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

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

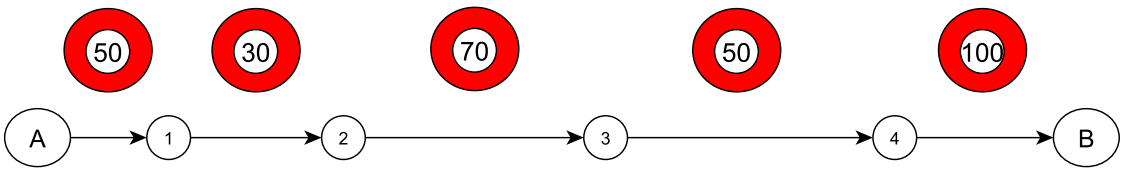
2η Σειρά Γραπτών Ασκήσεων - Ημ/νία Παράδοσης 22/12/2011

Άσκηση 1: Επιτροπή Αντιπροσώπων (ΚΤ 4.15)

Θεωρούμε n εθελοντές που εργάζονται, καλύπτοντας από μία βάρδια ο καθένας κάθε εβδομάδα, για έναν κοινωφελή σκοπό. Η βάρδια κάθε εθελοντή i καθορίζεται από τους χρόνους έναρξης s_i και ολοκλήρωσης f_i, και αντιστοιχεί στο συνεχές χρονικό διάστημα $[s_i, f_i)$. Ο επικεφαλής της προσπάθειας θέλει να επιλέξει μια επιτροπή αντιπροσώπων των εθελοντών με την οποία θα συναντιέται τακτικά. Επιθυμεί η επιτροπή να έχει όσο το δυνατόν λιγότερα μέλη, και θεωρεί ότι αυτή θα είναι πλήρης αν για κάθε εθελοντή i, υπάρχει στην επιτροπή τουλάχιστον ένας εθελοντής που η βάρδιά του επικαλύπτεται με αυτή του i. Να διατυπώσετε έναν αποδοτικό αλγόριθμο για τη βέλτιστη επιλογή της επιτροπής. Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας.

Άσκηση 2: Βιαστικός Μοτοσυκλετιστής

Ένας μοτοσυκλετιστής ταξιδεύει από μια πόλη A σε μια πόλη B μέσω ενδιάμεσων πόλεων $1, \ldots, n$. Ο (μοναδικός) δρόμος που συνδέει κάθε πόλη i με την επόμενή της $i+1$ έχει μήκος ℓ_i km και όριο ταχύτητας u_i km/h. Ο μοτοσυκλετιστής είναι κατά κανόνα νομοταγής και τηρεί το όριο ταχύτητας. Όμως σήμερα έχει καθυστερήσει, και κινδυνεύει να χάσει την έναρξη μιας πολύ ενδιαφέρουσας παράστασης στην πόλη B. Έτσι κινείται σε κάθε δρόμο με τη μέγιστη επιτρεπτή ταχύτητα. Επιπλέον αποφάσισε, για πρώτη και τελευταία φορά, να υπερβεί το όριο ταχύτητας κατά v km/h και για συνολικό χρόνο T. Χρειάζεται λοιπόν να επιλέξει σε ποια τμήματα $i_1, i_2, \ldots$ της διαδρομής και για πόσο χρόνο στο καθένα θα κινηθεί με ταχύτητα $u_{i_1} + v, u_{i_2} + v, \ldots$ ώστε ο συνολικός χρόνος που παραβιάζει το όριο ταχύτητας να είναι T και ο χρόνος άφιξής του στην πόλη B να ελαχιστοποιηθεί.



Σχήμα 1. Παράδειγμα διαδρομής και ορίων ταχύτητας για την Άσκηση 2.

