Περιγραφή Αλγορίθμων και Δομών Δεδομένων

Γενικά

Για την αποθήκευση των δεδομένων χρησιμοποιείται ένας hash πίνακας open addressing, η τεχνική table doubling ώστε ο πίνακας hash να μεγαλώνει ώστε να μπορεί να αποθηκεύσει αποτελεσματικά τα δεδομένα και η τεχνική linear hashing για τη διαχείριση των πιθανών συγκρούσεων. Τέλος, χρησιμοποιούνται και διάφορες απλές βοηθητικές συναρτήσεις όπως για παράδειγμα η next_prime στο αρχείο prime.py, οι οποίες δε θα αναφερθούν στο παρόν κείμενο αλλά ο ενδιαφερόμενος μπορεί να ανατρέξει στην υλοποίηση τους.

Συνάρτηση Hash

Η συνάρτηση hash που χρησιμοποιείται βρίσκεται στο αρχείο hash.py με όνομα hash και αντιστοιχεί μία λέξη-κάρτα σε ένα αριθμό από 0 μέχρι m-1, όπου m το εκάστοτε μέγεθος του πίνακα. Η τιμή hash της κάθε λέξης-κάρτα παράγεται από τη σχέση,

$$\left(\sum_{k=0}^{15} ord(c_k)p^k\right) mod\ m$$

όπου $ord(c_k)$ η Unicode τιμή του κ-οστού χαρακτήρα και p ο πρώτος, πρώτος αριθμός μετά το 62 (26 κεφαλαία + 26 μικρά + 10 αριθμοί).

Πίνακας Hash Open Addressing – Linear Hashing

Ο πίνακας hash είναι αυτός όπου αποθηκεύονται τα δεδομένα. Αφού βρεθεί η hash τιμή μιας λέξης-κάρτα, εισάγονται τα δεδομένα που αντιστοιχούν στη συγκεκριμένη συναλλαγή στον πίνακα. Αν η κάρτα έχει χρησιμοποιηθεί ξανά, τα νέα δεδομένα συμπεριλαμβάνονται στα ήδη υπάρχοντα που αφορούν τη συγκεκριμένη λέξη-κάρτα, αλλιώς αποθηκεύονται σε κάποια κενή θέση. Αρχικά, η υποψήφια θέση να υπάρχουν δεδομένα για τη λέξη-κάρτα είναι αυτή στη θέση της τιμής hash της στον πίνακα. Αν αυτή είναι κενή τότε σημαίνει ότι δεν έχει αποθηκευτεί πριν κάτι σχετικό με τη λέξη-κάρτα. Αν όμως δεν είναι τότε είτε τα δεδομένα που καταλαμβάνουν το χώρο έχουν σχέση με τη λέξη-κάρτα, είτε όχι. Στην περίπτωση που έχουν συμπεριλαμβάνονται τα νέα δεδομένα στα ήδη υπάρχοντα. Σε διαφορετική περίπτωση υπάρχει σύγκρουση και ακολουθείτε η διαδικασία linear hashing. Σύμφωνα με αυτή, εξετάζονται όλες οι επόμενες θέσεις του πίνακα με αντίστοιχο τρόπο μέχρις ότου είτε συμπεριληφθούν τα δεδομένα σε είδη υπάρχοντα ή βρεθεί κάποια κενή θέση για να αποθηκευτούν. Με τη μέθοδο αυτή, τα δεδομένα κάποιας λέξης-κάρτα, αν υπάρχουν, θα βρίσκονται σε κάποια από τις θέσεις της hash τιμής της μέχρι την πρώτη κενή θέση.

Πίνακας Hash Open Addressing – Table Doubling

Στο σημείο αυτό αξίζει να αναφερθεί πως το μέγεθος τους hash πίνακα είναι πάντα πρώτος αριθμός. Όταν ο λόγος των δεσμευμένων θέσεων προς το συνολικό μέγεθος ξεπεράσει κάποια τιμή (0.4, 0.5, 0.6) τότε δημιουργείτε νέος πίνακας hash με μέγεθος τον πρώτο, πρώτο αριθμό μετά το διπλάσιο του παλιού μεγέθους και σε αυτόν αποθηκεύονται όλα τα προηγούμενα δεδομένα ανάλογα με τη νέα τιμή hash της λέξης-κάρτα στην οποία αντιστοιχούν (η τιμή hash κάθε λέξης-κάρτα εξαρτάται από το μέγεθος του πίνακα hash). Με τον τρόπο αυτό, ελαχιστοποιούνται οι νέες συγκρούσεις που ενδέχεται να προκύψουν και μεγαλώνει η χωρητικότητα του πίνακα hash ώστε να δεχθεί νέα δεδομένα.

