TasPython
2η Σειρά Ασκήσεων

TasPython Team*
www.taspython.tk

27 Απριλίου 2009

1 Sorting and Searching

1.1 Sorting

Γράψτε ένα απλό πρόγραμμα το οποίο ταξινομεί μια λίστα με ακεραίους. Επιβεβαιώστε την ορθή λειτουργία του με έναν ελεγκτή ορθότητας[1].

Bonus: Κάντε το παραπάνω με δύο τρόπους. Οι πιο απλοί τρόποι ίσως να είναι insertion sort, bublesort, quicksort.

Υπόδειξη: Ως ελεγκτής ορθότητας μπορεί να χρησιμοποιηθεί ένα προγραμμα που κάνει iterate πάνω από τα ταξινομένα στοιχεία και επιβεβαιώνει ότι κάθε στοιχείο i είναι μικρότερο ή ίσο από το $i + 1$ συγκρίνοντας τα.

1.2 Searching

Υλοποιήστε τον αλγόριθμο binary search. Ο αλγόριθμος αυτός δέχεται ως είσοδο ένα ταξινομένο πίνακα (ή λίστα) και ελέγχει πάντα το μεσαίο στοιχείο των (υπολειπόμενων) στοιχείων. Αναλόγως με το αποτέλεσμα της σύγκρισης συνεχίζεται ο έλεγχος για την αναζήτηση της τιμής στόχου μέχρι να βρεθεί η επιθυμητή τιμή. Πιο επίσημα, παρατίθεται ο αλγόριθμος 1.

Σε έρευνα που έγινε σε επαγγελματίες προγραμματιστές, μετά από αρκετές ώρες δουλειάς, διαπιστώθηκε πως το αρκετά ένα μεγάλο ποσοστό (άνω του 90%) δεν κατάφερε να υλοποιήσει σωστά τον αλγόριθμο. Για αυτό βέβαια οφείλονται και δυσκολίες που σε ένα πρώτο επίπεδο δεν είναι τόσο εμφανείς.

*Υπεύθυνος Άσκησης: Δημήτρης Λεβεντέας

[1] Ένα πρόγραμμα το οποίο επιβεβαιώνει ότι η λίστα είναι ταξινομημένη

Algorithm 1 Binary Search

Require: Sorted list a of comparable elements. Length of list n. Value x of an element.

Ensure: The index that corresponds to the value x. If it does not exist, then return failure

```
1: L = 0
2: R = n + 1
3: p = ⌊(R − L)/2⌋
4: while p ≠ 0 do
5:     p = ⌊(R − L)/2⌋
6:     p = L + p
7:     if a[p] < x then
8:         L = p
9:     else if a[p] > x then
10:        R = p
11:    else
12:        return p
13:    end if
14: end while
15: return fail
```

2 Δομές Δεδομένων

2.1 Dictionaries

2.1.1 Τηλεφωνικός Κατάλογος

Δημιουργήστε δυο λίστες με n στοιχεία. Η πρώτη θα έχει n ονόματα ενώ η δεύτερη θα έχει n τηλέφωνα. Στην συνέχεια, γράψτε μια συνάρτηση που συνδιάζει αυτές τις δύο λίστες και παράγει μέσω μιας δομής *dictionary* έναν τηλεφωνικό κατάλογο.

2.1.2 Υπολογισμός Μ.Ο. Μαθημάτων

Ζητήστε από τον χρήστη να εισάγει ένα όνομα μαθήματος και το βαθμό που πήρε σε αυτό. Ύστερα με την χρήση λεξικού, θεωρήστε ως:

- Κλειδί (key) το όνομα του μαθήματος
- Τιμή (value) το βαθμό σε αυτό το μάθημα

και αποθηκεύστε τα στην δομή.

Στην συνέχεια, αφού τυπώσετε ως επιβεβαίωση τα δεδομένα που εισήγαγε ο χρήστης διατρέχοντας το λεξικό, υπολογίστε τον μέσο όρο (Μ.Ο.) με ακρίβεια δυο ψηφίων και τυπώστε τον.

