3ο Παραδοτέο στο μάθημα Αλγόριθμοι και Δομές Δεδομένων

Ον/μο: Σταματίου Μάριος-Χρήστος

AM:1066488

Έτος: 3ο

Θέμα: αποθήκευσης & αναζήτησης δεδομένων με χρήση HASHING

Αρχικά , δημιουργούμε μία κλάση CreditCard στην οποία θα εμπεριέχονται το ID , οι μέρες επίσκεψης και τα ποσά πληρωμών που θα αντιστοιχούν σε κάθε πελάτη.

Έπειτα ξεκινάμε να κατασκευάζουμε τον κύριο κορμό δημιουργίας του HashTable. Για την υλοποίηση του με την τεχνική Open Addressing και με Linear Probing δημιουργούμε 2 συναρτήσεις , την LinearProbing και την hashFunction . Η hashFunction επιστρέφει την τιμή n mod N – όπου n η κάθε επανάληψη και N το μέγεθος του πίνακα (πρώτος αριθμός) και η linearProbing ελέγχει αν κάποια κάρτα έχει επισκεφθεί ξανά το κατάστημα ώστε να προσθέσει την επόμενη πληρωμή και να σημειώσει την ημέρα επίσκεψης , καθώς παράλληλα τοποθετεί σε κενές θέσει στον πίνακα καινούριες κάρτες και μετράει τα collisions που προκύπτουν .

Στο HashTableGenerator δημιουργούμε τη λογική που περιγράφεται στην εκφώνηση της άσκησης . Μέσα σε μία while όπου θέτουμε ως όριο τα ταξίδια που έχουν οριστεί ελέγχουμε για το load factor κάθε φορά , με το υπερβεί ελάχιστα το όριο που έχουμε θέσει (0.6) αυξάνουμε τον πίνακα κατά findClosestPrime(2*N) κενά στοιχεία όπου το findClosestPrime βρίσκει τον πιο κοντινό αριθμό κοντά στο διπλάσιο του αριθμού N (για κάθε διπλασιασμό). Έπειτα , δημιουργούμε τα τυχαία δεδομένα (ID,money,day) των επισκέψεων και τα εισάγουμε με την linear Probing. Αν , πάλι , το load factor πλησιάσει στο 0.6 ενεργοποιούμε το condition (condition=True) , όπου πυροδοτεί τον «διπλασιασμό» του hashTable.

Έπειτα στις συναρτήσεις leastMoneySpent,dayWithLeastVisits,maximumCardVisits απαντώνται

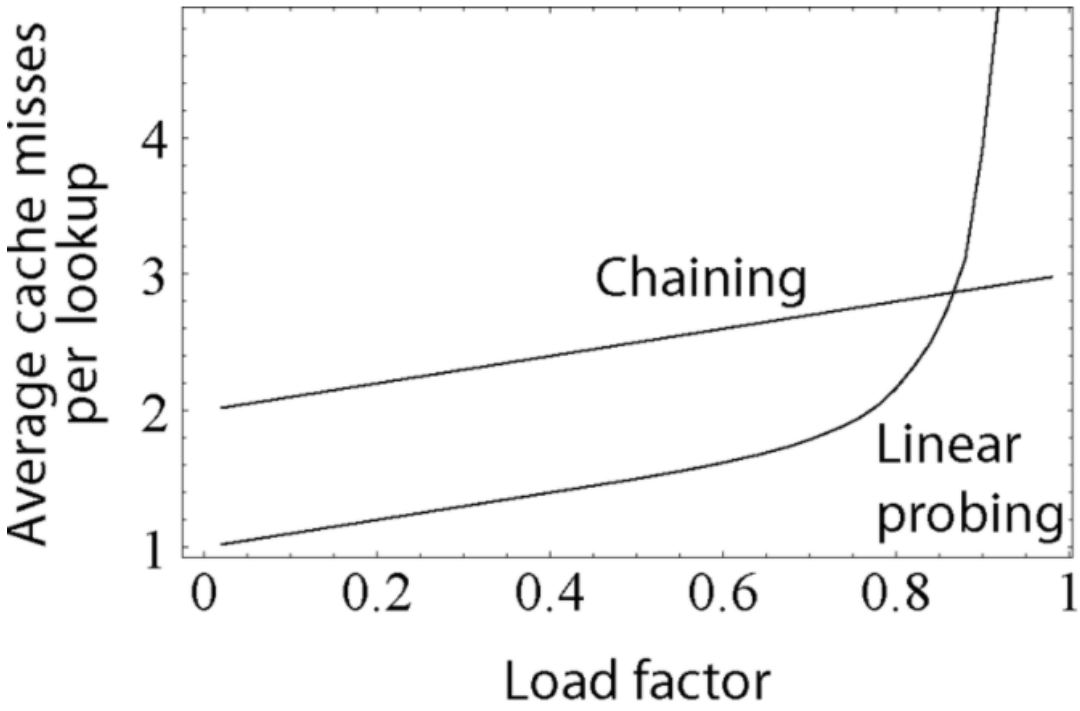
τα ερωτήματα 1,2,3 . Για την εύρεση της κάρτας με το μικρότερο ποσό πληρωμών εφαρμόζουμε τη λογική εύρεσης του μεγίστου και ελαχίστου χρονικής πολυπλοκότητας O(n). Με μία for-loop ελέγχουμε τα object του hashTable και βρίσκουμε το ελάχιστον συνολικό ποσό πληρωμών.

Αντίστοιχη λογική ακολουθούμε στην συνάρτηση maximumCardVisits μόνο που αυτή τη φορά επιλέγουμε τη μέγιστη τιμή των επισκέψεων.

Στην συνάρτηση dayWithLeastVisits δημιουργούμε μία λίστα με μηδενικά και μία με τα ονόματα των ημερών (από Δευτέρα έως Σάββατο) ελέγχουμε με μια for-loop τα στοιχεία του hashTable , και για όσες μέρες συμπεριλαμβάνονται στις επισκέψεις κάθε κάρτας επιστρέφουμε το index της λίστας days (με τα ονόματα των ημερών) και αυξάνουμε κατά 1 το αντίστοιχο index της daysCounter , όπου επιστρέφουμε στο τέλος της συνάρτησης το min της και της ημέρα στην οποία αντιστοιχεί.

Συμπεράσματα: Α) Ο χρόνος δημιουργίας του hashTable με 2.000.000 στοιχεία σχεδόν οχταπλασιάζεται από αυτόν με το hashTable των 1.000.000 στοιχείων – κάτι που αποτελεί αρκετά χρονοβόρο , καθώς ούτε ο χρόνος δημιουργίας του hashTable με τα 1.000.000 στοιχεία είναι μικρός(περίπου 1 ώρα και 35 λεπτά).

Β) Αυξάνοντας το load factor από 0.6 σε 0.7 και από 0.7 σε 0.8 , παρατηρείται , πάλι, αύξηση των συγκρούσεων που περιγράφεται – αρκετά εύστοχα – σύμφωνα με το παρακάτω διάγραμμα. Ειδικά , όταν θέτουμε το load factor σε 0.8 βλέπουμε εκθετική αύξηση .



Σημείωση: Για τον έλεγχο της ακεραιότητας των αποτελεσμάτων δεν προτείνεται η εισαγωγή μεγάλου αριθμού ταξιδιών , καθώς η διαδικασία είναι χρονοβόρα