Για παράδειγμα, στο Σχήμα 1, βλέπουμε μια διαδρομή από την πόλη A στην πόλη B με 4 ενδιάμεσους σταθμούς. Το μήκος των αντίστοιχων τμημάτων της διαδρομής είναι $\ell_0 = 7$km, $\ell_1 = 12$km, $\ell_2 = 14$km, $\ell_3 = 20.5$km, και $\ell_4 = 10$km, και το όριο ταχύτητας είναι $u_0 = 50$km/h, $u_1 = 30$km/h, $u_2 = 70$km/h, $u_3 = 50$km/h, και $u_4 = 100$km/h. Έτσι αν δεν παραβιάσει το όριο ταχύτητας, ο μοτοσυκλετιστής χρειάζεται 1 ώρα και $15'$ (1.25 ώρες) για να φτάσει στην πόλη B. Αν τώρα υπερβεί

το όριο ταχύτητας κατά 20km/h για $24'$ (0.4 ώρες), τότε μπορεί να οδηγήσει με 70km/h στο τμήμα $(A, 1)$, με 50km/h στο τμήμα $(1, 2)$, και με 70km/h στα πρώτα 4.2km του τμήματος $(3, 4)$. Η συνολική διάρκεια της υπέρβασης είναι $24'$, και πλέον ο μοτοσυκλετιστής χρειάζεται περίπου 1 ώρα (1.026 ώρες ακριβώς) για να φτάσει στην πόλη B.

Να διατυπώσετε έναν αποδοτικό αλγόριθμο για τη βέλτιστη κατανομή του χρόνου που ο μοτοσυκλετιστής υπερβαίνει το όριο ταχύτητας. Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας. Τι αλλάζει στο πρόβλημα (και στον αλγόριθμο) αν ο μοτοσυκλετιστής αποφασίσει να υπερβαίνει το εκάστοτε όριο ταχύτητας κατά έναν παράγοντα $\alpha > 1$;

Άσκηση 3: Βότσαλα στη Σκακιέρα (DPV 6.5)

Δίνονται μια σκακιέρα με 4 γραμμές και n στήλες, η οποία έχει ένα χρηματικό ποσό σε κάθε τετράγωνο, και $2n$ βότσαλα, καθένα από τα οποία μπορεί να τοποθετηθεί σε ένα τετράγωνο της σκακιέρας ως αντάλλαγμα για το αντίστοιχο ποσό. Θέλουμε να τοποθετήσουμε κάποια ή όλα τα βότσαλα στη σκακιέρα ώστε να μεγιστοποιήσουμε το ποσό που συγκεντρώνουμε. Δεν επιτρέπεται όμως να τοποθετήσουμε βότσαλα σε τετράγωνα που γειτνιάζουν είτε οριζόντια είτε κατακόρυφα (επιτρέπεται σε τετράγωνα που γειτνιάζουν διαγώνια).

(α) Θεωρούμε τον άπληστο αλγόριθμο ο οποίος ενόσω είναι εφικτό, συλλέγει το μεγαλύτερο ποσό που είναι διαθέσιμο σε τετράγωνο που δεν γειτνιάζει με τετράγωνο όπου έχει ήδη τοποθετηθεί βότσαλο. Να βρείτε ένα παράδειγμα όπου ο άπληστος αλγόριθμος αποτυγχάνει να υπολογίσει τη βέλτιστη λύση. Παρόλα αυτά, ο άπληστος αλγόριθμος εγγυάται τη συγκέντρωση ενός σημαντικού ποσοστού του ποσού που συγκεντρώνει η βέλτιστη λύση. Ποιο είναι αυτό το ποσοστό και γιατί;

(β) Να διατυπώσετε έναν αλγόριθμο δυναμικού προγραμματισμού ο οποίος έχει χρόνο εκτέλεσης $\Theta(n)$ και εγγυάται τη συγκέντρωση του μέγιστου δυνατού ποσού από τη σκακιέρα.

Άσκηση 4: Χωρισμός Κειμένου σε Γραμμές

Έχουμε ένα κείμενο που αποτελείται από n λέξεις μήκους $\ell_1, \ldots, \ell_n$ (το μήκος κάθε λέξης μετριέται σε χαρακτήρες), και θέλουμε να το χωρίσουμε σε γραμμές με ομοιόμορφο τρόπο. Οι γραμμές στοιχίζονται στα αριστερά και μπορούν να έχουν μέχρι C χαρακτήρες η καθεμία. Μεταξύ δύο λέξεων στην ίδια γραμμή πρέπει να υπάρχει (ακριβώς) ένα κενό (χαρακτήρας space), και δεν επιτρέπεται ο διαχωρισμός μιας λέξης σε δύο γραμμές (και βέβαια, η σειρά των λέξεων είναι δεδομένη). Έτσι αν σε μια γραμμή k εμφανίζονται οι λέξεις $i, \ldots, j$, τότε η γραμμή k έχει $s_k = C + 1 - \sum_{p=i}^{j}(\ell_p + 1)$ κενούς χαρακτήρες στα δεξιά.

