ΡΟΥΣΚΑ ΝΑΤΑΛΙΑ
1092581

Εργασία αποθήκευσης & αναζήτησης δεδομένων με χρήση HASHING

Στην παρούσα εργασία, έχουν υλοποιηθεί δύο αλγόριθμοι που αποθήκευουν τις συνολικές χρεώσεις σε 20000 κατόχους πιστωτικών καρτών. Ο πρώτος αλγόριθμος χρησιμοποιεί το λεξικό της python με key: id card και value: list with items(sublists charge-day). Ο δεύτερος χρησιμοποιεί ένα hash table με τα ίδια key,value το οποίο αυξάνει το μέγεθος του κάθε φορά που το load factor ξεπεράσει το 0.7. Υπολογίζονται διάφορα στατιστικά για το πλήθος και την ημέρα που χρεώνονται τυχαία οι κάτοχοι των καρτών.

Παρακάτω παρατίθενται τα αποτελέσματα του πρώτου αλγορίθμου για 1000000, 2000000, 5000000 χρεώσεις και οι χρόνοι που χρειάζονται για τη δημιουργία τους, καθώς και για τον υπολογισμό των στατιστικών.

--- 3.54seconds ---

Η κάρτα με τον μέγιστο αριθμό (80) συναλλαγών: 3029-5087-0171-3004

Η κάρτα με τον μέγιστο αριθμό (21035) χρεώσεων: 4389-5445-2208-0988

Η κάρτα με τον ελάχιστο αριθμό (5909) χρεώσεων: 7978-1988-1687-5031

Η ημέρα με τον ελάχιστο αριθμό (166448) χρεώσεων: Mon

--- 7.19seconds ---

Η κάρτα με τον μέγιστο αριθμό (140) συναλλαγών: 7119-0459-6133-0172

Η κάρτα με τον μέγιστο αριθμό (37355) χρεώσεων: 4294-7145-7532-5803

Η κάρτα με τον ελάχιστο αριθμό (15284) χρεώσεων: 9493-8816-7561-2031

Η ημέρα με τον ελάχιστο αριθμό (332758) χρεώσεων: Wed

ΡΟΥΣΚΑ ΝΑΤΑΛΙΑ
1092581

--- 15.83seconds ---

Η κάρτα με τον μέγιστο αριθμό (316) συναλλαγών: 1275-4921-4501-5431

Η κάρτα με τον μέγιστο αριθμό (84836) χρεώσεων: 3170-3105-3976-8798

Η κάρτα με τον ελάχιστο αριθμό (44930) χρεώσεων: 8008-6959-3485-7770

Η ημέρα με τον ελάχιστο αριθμό (832090) χρεώσεων: Wed

Παρατηρούμε ότι η πολυπλοκότητα του αλγορίθμου είναι O(n), αφού ο λόγος του χρόνου εκτέλεσης προς το πλήθος των χρεώσεων είναι σχεδόν σταθερός.

3.54 για 1εκατ. χρεώσεις

(7.19/2=3.595)

(15.83/5=3.166)

Παρακάτω παρατίθενται τα αποτελέσματα του δεύτερου αλγορίθμου για 1000000, 2000000, 5000000 χρεώσεις και οι χρόνοι που χρειάζονται για τη δημιουργία τους, καθώς και για τον υπολογισμό των στατιστικών.

--- 5.37 seconds ---

Η κάρτα με τον μέγιστο αριθμό (80) συναλλαγών: 3029-5087-0171-3004

Η κάρτα με τον μέγιστο αριθμό (21035) χρεώσεων: 4389-5445-2208-0988

Η κάρτα με τον ελάχιστο αριθμό (5909) χρεώσεων: 7978-1988-1687-5031

Η ημέρα με τον ελάχιστο αριθμό (230758) χρεώσεων: Fri

--- 10.34seconds ---

Η κάρτα με τον μέγιστο αριθμό (140) συναλλαγών: 7726-4202-6592-4554

Η κάρτα με τον μέγιστο αριθμό (37355) χρεώσεων: 4294-7145-7532-5803

Η κάρτα με τον ελάχιστο αριθμό (15284) χρεώσεων: 9493-8816-7561-2031

Η ημέρα με τον ελάχιστο αριθμό (465899) χρεώσεων: Wed

ΡΟΥΣΚΑ ΝΑΤΑΛΙΑ
1092581

```
--- 28.59 seconds ---
Η κάρτα με τον μέγιστο αριθμό (316) συναλλαγών: 1275-4921-4501-5431
Η κάρτα με τον μέγιστο αριθμό (84836) χρεώσεων: 3170-3105-3976-8798
Η κάρτα με τον ελάχιστο αριθμό (44930) χρεώσεων: 8008-6959-3485-7770
Η ημέρα με τον ελάχιστο αριθμό (1165486) χρεώσεων: Tue
```

5.37 για 1εκατ. χρεώσεις

(10.34/2=5.17)

(28.59/5=5.718)

Παρατηρούμε ότι η πολυπλοκότητα του αλγορίθμου είναι πάλι O(n), αφού ο λόγος του χρόνου εκτέλεσης προς το πλήθος των χρεώσεων είναι σχεδόν σταθερός. Ωστόσο, οι χρόνοι εκτέλεσης σε αυτή την περίπτωση είναι λίγο μεγαλύτεροι από ότι στον προηγούμενο αλγόριθμο. Αυτό εξηγείται, λόγω του ότι υλοποιούμε το δικό μας hash table, ένω στη πρώτη περίπτωση το λεξίκο είναι ένα είδος hash table ήδη ενσωματώμενο ως δομή δεδομένων στη python. Παρόλο που ο χρόνος αναζήτησης στο hash table είναι O(1), η δημιουργία των n χρεώσεων αυξάνει την πολυπλοκότητα σε O(n), κάτι που είναι αναπόφευκτο και στους δύο αλγορίθμους