ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Αλγόριθμοι και Πολυπλοκότητα
2^{η} Σειρά Γραπτών Ασκήσεων
Ακ. έτος 2010-2011

Λύρας Γρηγόρης
Α.Μ.: 03109687

27 Δεκεμβρίου 2011

1 Επιτροπή Αντιπροσώπων

Κάνουμε ταξινόμηση στον πίνακα των $[s_i, f_i)$ ως προς f_i. Ξεκινώντας από το πρώτο ($n = 0$), ελέγχω ποια από τα επόμενα στον ταξινομημένο πίνακα επικαλύπτονται με αυτό μέχρι να βρω κάποιο με το οποίο δεν έχει επικάλυψη έστω $[s_k, f_k)$. Κρατάω το τελευταίο ($[s_{k-1}, f_{k-1})$) από τα προηγούμενα. Στη συνέχεια, αν αυτό έχει επικάλυψη με το $[s_k, f_k)$ συνεχίζω για $n = k + 1$. Αλλιώς συνεχίζω τον ίδιο αλγόριθμο για $n = k$.

2 Βιαστικός Μοτοσικλετιστής

Ταξινομούμε τον πίνακα των ταχυτήτων-αποστάσεων με βάση τις ταχύτητες. Ξεκινώντας από τη μικρότερη "τρέχουμε" στο αντίστοιχο διάστημα με τη μέγιστη ταχύτητα ξεπερνώντας και το όριο κατά u. Αυτό μας παίρνει χρόνο $T_i = \frac{l_i}{u_i+u}$. Αφαιρούμε το χρόνο αυτό από τον συνολικό χρόνο που έχουμε και συνεχίζουμε (προφανώς αν ξεπεράσουμε το χρονικό όριο η απόσταση που θα διανύσουμε μέσα σε αυτό το τμήμα θα είναι $x = l_i * \frac{T}{T_i}$), για την επόμενη σε σειρά ταχύτητα. Με τον ίδιο τρόπο. Μόλις ο χρόνος που έχουμε εξαντληθεί, σταματάμε. Τα τμήματα που έχουμε επιλέξει για να υπερβούμε το όριο ταχύτητας θα ελαχιστοποιήσουν το χρόνο άφιξης στο B.

Έστω T_s ο συνολικός χρόνος από το A στο B. Συνεπώς έχουμε τη σχέση:

$$T_s = \sum_{i=1}^{n} \frac{l_i}{u_i}$$

Θέλουμε να ελαχιστοποιήσουμε το άθροισμα αυτό από θετικούς όρους (η ταχύτητα δεν έχει νόημα να είναι προσημασμένη σε αυτό το πρόβλημα), συνεπώς θέλουμε να ελαχιστοποιήσουμε τους όρους $k_1, k_2, k_3 ....$

Έστω ο όρος k:

$$T_k = \begin{cases} \dfrac{l_i}{u_i} \\ \dfrac{l_i}{u_i + u} \end{cases}$$

Ορίζω ΔT τη διαφορά:

$$\Delta T = \frac{l_i}{u_i} - \frac{l_i}{u_i + u} = l_i * \frac{u}{u_i * (u_i + u)}$$

$$\frac{\partial \Delta T}{\partial u_i} = l_i * u * (-\frac{2 * u_i + u}{(u_i * (u_i + u))^2}) < 0$$

Συνεπώς αυξάνοντας το u_i μειώνεται το ΔT, το κέρδος δηλαδή που θα είχαμε ξεπερνώντας το όριο ταχύτητας. Αυτό συνεπάγεται πως πρώτα θέλουμε να πάρουμε τα κομμάτια με τη μικρότερη ταχύτητα καθώς αυτά μας δίνουν το μέγιστο κέρδος. Το πρόβλημα λοιπόν ανάγεται σε συνεχές Knapsack όπου το Greedy κριτήριο είναι η "μικρότερη ταχύτητα πρώτη".

Σε περίπτωση που αποφασίσουμε να υπερβούμε την ταχύτητα κατά έναν παράγοντά $a > 1$ τότε η ΔT έχει τη μορφή:

$$\Delta T = \frac{l_i}{u_i} - \frac{l_i}{a * u_i} = \frac{a-1}{a} * \frac{l_i}{u_i} = \frac{a-1}{a} * T_i$$

Συνεπώς τώρα το κριτήριο αλλάζει και πλέον θέλουμε πρώτα τα κομμάτια που παίρνουν περισσότερο χρόνο για να τα διανύσουμε. Κατά τα άλλα ο αλγόριθμος παραμένει ίδιος.

3 Βότσαλα στη Σκακιέρα

α′

Εγγυάται να μας βρει τη βέλτιστη λύση κατά 25%.

