Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

2η Σειρά Προγραμματιστικών Ασκήσεων - Ημ/νία Παράδοσης 16/11/2009

Άσκηση 1: Αλγόριθμοι Ταξινόμησης

Να υλοποιήσετε τους αλγόριθμους ταξινόμησης Insertion-sort (πιθανότατα τον έχετε υλοποιήσει ήδη για την 1η προγραμματιστική εργασία), Heapsort, Mergesort, και Quicksort, και να πειραματιστείτε για διάφορες ακολουθίες αριθμών συγκρίνοντας τον χρόνο εκτέλεσης και το πλήθος των συγκρίσεων μεταξύ στοιχείων του πίνακα για τους παραπάνω αλγόριθμους.

Για διάφορες τιμές του n, $200 \leq n \leq 5000$, να αρχικοποιήσετε τον πίνακα με n τυχαίους ακέραιους αριθμούς, να εφαρμόσετε τους παραπάνω αλγόριθμους, να σημειώσετε τον χρόνο εκτέλεσης και το πλήθος των συγκρίσεων σε κάθε περίπτωση, και να τα αναπαραστήσετε γραφικά (θα φτιάξετε δύο γραφικές παραστάσεις, μία για τον χρόνο εκτέλεσης και μία για το πλήθος των συγκρίσεων, σε αμφότερες θα υπάρχει μία καμπύλη για κάθε αλγόριθμο). Τα αρχεία των γραφικών παραστάσεων πρέπει να "ανέβουν" στο moodle μαζί με το πρόγραμμά σας.

Άσκηση 2: Πιθανοτικός και Ντετερμινιστικός Υπολογισμός Ενδιάμεσου Στοιχείου

(α) Να υλοποιήσετε τον πιθανοτικό αλγόριθμο "διαίρει-και-βασίλευε" που υπολογίζει το ενδιάμεσο στοιχείο (median) ενός (μη ταξινομημένου) πίνακα ακεραίων σε αναμενόμενο γραμμικό χρόνο.

(β) Να υλοποιήσετε τον ντετερμινιστικό αλγόριθμο "διαίρει-και-βασίλευε" που υπολογίζει το ενδιάμεσο στοιχείο (median) σε ενός (μη ταξινομημένου) πίνακα ακεραίων σε γραμμικό χρόνο.

(γ) Για διάφορες τιμές του n, $200 \leq n \leq 5000$, να αρχικοποιήσετε τον πίνακα με n τυχαίους ακέραιους αριθμούς, να εφαρμόσετε τους παραπάνω αλγόριθμους, και να συγκρίνετε τον χρόνο εκτέλεσης και το πλήθος των συγκρίσεων μεταξύ στοιχείων του πίνακα που απαιτούνται για την εύρεση του ενδιάμεσου στοιχείου. Να αναπαραστήσετε τα αποτελέσματα της σύγκρισης γραφικά όπως στο προηγούμενο ερώτημα.

Άσκηση 3: Εισιτήρια Διαρκείας

Ένας εργαζόμενος έχει καθορίσει ποιες από τις επόμενες T ημέρες θα πάει στο γραφείο με το αυτοκίνητο. Θεωρούμε λοιπόν δεδομένη μια συνάρτηση $\text{drive} : \{1, \ldots, T\} \mapsto \{0, 1\}$ τέτοια ώστε $\text{drive}(i) = 1$ αν ο εργαζόμενος πάει στη δουλειά με το αυτοκίνητο, και $\text{drive}(i) = 0$ διαφορετικά, $i = 1, \ldots, T$. Ο εργαζόμενος μπορεί να παρκάρει μόνο στον στεγαζόμενο χώρο κοντά στο γραφείο για τον οποίο όμως χρειάζεται εισιτήριο. Υπάρχουν διαθέσιμοι k διαφορετικοί τύποι εισιτηρίων: $(d_1, c_1), \ldots, (d_k, c_k)$, όπου $d_1 < \cdots < d_k$, $c_1 < \cdots < c_k$, και το εισιτήριο j έχει διάρκεια d_j ημερών και κοστίζει c_j ευρώ, $j = 1, \ldots, k$. Υποθέτουμε ότι όσο μεγαλύτερη διάρκεια έχει ένας τύπος εισιτηρίου, τόσο μικρότερο είναι το κόστος ανά ημέρα (δηλ. αν $d_i > d_j$, τότε $c_i/d_i < c_j/d_j$).

Για παράδειγμα, μπορεί να υπάρχει ημερήσιο εισητήριο με κόστος 1 ευρώ, εβδομαδιαίο με κόστος 5 ευρώ, μηνιαίο με κόστος 18 ευρώ, εξαμηνιαίο με κόστος 90 ευρώ, και ετήσιο με κόστος 120 ευρώ.

Να διατυπώσετε και να υλοποιήσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο για τον υπολογισμό μιας συλλογής εισιτηρίων ελάχιστου κόστους που καλύπτει όλες τις ημέρες που ο εργαζόμενος πηγαίνει στο γραφείο με το αυτοκίνητο. Ποιός είναι ο χρόνος εκτέλεσης του αλγόριθμου που διατυπώσατε;

Για την υλοποίηση, το πρόγραμμά σας πρέπει να διαβάζει την είσοδό του από αρχείο που στην πρώτη του γραμμή αναφέρει τα T και k, στη δεύτερη γραμμή αναφέρει τις T τιμές της συνάρτησης drive, και στις k επόμενες γραμμές το ζεύγος (διάρκεια, κόστος) για κάθε τύπο εισιτηρίου. Για παράδειγμα, το παρακάτω αρχείο αντιστοιχεί σε ένα σενάριο 14 ημερών με 3 είδη εισιτηρίων:

```
14 3
1 0 1 1 1 1 1 0 1 0 0 0 0 1
1 1
7 5
14 10
```

Η έξοδος του προγράμματος θα αναφέρει στην πρώτη γραμμή το συνολικό κόστος και το πλήθος ℓ των εισιτηρίων που αγοράζονται, και σε καθεμία από τις ℓ επόμενες γραμμές, την ημέρα, τη χρονική διάρκεια, και το κόστος κάθε εισιτηρίου που αγοράζει η βέλτιστη λύση. Έτσι η έξοδος για το προηγούμενο παράδειγμα είναι:

```
7 3
1 1 1
3 7 5
14 1 1
```