Απαντήσεις στις ερωτήσεις

Για την εύρεση της κάρτας με το μεγαλύτερο ποσό πληρωμών, της κάρτας με το μεγαλύτερο αριθμό επισκέψεων και της ημέρας με το μεγαλύτερο πλήθος επισκέψεων, εξετάζεται μία φορά, μία-μία, κάθε λέξη-κάρτα και τα δεδομένα που της αντιστοιχούν. Για την εύρεση του πλήθους των συγκρούσεων, γίνεται χρήση της μεταβλητής collisions η οποία αυξάνεται κατά ένα κάθε φορά που η θέση η οποία εξετάζεται για εισαγωγή δεδομένων δεν είναι κενή και δεν αντιστοιχεί στην λέξη-κάρτα στην οποία αντιστοιχούν τα δεδομένα προς εισαγωγή. Σημειώνεται ότι κάθε φορά που γίνεται table doubling η μεταβλητή collisions λαμβάνει την τιμή 0.

Έξοδος προγράμματος και παρατηρήσεις

Παρακάτω παρουσιάζεται η έξοδος του προγράμματος για τρεις διαφορετικές τιμές του μέγιστου load factor, του λόγου δηλαδή των χρησιμοποιημένων θέσεων προς το συνολικό πλήθος των διαθέσιμων θέσεων του hash πίνακα.

```
discrypt@olivia:Up Problem Set 3(master)$ python hash.py
The hash table H is of size 130027, has 43680 places occupied and its load factor is 0.34.
Card 12CA5678901234DB (hash value: 58462) visited 39 times and has spent the most (2613). It can be found at H[58462].
Card 1A345678901C3B5D (hash value: 48665) has the most visits (47) and it can be found at H[48665].
Tuesday is the day with the most visits (143210).
There were 241281 collisions in the final table.
```

Figure 1 Max Load factor 0.4

```
discrypt@olivia:Up Problem Set 3(master)$ python hash.py
The hash table H is of size 130027, has 43680 places occupied and its load factor is 0.34.
Card 12CA5678901234DB (hash value: 58462) visited 39 times and has spent the most (2613). It can be found at H[58462].
Card 1A345678901C3B5D (hash value: 48665) has the most visits (47) and it can be found at H[48665].
Tuesday is the day with the most visits (143210).
There were 239813 collisions in the final table.
```

Figure 2 Max Load factor 0.5

```
discrypt@olivia:Up Problem Set 3(master)$ python hash.py
The hash table H is of size 130027, has 43680 places occupied and its load factor is 0.34.
Card 12CA5678901234DB (hash value: 58462) visited 39 times and has spent the most (2613). It can be found at H[58462].
Card 1A345678901C3B5D (hash value: 48665) has the most visits (47) and it can be found at H[48665].
Tuesday is the day with the most visits (143210).
There were 234034 collisions in the final table.
```

Figure 3 Max Load factor 0.6

Παρατηρούμε ότι όλα τα αποτελέσματα εκτός του αριθμού των συγκρούσεων είναι ανεξάρτητα του μέγιστου load factor που επιλέγεται, όπως ήταν αναμενόμενο. Ωστόσο, όσο μεγαλώνει ο μέγιστος load factor, μικραίνει ο αριθμός των συγκρούσεων στον τελικό πίνακα. Τούτο διότι, αφού το μέγεθος του τελικού πίνακα είναι το ίδιο, όσο μεγαλύτερος ο μέγιστος load factor, τόσο περισσότερες συναλλαγές με την ίδια κάρτα θα αντιμετωπίζονται ως μία δεσμίδα δεδομένων και όχι ως μεμονωμένες συναλλαγές. Έστω, δηλαδή, ότι οι συναλλαγές μίας κάρτας είναι n και δημιουργούν k συγκρούσεις στον τελικό πίνακα, τότε $\frac{k}{n} = \lambda$, ακέραιος. Αν ομαδοποιήσουμε h από τις n συναλλαγές πριν εισαχθούν, τότε ο αριθμός των συγκρούσεων μειώνεται σε $\lambda(n-h+1) = \frac{k(n-h+1)}{n} = k\left(1 - \frac{h-1}{n}\right) < k$.