Online Διαγωνισμοί HelleniCO

Οκτώβριος 2012

Εγχειρίδιο Λύσεων

Περιεχόμενα

Μπορείτε να συζητήσετε τις απορίες σας πάνω στις λύσεις και τα θέματα στη διεύθυνση
`http://www.pdpforum.eu.org/forum/viewtopic.php?f=3&t=2257`

1 Δρόμος προς το σχολείο (trlights)

Δημιουργός: Γιάννης Χατζημίχος

Εκφώνηση: `http://www.hellenico.gr/contest/task.php?name=trlights`

Αρχικά αρκεί να διαβάσουμε το input και να αποθηκεύσουμε σε μία μεταβλητή, έστω G, το πλήθος των πράσινων φαναριών. Έτσι, για τις δύο πρώτες περιπτώσεις αρκεί να δούμε αν η τιμή του G είναι ίση με 10 ή μικρότερη ή ίση του 5. Όσων αφορά την περίπτωση όπου G είναι άρτιος, υπάρχουν πολλοί τρόποι να το ελέγξουμε:

- Αν το G είναι ίσο με 0 ή ίσο με 2 ή ίσο με 4
- Αν το υπόλοιπο διαίρεσης με το 2 είναι ίσο με 0 (G $\bmod 2 == 0$)
- Αν το τελευταίο ψηφίο της δυαδικής αναπαράστασης του G είναι ίσο με 0 ($G \wedge 1 == 0$).

Από τους τρεις τρόπους, ο τελευταίος είναι ο υπολογιστικά φθηνότερος.

2 Πάσο (paso)

Δημιουργός: Βαγγέλης Κηπουρίδης

Εκφώνηση: `http://www.hellenico.gr/contest/task.php?name=paso`

Έστω x,y οι συντεταγμένες του πάνω αριστερά τετραγώνου ενός υποπίνακα. Μπορούμε να υπολογίσουμε το πλήθος των 0 και των 1 αυτού του υποπίνακα πολύ απλά με τον κώδικα:

```
1 count[0] = count[1] = 0;
2 for ( i=x; i<x+3; ++i ) {
3   for (j=y; j<y+3; ++j ) {
4     ++count[ A[i][j] ];
5   }
6 }
```

Επομένως το πρόβλημα πλέον είναι η εύρεση όλων των πάνω αριστερά τετραγώνων των 3*3 υποπινάκων. Παρατηρούμε ότι εμφανίζονται στις θέσεις 1-1, 1-4, 1-7... 4-1, 4-4, 4-7... Στον παρακάτω κώδικα το i και το j αυξάνονται σε κάθε επανάληψη κατά 3:

```
1 for ( i=1; i<=3*N; i += 3 ) {
2   for ( j=1; j<=3*N; j += 3 ) {
3     //Use the previous code
4   }
5 }
```

3 Επιχείρηση ενυδρείο (aquarium)

Δημιουργός: Κυριάκος Αξιώτης

Εκφώνηση: `http://www.hellenico.gr/contest/task.php?name=aquarium`

Αρχικά παρατηρούμε ότι η συνολική δυσκολία των ενυδρείων είναι αντιστρόφως ανάλογη με τον όγκο του νερού που έχουμε βάλει στα ενυδρεία, δηλαδή στην ουσία, όσο πιο πολύ νερό βάζουμε τόσο πιο πολύ μειώνεται η δυσκολία και συνεπώς τόσο πιο εύκολα μπορούν τα δελφίνια να δραπετεύσουν. Άρα θα πρέπει να βρούμε την τιμή του όγκου για την οποία η συνολική δυσκολία είναι όσο πιο κοντά γίνεται στο K (συγκεκριμένα θα πρέπει να είναι ακριβώς K).

Μπορούμε να χρησιμοποιήσουμε **δυαδική αναζήτηση**[1] στο ύψος του νερού (στο ψηλότερο ενυδρείο) ξεκινώντας από το διάστημα $[0, max\_height]$ (όπου max_height το ύψος του ψηλότερου ενυδρείου) και κάθε φορά να ελέγχουμε αν είναι εφικτή η απόδραση.

Αν είναι, τότε μπορούμε να περιορίσουμε το διάστημα στο πρώτο μισό, ενώ αν όχι στο δεύτερο μισό. Αφού βρούμε το βέλτιστο ύψος m μπορούμε με ένα απλό πέρασμα στα ενυδρεία να υπολογίσουμε τον όγκο του νερού που σπαταλήθηκε από τον τύπο

$$V = \sum_i L \cdot W \cdot (H(i) - m) = L \cdot W \cdot \sum_i H(i) – L \cdot W \cdot N \cdot m$$

Η πολυπλοκότητα είναι $O(N \log(max\_height))$.

