



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Αλγόριθμοι και Δομές Δεδομένων

PROJECT 3

ΧΡΗΣΤΑΚΗΣ ΜΑΚΑΡΙΟΣ | ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

1 Διαδικασία

Ο συνημμένος κώδικας αναπτύχθηκε σε γλώσσα Python και υλοποιεί έναν αλγόριθμο δημιουργίας 1 εκατομμυρίου τυχαίων πιστωτικών καρτών και οργάνωσής τους σε έναν hash table. Επιπλέον πραγματοποιείται ψαξίμο για το ποια κάρτα έχει τα περισσότερα χρήματα στον αριθμό της (cost) όπως και ποιά έχει τις περισσότερες αγορές (visits).

2 Σύντομος σχολιασμός του κώδικα

2.1 Συνάρτηση makecards

Η συνάρτηση αυτή παίρνει ως όρισμα μια (κενή/αρχικοποιημένη με 0) λίστα (ο hash table μας) και την γεμίζει με στοιχεία καρτών τα οποία είναι λίστες της μορφής [string card_number, int cost, int visits], όπου:

- Card_number είναι ο τυχαία δημιουργημένος αριθμός της κάρτας, βάζοντας τα γράμματα ABCD σε τυχαίες αλλά διαφορετικές θέσεις του αλφαριθμητικού '1234567890123456'
- Cost είναι ένας τυχαίος αριθμός στο διάστημα [10,1000] που αντιπροσωπεύει το ποσό της συναλλαγής
- Visits είναι ο αριθμός των επισκέψεων/πληρωμών που έχουν γίνει με αυτή τη συγκεκριμένη κάρτα.

Τα στοιχεία αυτά αφού περάσουν από την hash function polyhash() προσθέτονται στη λίστα με χρήση της συνάρτησης addtolist(), η οποία μας υποδεικνύει αν υπήρξε σύγκρουση καθώς προσθέταμε το νέο στοιχείο στον hash table.

Τέλος επιστρέφει τον αριθμό των συγκρούσεων που προέκυψαν.

2.2 Συνάρτηση polyhash

Η συνάρτηση αυτή παίρνει ως όρισμα ένα αλφαριθμητικό και με βάση αυτό, κατασκευάζει μια λίστα με τους συντελεστές ενός πολυωνύμου μετατρέποντας κάθε χαρακτήρα του string στον αντίστοιχο ASCII αριθμό του.

Έπειτα μέσω κλήσης της συνάρτησης horner(), υπολογίζεται η τιμή του πολυωνύμου για $x = 33$ (εμπειρικά για αυτό το x έχουμε τις λιγότερες συγκρούσεις) και γίνεται mod με τον πρώτο (prime) αριθμό 87383, ο οποίος είναι ο αμέσως μεγαλύτερος πρώτος αριθμός από το διπλάσιο των πιθανών συνδυασμών καρτών.

2.3 Συνάρτηση addtolist

Με αυτή την συνάρτηση προσθέτουμε στον hash table (μεταβλητή cards_list) ένα στοιχείο κάρτας που έχει αντιστοιχηθεί μέσω της hash function στη θέση index του πίνακα.

Εάν η θέση είναι ήδη κατελλειμένη χρησιμοποιείται η τεχνική του open addressing, όπου ξεκινώντας από τη αυτή ψάχνεται η επόμενη ελεύθερη θέση στον πίνακα και αποθηκεύεται σε αυτή και έπειτα επιστρέφεται 1 για να υποδηλώσει πως υπήρξε σύγκρουση (collision) κατά την προσθήκη της κάρτας στον πίνακα.

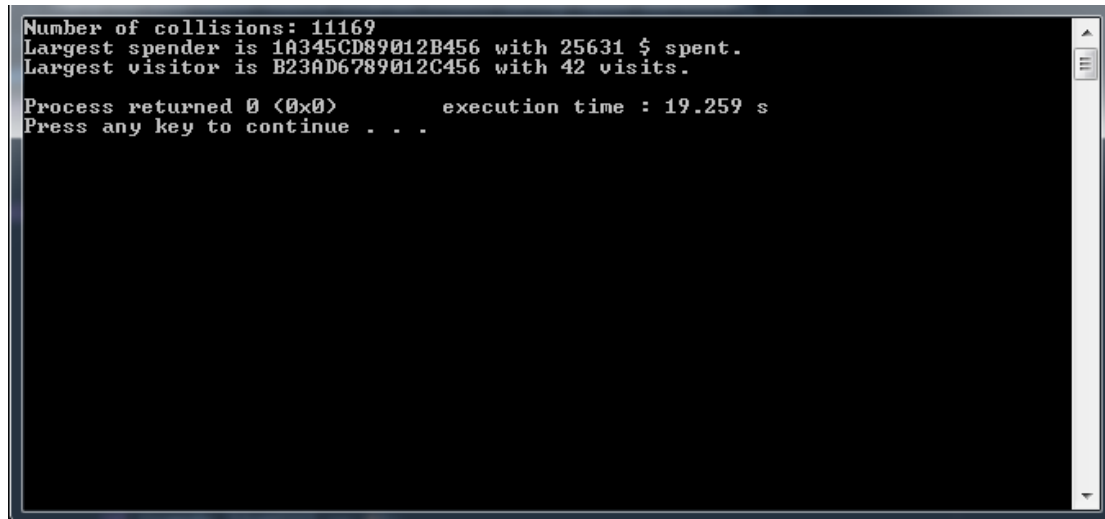
Αν όμως βρεθεί η ίδια κάρτα (ίδιο όνομα κάρτας) στον πίνακα, τότε προστίθεται το ποσό της καινούριας συναλλαγής (cost) στο τρέχων και αυξάνεται ο αριθμός των επισκέψεων (visits) για την κάρτα αυτή. Στην περίπτωση αυτή **δεν θεωρούμε πως υπάρχει σύγκρουση**, απλά ψάχνεται η ήδη τοποθετημένη κάρτα για ενημέρωση των στοιχείων της.

2.4 Συνάρτηση find_largest

Η συνάρτηση αυτή διατρέχει τον hash table και προσδιορίζει την θέση της κάρτας με τα περισσότερα ξοδεμένα λεφτά, όπως και της κάρτας με τις περισσότερες επισκέψεις και επιστρέφει αυτούς τους δυο δείκτες. Δυστυχώς αυτός ο αλγόριθμος είναι $O(N)$ και όχι $O(1)$ διότι δεν μπορούμε να ξέρουμε εκ των προτέρων ποια κάρτα θα έχει τα χαρακτηριστικά αυτά, έτσι ώστε να ψάξουμε για αυτή συγκεκριμένα.

3 Αποτελέσματα

Το αποτέλεσμα της εκτέλεσης φαίνεται παρακάτω:



```
Number of collisions: 11169
Largest spender is 1A345CD89012B456 with 25631 $ spent.
Largest visitor is B23AD6789012C456 with 42 visits.

Process returned 0 (0x0)          execution time : 19.259 s
Press any key to continue . . .
```