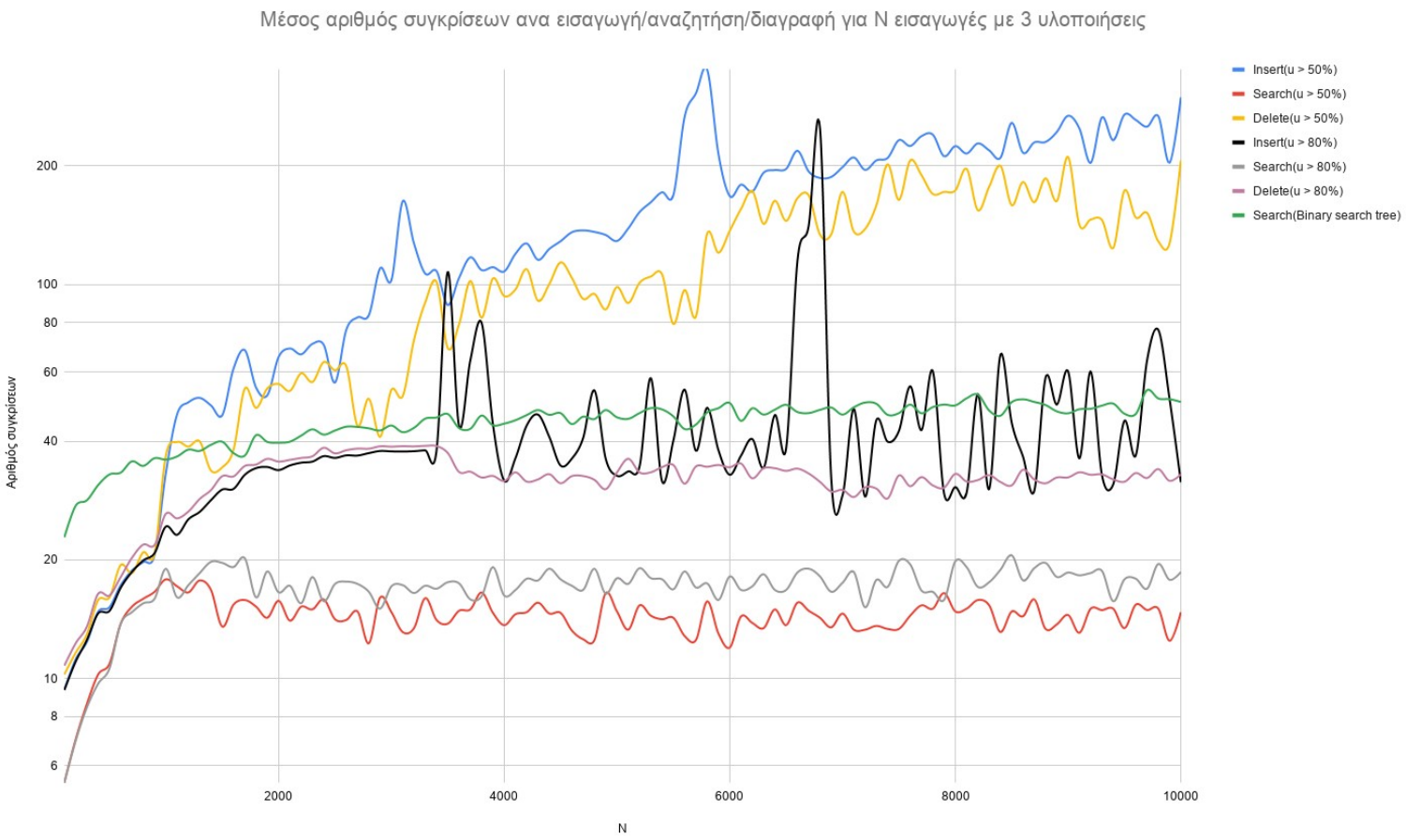
Μετρήσεις



Εισαγωγή στοιχείων

Για την εισαγωγή στοιχείων έχουμε δεδομένα για τις μεθόδους Linear Hashing με split factor u> 50% και u>80%. Όπως παρατηρούμε, αρχικά, οι δύο υλοποιήσεις είναι ίσες μέχρι N~1000 οπότε το load factor και στις δύο περιπτώσεις είναι <50%. Από εκεί και ύστερα η υλοποίηση με split factor 50% εμφανίζει πολύ περισσότερες συγκρίσεις σε σύγκριση με την άλλη. Μάλιστα παρατηρείται ότι εμφανίζει τις περισσότερες συγκρίσεις από όλες τις διεργασίες(διαγραφές, αναζητήσεις).

Στην περίπτωση με split factor u> 50% έχουμε μια αρκετά γραμμική αύξηση ενώ στην άλλη περίπτωση παρατηρώ μια λογαριθμική αύξηση της καμπύλης.

