Πανεπιστημιο Ιωαννινων, Τμημα Πληροφορικης και Τηλεπικοινωνιων

Αλγοριθμοι και Πολυπλοκοτητα

Γρήγορος υπολογισμός συγκεντρωτικών χρεώσεων πιστωτικών καρτών.

Κωνσταντίνος Ράντος
int02565@uoi.gr

Πρόβλημα

Το πρόβλημά μας ήταν να δημιουργήσουμε 1.000.000 χρεώσεις πιστωτικών καρτών σε 20.000 τυχαίους αριθμούς καρτών και να παρουσιάσουμε τα αντίστοιχα δεδομένα.Αρχικά η υλοποίηση αυτή θα πρέπει να γίνει με την χρήση κατάλληλης δεδομένων και στην συνέχεια με την χρήση ενός δικού μας πίνακα κατακερματισμού.

1 Έτοιμες δομές δεδομένων

Αρχικά, προσπάθησα να υλοποιήσω το πρόβλημα χρησιμοποιώντας pandas και ένα DataFrame. Αυτό έκανε τον κώδικα να τρέχει περίπου σε 0.35 δευτερόλεπτα, γεγονός μη ικανοποιητικό. Στη συνέχεια, υλοποίησα το ίδιο πρόβλημα με την χρήση dictionary, με τον αριθμό της κάρτας ως κλειδί και το συνολικό ποσό των χρεώσεων ως τιμή. Παρατηρήθηκε ότι ο κώδικας υλοποιείται σε μόλις 0.01-0.02 δευτερόλεπτα, δηλαδή πολύ ταχύτερα από τη χρήση ενός DataFrame.

2 Πίνακας κατακερματισμού

Στην συνέχεια με την κλάση LinearProbingHashTable και τις επιμέρους συναρτήσεις(put,get,insert,rehash) γίνεται η υλοποίηση του πίνακα κατακερματισμού με ανοικτή διευθυνσιοδότηση και γραμμική διερεύνηση.Ο πίνακας ξεκινά με μέγεθος 101 και αν ο συντελεστής φόρτωσης ξεπεράσει το 0.7, τότε ενργοποιείται

η συνάρτηση rehash και το μέγεθός του νέου πίνακα γίνεται ίσο με τον μικρότερο πρώτο αριθμό που είναι διπλάσιος του προηγούμενου μεγέθους. Στο τέλος έκανα μια προσπάθεια να παρουσιάσω τα δεδομένα των πιστωτικών καρτών κάνοντας χρήση του πίνακα κατακερματισμού, η οποία για μικρό αριθμό χρεώσεων (μέχρι 10000) εκτελείται γρήγορα, ενώ για περισσότερες χρεώσεις χρειάζεται πάρα πολύ χρόνο. Το πρόβλημα που αντιμετώπισα είναι πώς οι δύο υλοποίησεις για τον ίδιο αριθμό χρεώσεων δεν εμφανίζουν τα ίδια αποτελέσματα όπως θα έπρεπε.

3 Συμπεράσματα

Η υλοποίηση με dictionaries στην Python θεωρείται πιο γρήγορη σε σύγκριση με άλλες μεθόδους λόγω της αποτελεσματικής δομής δεδομένων που παρέχουν. Τα dictionaries στην Python χρησιμοποιούν κατακερματισμό (hashing) για τη γρήγορη αναζήτηση τιμών, επιτρέποντας στο πρόγραμμα να προσπελάσει αμέσως το αντίστοιχο κλειδί και να αντλήσει την τιμή του, χωρίς την ανάγκη για γραμμική αναζήτηση. Αυτό οδηγεί σε σημαντική επιτάχυνση στις λειτουργίες αναζήτησης και πρόσβασης σε δεδομένα, καθιστώντας τη χρήση dictionaries εξαιρετικά αποδοτική.