Αλγόριθμοι και Πολυπλοκότητα
2η Σειρά Γραπτών Ασκήσεων

Γιαννέλος Γιάννης
ΑΜ:03108088

9 Ιανουαρίου 2012

Άσκηση 2: Βιαστικός μοτοσυκλετιστής

Παραβίαση ορίου ταχύτητας κατα $v_i + V$

Έστω t_{sum} ο συνολικός χρόνος που θα κάνει ο μοτοσυκλετιστής για να φτάσει στην πόλη Β. Ταξινομούμε τον πίνακα των αποστάσεων-μεγίστων ταχυτητών σε άυξουσα σειρά σύμφωνα με τις ταχύτητες και έστω τα νέα ταξινομημένα ζεύγη: (l'_i, v'_i) για $i = 1, 2..n$.

Ο βέλτιστος χρόνος άυξησης προκύπτει αν μεγιστοποιήσουμε σε κάθε διάστημα την διαφορά του κανονικού χρόνου (με νόμιμο όριο ταχύτητας) και του χρόνου ξεπερνώντας το εκάστοτε όριο ταχύτητας. Αρα:

$$\Delta T = \frac{l_i}{v_i} - \frac{l_i}{v_i+V} = V \cdot \frac{l_i}{v_i} \cdot \frac{1}{v_i+V}$$

Όμως το $\frac{l_i}{v_i}$ ανεξάρτητο του V και το V παραμένει σταθερό. Άρα μεγιστοποιούμε το ΔT αν διαλέξουμε πρώτα και εξαντλήσουμε χρονικά τα διαστήματα με μικρότερο όριο ταχύτητας. Επομένως ένας αλγόριθμος που θα έλυνε αυτό το προβλήμα θα είχε τα εξής βήματα:

1. $i = 0$
2. Όσο είναι εφικτό εξάντλησε χρονικά το διάστημα με ταχύτητα v'_i
3. Τότε $T \leftarrow T - \frac{l'_i}{v'_i+V}$
4. $i \leftarrow i + 1$
5. Αν $T > 0$ πήγαινε στο βήμα 2

Παραβίαση ορίου ταχύτητας κατα $\alpha \cdot v_i$

Στην περίπτωση που η παραβίαση του ορίου ταχύτητας γίνεται κατά $\alpha \cdot v_i$, σύμφωνα με τα παραπάνω το ΔT θα είναι: $\Delta T = \frac{l_i}{v_i} - \frac{l_i}{\alpha \cdot v_i} = \frac{l_i}{v_i} \cdot \frac{\alpha-1}{\alpha}$. Επομένως ο αλγόριθμός μας θα πρέπει να διαλέγει πρώτα, μέχρι να τελειώσει ο χρόνος που παραβιάζεται το όριο ταχύτητας, τα διαστήματα με μεγαλύτερο $T_i = \frac{l_i}{v_i}$.

Άσκηση 3: Βότσαλα στη σκακιέρα

(α)

Ο άπληστος αλγόριθμος που περιγράφει η εκφώνηση αν και δεν λειτουργεί πάντα σωστά, μπορεί να μας εξασφαλίσει ένα συγκεκριμένο ποσοστό του ποσού της βέλτιστης λύσης. Ένα παράδειγμα όπου ο αλγόριθμος μας δίνει λάθος αποτέλεσμα είναι το εξής: Έστω ότι υπάρχουν 2 τετράγωνα που συνορεύουν με ένα

άλλο (για παράδειγμα απο δεξιά και απο πάνω) και έχουν άθροισμα κέρδους μεγαλύτερο απο το ποσό του τρίτου τετραγώνου, αλλά το καθένα ξεχωριστά περιέχει λιγότερα χρήματα. Ακόμα στο παράδειγμα έστω ότι η επιλογή των 2 τετραγώνων είναι εφικτή στην δεδομένη κίνηση και αποτελούν τα τετράγωνα με τα ακριβώς λιγότερα χρήματα απο το 3ο. Ο αλγόριθμος θα διαλέξει πρώτο το επιτρεπτό τετράγωνο με τα περισσότερα λεφτά και μετα δεν θα είναι εφικτό να διαλέξει τα άλλα 2, που σε άλλη περίπτωση θα έδιναν μεγαλύτερο κέρδος. Έτσι εγγυάται να μας βρεί τη βέλτιστη λύση κατα 25%.

(β)

Μια βέλτιστη λύση του προβήματος θα μπορούσε να βρεθεί με την χρήση δυναμικού προγραμματισμού. Έστω i_1, i_2, i_3, i_4 οι γραμμές της σκακιέρας και $A[i, j]$ πίνακας με τα χρηματικά ποσά της σκακιέρας. Σε κάθε στήλη έχουμε 8 διαφορετικές δυνατές επιλογές τετραγώνων τις:

$$I = \{(i_1, i_3), (i_2, i_4), (i_1, i_4), (i_1), (i_2), (i_3), (i_4), (\varnothing)\}$$

Για την εφαρμογή του δυναμικού προγραμματισμού θα χρησιμοποιήσουμε έναν πίνακα $Dyn[8][n]$ με διαστάσεις $8 \times N$ (κάθε γραμμή εκφράζει μια απο τις δυνατές επιλογές και κάθε στήλη τις στήλες της σκακιέρας). Κάθε θέση του πίνακα $Dyn[i, j]$ θα δείχνει το χρηματικό ποσό αν διαλέξουμε τον i-οστό συνδιασμό γραμμών συν το μέγιστο συνδιασμό για τις προηγούμενες δυνατές στήλες. Άρα:

$$Dyn[i, j] = \sum_{k \in I} A[k, j] + max(Dyn[l, j - 1])$$

- όπου k οι γραμμές των τετραγώνων του i-οστού συνδιασμού
- l οι εφικτοί συνδυασμοί για την προηγούμενη στήλη δεδομένης της επιλογής στην j στήλη