β′

Για να λύσουμε το πρόβλημα χρησιμοποιούμε δυναμικό προγραμματισμό. Θεωρώ τις 4 σειρές τετραγώνων $X[0..n-1], Y[0..n-1], Z[0..n-1], W[0..n-1]$, και έναν πίνακα $n \times 8$ ($dTable[n][8]$). Στον παρακάτω πίνακα βλέπουμε όλους τους δυνατούς συνδυασμούς που μπορούμε να έχουμε παίρνοντας το στοιχείο i από κάθε γραμμή χρησιμοποιώντας τον περιορισμό του προβλήματος. Αν σε κάθε βήμα κρατάμε και από ποια τιμή προέρχεται το τρέχον άθροισμα, μετά από n επαναλήψεις θα έχουμε οκτώ στοιχεία το μεγαλύτερο από τα οποία είναι η απάντηση του προβλήματος. Για να ανακατασκευάσουμε το άθροισμα και να βρούμε ποια τετράγωνα χρησιμοποιήσαμε τελικά μπορούμε να ακολουθήσουμε τους προγόνους κάθε μερικού αθροίσματος ξεκινώντας από το αποτέλεσμα μέχρι να φτάσουμε στην αρχή του πίνακα. Η πολυπλοκότητα του αλγορίθμου είναι $\Theta(65 * n) = \Theta(n)$.

Ορίζω την αντικατάσταση:

$$dTable[i] : [k_1], [k_2], .., [k] = dTable[i][k_1], dTable[i][k_2], .., dTable[i][k]$$

choices	iteration 1	iteration i	iteration $i+1$		
$X+Z$	$X[0]+Z[0]$	$dTable[i][0]$	$X[i+1]+Z[i+1]$	$+$	$max\{dTable[i] : [1], [4], [6], [7]\}$
$Y+W$	$Y[0]+W[0]$	$dTable[i][1]$	$Y[i+1]+W[i+1]$	$+$	$max\{dTable[i] : [0], [3], [5], [7]\}$
$X+W$	$X[0]+W[0]$	$dTable[i][2]$	$X[i+1]+W[i+1]$	$+$	$max\{dTable[i] : [4], [5], [7]\}$
X	$X[0]$	$dTable[i][3]$	$X[i+1]$	$+$	$max\{dTable[i] : [1], [4], [5], [6], [7]\}$
Y	$Y[0]$	$dTable[i][4]$	$Y[i+1]$	$+$	$max\{dTable[i] : [0], [2], [3], [5], [6], [7]\}$
Z	$Z[0]$	$dTable[i][5]$	$Z[i+1]$	$+$	$max\{dTable[i] : [1], [2], [3], [4], [6], [7]\}$
W	$W[0]$	$dTable[i][6]$	$W[i+1]$	$+$	$max\{dTable[i] : [0], [3], [4], [5], [7]\}$
0	0	$dTable[i][7]$	0	$+$	$max\{dTable[i] : [0], [1], [2], [3], [4], [5], [6], [7]\}$

Πίνακας 1: Πίνακας αναδρομικών σχέσεων.

4 Χωρισμός Κειμένου σε Γραμμές

$$\sum_{k=1}^{m} s_k^2 = m * (c+1)^2 - 2 * (c+1) * \sum_{k=1}^{m} S_{cw_k} + \sum_{k=1}^{m} {S_{cw_k}}^2 \tag{1}$$

$$\frac{\partial S_k^2}{\partial S_{cw_k}} = -2 * (c+1) + 2 * S_{cw_k} \tag{2}$$

$$S_{cw_k} = c + 1 - S_k \tag{3}$$

Επιθυμούμε να ελαχιστοποιήσουμε το κόστος (1), εφόσον $S_{cw_k} \leq c+1$ η σχέση (2) μας δίνει $\frac{\partial S_k^2}{\partial S_{cw_k}} \leq 0$, συνεπώς πρέπει να μεγιστοποιήσουμε το άθροισμα των γραμμάτων ανά γραμμή (3) για όλες τις γραμμές.

Για την επίλυση αυτού του προβλήματος θα πρέπει να το μετασχηματίσουμε στο πρόβλημα απόφασης "μπορώ να βρω κατανομή τέτοια ώστε να μπορώ να χωρέσω σε κάθε γραμμή το πολύ d χαρακτήρες;" το οποίο είναι επιλύσιμο σε γραμμικό χρόνο και να κάνουμε δυαδική αναζήτηση στο διάστημα $[max\{l_i : i = 1, .., n\}, c+1]$.

Από όλες τις πιθανές λύσεις θέλουμε την ελάχιστη που είναι και η βέλτιστη.

5 Αντίγραφα Αρχείου

Αναζητούμε σε πρώτο στάδιο τον πρώτο εξυπηρετητή από το τέλος που έχει μη μηδενικό b_j. Αυτός είναι ο τελευταίος εξυπηρετητής που θα έχει το αρχείο που θέλουμε. Με αρχή τον εξυπηρετητή j κοιτάμε τους προηγουμένους του. Όσο $c_i \geq (i-j) * b_i$ συνεχίζουμε στο c_{i-1}. Όταν δεν ισχύει η συνθήκη έχουμε δύο επιλογές. Είτε να έχουμε το αρχείο στον εξυπηρετητή i είτε στον $i+1$. Αυτό το διαλέγουμε με μια ακόμα σύγκριση. Αν $c_i + distance_{i+1} * b_{i+1} \leq c_{i+1} + b_i$ επιλέγουμε να τοποθετήσουμε το αρχείο στον εξυπηρετητή S_i, αλλιώς επιλέγουμε τον S_{i+1}. Συνεχίζουμε τον αλγόριθμο για ακόμα μικρότερα i μέχρι να φτάσουμε στην αρχή.

6 Έλεγχος Ταξινόμησης

7 Bonus: Δρομολόγηση Εργασιών