Δομές Δεδομένων και Αρχείων -3η άσκηση: Linear Hashing

Νίκου Γεώργιος-Νεκτάριος ΑΜ:2016030125

Η επίλυση της τρίτης άσκησης έγινε στην γλώσσα Java. Για την υλοποίηση του αλγορίθμου του Linear Hashing χρησιμοποίησα τον κώδικα που υπήρχε στο περσινό φροντιστήριο του μαθήματος. Η αρχική υλοποίηση περιείχε 3 κλάσεις: μια Main, μία κλάση για Hash Bucket και μια κλάση για τον αλγόριθμο του Linear Hashing.

Συγκεκριμένα για την κλάση του Linear Hashing οι μεταβολές που πραγματοποίησα ήταν, αρχικά να προσθέσω την μέθοδο για τις μετρήσεις και να την καλώ σε κάθε ανάθεση ή σύγκριση. Επιπλέον στην μέθοδο deleteKey,όπου γίνονται οι διαγραφές, αφαίρεσα τον έλεγχο του load factor για split ώστε να ταιριάζει με την εκφώνηση που ζητάει μόνο έλεγχο για merge. Τέλος έθεσα το κριτήριο συγχώνευσης ίσο με 0.5 και το κριτήριο διάσπασης να παίρνεται από τα ορίσματα των στιγμιοτύπων της κλάσης για τις δύο περιπτώσεις, δηλαδή για u> 50% και u> 80%.

Για την κλάση Hash Bucket δεν έκανα τροποποιήσεις, παρά μόνο πρόσθεσα την μέθοδο καταγραφής των συγκρίσεων καθώς και της κλήσης της σε περιπτώσεις αναθέσεων ή συγκρίσεων.

Στην κλάση MainClass, αρχικά, όρισα 2 στιγμιότυπα της κλάσης LinearHashing (για κριτήριο διάσπασης u> 50% και u> 80%) και ένα στιγμιότυπο για την κλάση του ΔΔΕ. Το αρχείο με τους 10000 αριθμούς εισήχθη σε ένα int array και ορίστηκαν μερικά string templates για την ευανάγνωστη εκτύπωση των αποτελεσμάτων των μετρήσεων στην κονσόλα.
Έπειτα δημιουργήθηκε ένας βρόχος 100 επαναλήψεων για την εισαγωγή των 10000 αριθμών ανά εκατοντάδες και μηδενίζονται οι μετρητές των προσβάσεων σε κάθε επανάληψη. Για τις μετρήσεις έγιναν τα παρακάτω:

- Εισαγωγή: Σε ένα βρόχο 100 επαναλήψεων εισάγονται σειριακά οι αριθμοί της εκάστοτε εκατοντάδας στις τρεις μεθόδους και προστίθενται ο αριθμός προσβάσεων για κάθε μέθοδο.
- Αναζήτηση: Γεμίζω ένα ArrayList 100 θέσεων με τους αριθμούς 1-100 και σε ένα βρόχο 50 επαναλήψεων μέσω της rand επιλέγεται ένας αριθμός από τους 100 και το στοιχείο της εκατοντάδας σε αυτή την θέση αναζητείται με τους τρεις τρόπους. Ύστερα επιστρέφεται και ενημερώνεται η τιμή των προσβάσεων και αφαιρείται από την Arraylist με remove ο συγκεκριμένος αριθμός ώστε να μην χρησιμοποιηθεί ξανά.
- Διαγραφή: Για την διεργασία της διαγραφής πραγματοποιείται ξανά η διαδικασία με την ArrayList όπως προηγουμένως. Ωστόσο δεν χρησιμοποιείται βρόχος 50 επαναλήψεων διότι σε περιπτώσεις διαγραφής όπου γίνεται άδειασμα του overflow bucket δεν διαγράφεται πάντα το στοιχείο που ορίστηκε. Έτσι για τις διαγραφές με Linear Hashing ελέγχεται αν έχουν διαγραφεί 50 στοιχεία και ύστερα τερματίζεται η επανάληψη. Αντίθετα για διαγραφή με ΔΔΕ απλά χρειάζονται 50 επαναλήψεις καθώς δεν υπάρχει τέτοιος κίνδυνος. Τέλος αφαιρείται ο αριθμός που διαγράφηκε από τις μεθόδους και ενημερώνεται ο αριθμός προσβάσεων.

Μετά το πέρας των επαναλήψεων για εισαγωγή/αναζήτηση/διαγραφή 100 αριθμών γίνεται η εκτύπωση των 9 μετρήσεων κατά μέσο όρο για το εκάστοτε N και προχωράει το πρόγραμμα στην επόμενη εκατοντάδα αριθμών.

Για την υλοποίηση με ΔΔΕ αξιοποίησα την υλοποίηση της προηγούμενης εργασίας και την εμπλούτισα με την μέθοδο για την διαγραφή αριθμών.

Το πρόγραμμα τρέχει κανονικά και περνάει χωρίς λάθη από τον compiler. Η μεγαλύτερη δυσκολία που συνάντησα ήταν στην διαγραφή των αριθμών καθώς η υλοποίηση αρχικά είχε πρόβλεψη για split και επίσης εμφάνιζε ιδιαίτερη συμπεριφορά κατά το άδειασμα ή σε περιπτώσεις υπερχείλισης των overflow buckets.

Πηγές

Ενδεικτικός κώδικας υλοποίησης Linear Hashing (φροντιστήριο 2018-2019)
http://delab.csd.auth.gr/papers/LinearHashing2017.pdf + διαφάνειες: μελέτη Linear hashing αλγορίθμου
https://github.com/Priyansh2/linear_hashing : υλοποίηση Litwin linear hashing του σε C++

https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/ : διαγραφή σε ΔΔΕ