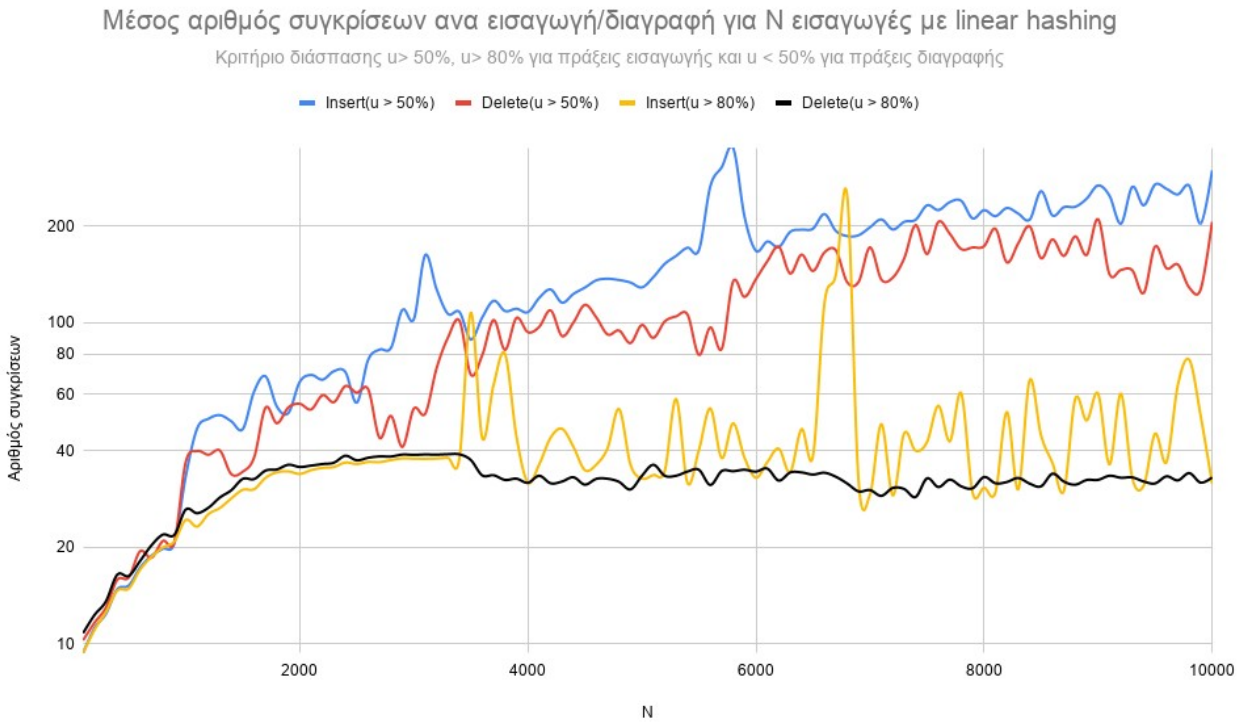
Το αποτέλεσμα είναι αναμενόμενο διότι το χαμηλό split factor της περίπτωσης με split factor u> 50% αναγκάζει σε split αρκετά νωρίς, μόλις οι μισές θέσεις του πίνακα και των overflow buckets καταληφθούν. Επιπλέον και η ύπαρξη ίσου split/merge factor αναγκάζει το load factor να ταλαντεύεται γύρω από το 50% οπότε προκαλεί συνεχόμενα splits/merges και ακολούθως πολλές συγκρίσεις. Αντιθέτως, στην άλλη περίπτωση έχουμε λιγότερα buckets αλλά και πιο ισορροπημένο αριθμό συγκρίσεων.

Αναζήτηση στοιχείων

Σε αυτή την περίπτωση παρουσιάζονται στο διάγραμμα και οι τρεις υλοποιήσεις. Παρατηρώ πως οι καμπύλες και των τριών παρουσιάζουν λογαριθμική καμπύλη σε σχέση με τις τιμές του N. Ειδικότερα, πιο αποδοτικές εμφανίζονται οι liner hashing υλοποιήσεις και συγκεκριμένα πιο βέλτιστη αυτή με split/merge factor 50%. Αυτή η αποτύπωση ήταν η αναμενόμενη καθώς το linear hashing προσφέρει καλύτερη απόδοση στην αναζήτηση λόγω της hash συνάρτησης που προσφέρει καλύτερο locality άρα και μειωμένες αναζητήσεις. Η μέθοδος με ΔΔΕ εμφανίζεται ελαφρώς χειρότερη ωστόσο φαίνεται να έχει πιο μικρή κλίση στην καμπύλη της παρά τις αρχικά σχετικα μεγάλες τιμές της.

Διαγραφή στοιχείων

Η διαγραφή στοιχείων παρουσιάζει παρόμοια συμπεριφορά με την εισαγωγή στοιχείων για τις δύο μεθόδους linear hashing. Για split/merge factor 50% παρατηρώ μια σχεδόν γραμμική καμπύλη λίγο χαμηλότερη από την αντίστοιχη της εισαγωγής. Αντίστοιχα για split factor 80% παρατηρώ μια λογαριθμική καμπύλη η οποία είναι πάλι λίγο χαμηλότερη της εισαγωγής. Αυτό είναι κάτι που το αναμέναμε διότι η διαφορά μεταξύ split-merge factor στην δεύτερη περίπτωση επιτρέπει το load factor να κυμαίνεται ανάμεσα σε αυτά τα όρια και να προκαλεί λίγες συγχωνεύσεις. Αντίθετα για ίση τιμή split-merge factor το load factor διαρκώς κινείται γύρω από το 50% και προκαλεί πολλές συγχωνεύσεις άρα και πολλές συγκρίσεις.



Όπως και στο παραπάνω διάγραμμα παρατηρείται η διαφορά μεταξύ των δύο περιπτώσεων. Για split factor u>80% και merge factor u<50% έχουμε αριθμητικά λιγότερα buckets και αυτό προσφέρει μια ισορροπία ανάμεσα στον αριθμό συγκρίσεων σε εισαγωγές, αναζητήσεις και διαγραφές. Αντίθετα το όριο split/merge factor u>50% προσφέρει περισσότερα buckets και ταχύτερες αναζητήσεις αλλά με μεγαλύτερο κόστος σε αριθμό συγκρίσεων σε εισαγωγές και αναζητήσεις. Σαν συμπέρασμα προκύπτει ότι ανάλογα τις ανάγκες μας μπορούμε να επιλέξουμε σχεδίαση ώστε να μας εξυπηρετεί καλύτερα. Η μέθοδος με split factor 80% προσφέρει αρκετά γρήγορη αναζήτηση χωρίς να έχει μεγάλο κόστος.