Να διατυπώσετε έναν αποδοτικό αλγόριθμο που χωρίζει το κείμενο σε γραμμές ώστε να ελαχιστοποιείται το άθροισμα των τετραγώνων του πλήθους των κενών χαρακτήρων που έχουν οι γραμμές στα δεξιά. Δηλαδή, αν το κείμενο χωρίζεται σε m γραμμές με $s_1, \ldots, s_m$ κενά στα δεξιά, το "κόστος" αυτού του χωρισμού είναι $\sum_{k=1}^{m} s_k^2$. Πρέπει λοιπόν ο αλγόριθμος να υπολογίζει έναν διαχωρισμό του κειμένου σε γραμμές ο οποίος ελαχιστοποιεί αυτό το "κόστος". Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας.

Άσκηση 5: Αντίγραφα Αρχείου (KT 6.12)

Έχουμε n διακομιστές με ετικέτες $S_1, \ldots, S_n$, και θέλουμε να βρούμε σε ποιους από αυτούς θα διατηρούμε αντίγραφο ενός αρχείου F. Το εβδομαδιαίο κόστος διατήρησης ενός αντίγραφου του F

στον διακομιστή S_i είναι c_i. Γνωρίζουμε ακόμη ότι σε εβδομαδιαία βάση, ο διακομιστής S_i δέχεται b_i αιτήματα προσπέλασης του F. Αν υπάρχει αντίγραφο του F στον διακομιστή S_i, η εξυπηρέτηση των αιτημάτων έχει αμελητέο κόστος. Διαφορετικά, κάθε τέτοιο αίτημα προωθείται στον επόμενο διακομιστή $S_{i+1}, S_{i+2}, \ldots, S_n$, διαδοχικά, μέχρι να βρεθεί ο πρώτος που έχει αντίγραφο του F. Αν ο πρώτος διακομιστής με αντίγραφο του F είναι ο S_j, $j > i$, το συνολικό κόστος εξυπηρέτησης των b_i αιτημάτων που δέχεται ο S_i είναι $b_i(j-i)$ (σημειώνουμε ότι τα αιτήματα του S_i δεν προωθούνται ποτέ στους διακομιστές $S_{i-1}, S_{i-2}, \ldots, S_1$, με δείκτη μικρότερο του i). Αν δεν υπάρχει διακομιστής S_j, $j > i$, με αντίγραφο του F, το κόστος εξυπηρέτησης των αιτημάτων θεωρείται μη αποδεκτό, π.χ. είναι μεγαλύτερο του c_i.

Να διατυπώσετε έναν αποδοτικό αλγόριθμο που με δεδομένα το παραπάνω πρωτόκολλο αναζήτησης του F στους διακομιστές, το κόστος διατήρησης $c_1, \ldots, c_n$, και το πλήθος $b_1, \ldots, b_n$ των αιτημάτων προσπέλασης στους διακομιστές, υπολογίζει σε ποιους διακομιστές πρέπει να διατηρείται αντίγραφο του αρχείου F ώστε να ελαχιστοποιηθεί το συνολικό κόστος διατήρησης των αντιγράφων και εξυπηρέτησης των αιτημάτων προσπέλασης.

Άσκηση 6 (bonus): Έλεγχος Ταξινόμησης.

Ένας πίνακας ακεραίων $A[1 \ldots n]$ είναι *σχεδόν ταξινομημένος* αν υπάρχουν $n/4$ το πολύ στοιχεία τα οποία αν διαγράψουμε, ο υποπίνακας του A που απομένει είναι ταξινομημένος. Για παράδειγμα, ο πίνακας $[1, 2, 3, 5, 4, 7, 9, 6, 10, 11, 12, 8]$ είναι σχεδόν ταξινομημένος, αφού η διαγραφή των στοιχείων 5, 6, και 8 δίνει τον πίνακα $[1, 2, 3, 4, 7, 9, 10, 11, 12]$, που είναι ταξινομημένος. Για απλότητα, υποθέτουμε στη συνέχεια ότι όλα τα στοιχεία του πίνακα A είναι διαφορετικά.