2.2 Hashing

2.2.1 Συνάρτηση Εισόδου Χρήστη

Μέσω του *hashing* μπορούμε να αντιστοιχίσουμε σε ένα σύνολο τιμών, ένα μικρότερο όσον αφορά την πληθικότητα σύνολο τιμών.

Η συνάρτηση *hash()* της Python παίρνει ως όρισμα ένα αλφαριθμητικό (string) και επιστρέφει έναν αριθμό.

Θα χρησιμοποιήσουμε την hash καθώς και το τελεστή υπόλοιπο διαίρεσης (modulo) σε αυτό το πρόβλημα. Θα έχουμε μια μυστική λέξη (password), στην οποία θα κάνουμε hash και στην συνέχεια modulo κατά έναν τριψήφιο ή τετραψήφιο ακέραιο αριθμό. Αυτός ο αριθμός (KEY), όπως και η μυστική λέξη ($password$), θα ορίζονται στην αρχή του προγράμματος.

Στην συνέχεια θα δίνονται 3 ευκαιρίες στον χρήστη να μαντέψει την μυστική λέξη προκειμένου να του εμφανίσει ένα μήνυμα. Στην λέξη που θα εισάγεται από τον χρήστη, θα γίνεται hashing και moduloκαι στην συνέχεια θα συγκρίνεται με την τιμή που προέκυψε από την πραγματική μυστική λέξη. Αν οι δυο τιμές ταιριάζουν, τότε με μεγάλη πιθανότητα έχει εισαχθεί η μυστική λέξη. Προαιρετικά, στην συνέχεια μπορείτε να εξακριβώνεται αν όντως έχει εισαχθεί η σωστή λέξη.

2.3 Tuples

2.3.1 Έλεγχος Σύγκρουσης Μπαλλών

Χρησιμοποιόντας πλειάδες, μοντελοποιήστε μπάλλες στις 3 διαστάσεις. Για αυτό τον σκοπό χρησιμοποιήστε μια πλειάδα (x, y, z) όπου (x, y) δείχνει το κέντρο μιας μπάλλας, ενώ z την ακτίνα της.

Μια πιθανή σύγκρουση πρέπει να ανιχνεύεται, αν το άθροισμα των ακτινών $z_1 + z_2$ τους είναι μεγαλύτερο από την απόσταση που απέχουν τα κέντρα τους $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

3 Έλεγχος εισόδου – Σύνθετοι Βρόγχοι Επανάληψης

Οι βρόγχοι επανάληψης αποτελούν ένα κύριο μέρος της προγραμματιστικής ρουτίνας και είναι τα σημεία που πρέπει να είμαστε περισσότεροι προσεκτικοί καθώς συνήθως αποτελούν bottleneck στον κώδικα μας. Για τον σκοπό αυτό λοιπόν, στην παρούσα άσκηση θα υλοποιήσουμε ένα σχετικά απλό παιχνίδι, που όμως απαιτεί λίγη προσοχή.

3.1 Παιχνίδι με Σπίρτα

Υπάρχουν δυο παίκτες. Το παιχνίδι αρχίζει με 100 σπίρτα. Ο κάθε ένας παίκτης, μπορεί να τραβήξει από $[1, 5]$ σπίρτα. Νικητής είναι αυτός που τραβάει τραβάει τα τελευταία σπίρτα. Πρώτα παίζει ο παίκτης A, και ύστερα ο παίκτης B, εναλλάξ.

Υπόδειξη: Υλοποιήστε ένα διπλά εμφολευμένο βρόγχο, όπου στον εξωτερικό βρόγχο, συνεχίζεται το παιχνίδι όσο απομένουν σπίρτα και γίνεται εναλλαγή των παικτών, ενώ στον εσωτερικό, ζητείται έγκυρη είσοδος από τον χρήστη.

4 Παρατηρήσεις

Οι απαντήσεις των ασκήσεων θα αναρτηθούν την Παρασκευή 1 Μαϊου 2009.

Ο υπεύθυνος για αυτές θα καθοριστεί μέσω email μέχρι την Πέμπτη 30 Απριλίου 2009.

Για οποιεσδήποτε απορίες ή υποδείξεις χρησιμοποιήστε το forum.