο αρχικός πίνακας.

### Διευκρινήσεις JoinInSameBucket:

Αυτή η συνάρτηση χρησιμοποιείται όταν θέλουμε να κάνουμε *join* μεταξύ δύο πινάκων οι οποίοι υπάρχουν ήδη στο **ίδιο** bucket της ενδιάμεσης μνήμης. Το *join* γίνεται θεωρώντας ότι έχουμε έναν ενιαίο ενδιάμεσο πίνακα από τον οποίο αρχικά παίρνουμε την πρώτη γραμμή όπου σε αυτήν υπάρχουν τα **rowIDs** του καθένα από τους δύο πίνακες. Έτσι, αν τα στοιχεία στα οποία οδηγούν τα **rowIDs** είναι ίδια, κρατάμε αυτήν την γραμμή, αλλιώς την σβήνουμε.

*Οι βοηθητικές συναρτήσεις **TableExistsInMidStruct** και **CreateTableForJoin** χρησιμοποιούνται για την εύρεση ή δημιουργία πινάκων για την εκτέλεση του *join*.*

*Τέλος, οι συναρτήσεις που καθορίζουν ποια συνάρτηση **comparison** ή **join** θα κληθεί είναι οι **DoAllJoinPred** και **DoAllCompPred**.*