Στοχεύουμε στη διατύπωση ενός πιθανοτικού αλγόριθμου *λογαριθμικού χρόνου* που θα διακρίνει πίνακες που είναι (πλήρως) ταξινομημένοι από πίνακες που δεν είναι σχεδόν ταξινομημένοι[1]. Συγκεκριμένα, αν ο πίνακας A είναι πλήρως ταξινομημένος, ο αλγόριθμος θα αποφαίνεται πάντα (δηλ. με πιθανότητα 1) ότι ο A είναι σχεδόν ταξινομημένος. Αν ο πίνακας A δεν είναι σχεδόν ταξινομημένος, ο αλγόριθμος θα αποφαίνεται, με πιθανότητα τουλάχιστον 0.9, ότι ο A δεν είναι σχεδόν ταξινομημένος.

(α) Θεωρούμε τον αλγόριθμο που επιλέγει τυχαία (και ανεξάρτητα) k θέσεις $a_1, \ldots, a_k$ του πίνακα A, και αποφαίνεται ότι ο πίνακας A είναι σχεδόν ταξινομημένος αν για κάθε θέση a_i που έχει επιλεγεί, ισχύει ότι $A[a_i - 1] \leq A[a_i] \leq A[a_i + 1]$. Αν υπάρχει έστω μία θέση a_i που δεν ικανοποιεί αυτό το κριτήριο, ο αλγόριθμος αποφαίνεται ότι ο πίνακας A δεν είναι σχεδόν ταξινομημένος. Να δώσετε παράδειγμα πίνακα A με n στοιχεία όπου ο αλγόριθμος χρειάζεται να ελέγξει $k = \Omega(n)$ θέσεις ώστε η πιθανότητα λάθους να γίνει μικρότερη του 0.1.

(β) Υποθέτουμε ότι εφαρμόζουμε την παρακάτω εκδοχή της Δυαδικής Αναζήτησης σε έναν πίνακα A που μπορεί να μην είναι ταξινομημένος (με κίνδυνο φυσικά η αναζήτηση να αποτύχει, αν και το στοιχείο x υπάρχει στον A):

```
BINARY-SEARCH(A, x, low, up)
    if low = up then return low;
    else mid ← ⌈(low + up)/2⌉;
        if x < A[mid] then return BINARY-SEARCH(A, x, low, mid − 1);
        else return BINARY-SEARCH(A, x, mid, up);
```

[1] Εκ πρώτης όψης, αυτό είναι ένας φιλόδοξος στόχος. Δείτε ότι οποιοσδήποτε ντετερμινιστικός αλγόριθμος για αυτό το πρόβλημα πρέπει να έχει τουλάχιστον γραμμικό χρόνο εκτέλεσης, αφού θα πρέπει να διαβάσει τουλάχιστον $3n/4$ από τα στοιχεία του πίνακα A.

΄Εστω λοιπόν ότι για κάποιες τιμές x, y, η BINARY-SEARCH$(A, x, 1, n)$ επιστρέφει τη θέση k και η BINARY-SEARCH$(A, y, 1, n)$ επιστρέφει τη θέση ℓ. Να δείξετε ότι αν $k < \ell$, τότε $x < y$.

(γ) Θεωρούμε τώρα τον αλγόριθμο που επιλέγει τυχαία (και ανεξάρτητα) k θέσεις $a_1, \ldots, a_k$ του πίνακα A, και αποφαίνεται ότι ο πίνακας A είναι σχεδόν ταξινομημένος αν για κάθε θέση a_i που έχει επιλεγεί, ισχύει ότι $a_i =$ BINARY-SEARCH$(A, A[a_i], 1, n)$. Αν υπάρχει έστω μία θέση a_i που δεν ικανοποιεί αυτό το κριτήριο, ο αλγόριθμος αποφαίνεται ότι ο πίνακας A δεν είναι σχεδόν ταξινομημένος. Χρησιμοποιώντας το (β), να δείξετε ότι για κάθε πίνακα A με n θέσεις, αρκεί ο αλγόριθμος να ελέγξει $k = \Theta(1)$ θέσεις, ώστε η πιθανότητα λάθους να γίνει μικρότερη του 0.1.