Για να συμπληρώσουμε τον πίνακα χρειάζομαστε $\Theta(64n)$ αφού θέλουμε $\Theta(8n)$ για την προσπέλαση κάθε στοιχείου, και για κάθε στοιχείο του πίνακα θέλουμε $\Theta(8)$ για να βρούμε το μέγιστο για την προηγούμενη στήλη (πιθανά όχι όλα τα στοιχεία της, αλλά μόνο αυτά που επιτρέπεται).Ακόμα μπορούμε να κρατάμε κατά την συμπλήρωση του πίνακα Dyn το $sol = max$ που υπολογίζει τη λύση του προβλήματος.

Άσκηση 4: Χωρισμός κειμένου σε γραμμές

Όπως ορίζεται και στην εκφώνηση, το κείμενο που θέλουμε να χωρίσουμε αποτελείται απο n λέξεις μήκους $l_1, l_2, ..., l_n$ χαρακτήρες η κάθε μια. Οι γραμμές θα

πρέπει να έχουν μέγεθος το πολύ C χαρακτήρες και κάθε γραμμή στοιχίζεται στα αριστερά. Ακόμα κάθε λέξη χωρίζεται απο την επόμενη με τον κενό χαρακτήρα χωρίς να επιτρέπεται να χωριστεί σε 2 γραμμές και δεν επιτρέπεται η αναδιάταξη τους. Άρα συνολικά, αν στη γραμμή k εμφανίζονται οι λέξεις $i, ..., j$ τότε η γραμμή έχει αριθμό κενών χαρακτήρων στα δεξιά που δίνεται απο τον τύπο $s_k = C+1-\sum_{p=1}^{j}(l_p+1)$. Το ζητούμενο της άσκησης είναι να βρούμε αλγόριθμο που να χωρίζει κατάλληλα το κείμενο έτσι ώστε να ελαχιστοποιείται το άθροισμα των τετραγώνων του πλήθους των κενών χαρακτήρων που έχουν οι γραμμές στα δεξιά, δηλαδή το $\sum_{k=1}^{m} s_k^2$. Για την επίλυση θα εργαστούμε με την μέθοδο σχεδίασης bottom-up χρησιμοποιώντας δυναμικό προγραμματισμό:

- Η κύρια ιδέα στηρίζεται στο ότι αν μπορούμε να υπολογίσουμε το ελάχιστο $\sum_{k=1}^{\lambda} s_k^2$ για τις πρώτες λ σειρές, τότε αν προσθέσουμε τις λέξεις για την $\lambda+1$ σειρά, δεν αλλάζει κάτι στην στοίχιση των προηγούμενων σειρών παρα μόνο της τελευταίας.

- Υπολογίζουμε τον πίνακα S όπου για κάθε i, j το στοιχείο $S[i, j]$ δείχνει το $s_k = C + 1 - \sum_{p=1}^{j}(l_p + 1)$.

- Με την παραπάνω ιδέα είναι εφικτό να φτιάξουμε μια αναδρομική σχέση για το "κόστος" της διάταξης των πρώτων i λέξεων του κειμένου . Η σχέση αυτή θα είναι:

$$C[i] = \begin{cases} 0, & i = 0 \\ \min\limits_{1 \le k \le i} C[k-1] + S[k, i], & i > 0 \end{cases}$$

Η πολυπλοκότητα του αλγορίθμου για τον υπολογισμό του πίνακα S είναι $O(n \cdot C)$ όπου n το πλήθος των λέξεων και C το μέγιστο πλήθος χαρακτήρων ανά γραμμή. Το ίδιο ισχύει και για τον υπολογισμό του $C[n]$.

Άσκηση 5: Αντίγραφα αρχείου

Για την επιλογή των διακομιστών που θα φυλάσσεται το αντίγραφο του αρχείου F θα χρησιμοποιήσουμε τον παρακάτω αλγόριθμο. Έστω $M(j)$ συνάρτηση που μας δίνει το ελάχιστο κόστος αν τοποθετήσουμε το αρχείου στον server j. Χρειάζεται να υπολογίσουμε την καλύτερη θέση για να τοποθετήσουμε το αντίγραφο του αρχείου F πριν τον j, έστω αυτή i $(i < j)$. Τότε για όλους του servers μέχρι τον i το κόστος θα είναι $M(i)$ και το κόστος για τους servers i μέχρι $j-1$ θα είναι $b_i \cdot (0 + 1 + 2 + ... + (j - 1 - 1)) = b_i \cdot \binom{j-1}{2}$. Ακόμα θα πρέπει να υπολογίσουμε μαζί με αυτά και το εβδομαδιαίο κόστος c για την αποθήκευση του αρχείου στον server. Άρα συνολικά:

$$M(j) = c_j + min_{0 \leq i < j}(M(i) + b_i \cdot \binom{j-1}{2})$$

Αρά αν $M(0) = 0$ μας αρκεί να υπολογίσουμε το $M(n)$ και για να βρούμε ποιοί συγκεκριμένα θα είναι οι διακομιστές αρκεί να βρούμε τα διαδοχίκά $M(i)$.