[1] Ενα ενδιαφέρον εκπαιδευτικό άρθρο πάνω στη δυαδική αναζήτηση βρίσκεται στην σελίδα `http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=binarySearch`.

4 Συλλογή Επαναστατών (rebels)

Δημιουργός: Βασίλης Νάκος

Εκφώνηση: `http://www.hellenico.gr/contest/task.php?name=rebels`

Βασιζόμαστε στις παρακάτω παρατηρήσεις:

Θεωρούμε τα φορτηγάκια f_i σε αύξουσα σειρά ($f_1 \leq f_2 \leq ... \leq f_k$). Όμοια για τους επαναστάτες ($r_1 \leq r_2 \leq ... \leq r_n$).

***Παρατήρηση* 1.** *Υπάρχει μια βέλτιστη λύση όπου όλα τα φορτηγάκια πηγαίνουν στην τελική τους θέση και μετά έρχονται όλοι οι επαναστάτες πάνω τους.*

Απόδειξη. Πράγματι, έστω ότι υπάρχει ένα φορτηγάκι για το οποίο δεν ισχύει αυτό και υπάρχει τουλάχιστον ένας επαναστάτης που μπαίνει μέσα του πριν αυτό φτάσει στην τελική του θέση. Τότε, έστω το σημείο που ήταν πριν κουνηθεί στην τελική του θέση. Είναι εύκολο να δεις ότι όλους τους επαναστάτες που φορτώθηκαν πάνω του ενώ ήταν σε αυτό το σημείο, μπορείς να τους πας στο τελικό σημείο χωρίς να αυξηθεί το κόστος και το φορτηγάκι να μην το σταματήσεις καν σε αυτό το σημείο. Επαγωγικά για το πλήθος των σημείων που σταματάει κάθε φορτηγάκι και για όλα τα φορτηγάκια, ισχύει το ζητούμενο. □

***Παρατήρηση* 2.** *Αν οι επαναστάτες r_{i-1} και r_{i+1} μπούνε στο φορτηγάκι j στη βέλτιστη λύση, τότε στο ίδιο θα μπει και ο επαναστάτης r_i.*

Απόδειξη. Πράγματι, αν $f_1^*, f_2^*, ..., f_k^*$ (εύκολα $f_1^* \leq f_2^* \leq ... \leq f_k^*$) οι τελικές θέσεις των φορτηγών, τότε με περιπτώσεις για τη θέση του f_j^* (στο διάστημα $(-\infty, r_{i-1}]$, στο διάστημα $[r_{i-1}, r_i]$, στο διάστημα $[r_i, r_{i+1}]$ και στο $[r_{i+1}, +\infty)$) βρίσκουμε ότι δεν υπάρχει άλλο φορτηγάκι l του οποίου η τελική θέση να είναι πιο κοντά στον i επαναστάτη. □

Αυτό μας λέει ότι σε μία βέλτιστη λύση, η ανάθεση των επαναστατών σε φορτηγάκια θα είναι κάπως έτσι : Από r_1 ως r_{i_1} στο πρώτο φορτηγάκι, από r_{i_1+1} ως r_{i_2} στο δεύτερο φορτηγάκι, για κάποιους δείκτες $i_1 < i_2$, και ούτω καθεξής.

***Παρατήρηση* 3.** *Αν σε μία βέλτιστη λύση οι επαναστάτες r_i ως r_l ($i < l$) μπούνε στο φορτηγάκι f_j, τότε το κόστος για να μπουν όλοι αυτοί οι επαναστάτες στο εν λόγω φορτηγάκι, ισούται με το άθροισμα των αποστάσεων των r_i ως r_l αλλά και του f_j από τη διάμεσό τους.*

Απόδειξη. Στην πραγματικότητα πρέπει να διαλέξουμε ένα σημείο f_j^* ώστε να ελαχιστοποιείται το $|f_j^* - f_j| + (|r_i - f_j^*| + ... + |r_l - f_j^*|)$. Θα αποδείξουμε ότι το συγκεκριμένο σημείο είναι η **διάμεσος** του συνόλου $F = \{f_j\} \cup \{r_i, ..r_l\}$, και όπου δηλαδή πρέπει να κινηθεί το φορτηγάκι. Πράγματι, αν θεωρήσουμε τα σημεία του συνόλου F τα ταξινομημένα σε αύξουσα σειρά ως $x_1, x_2, .., x_{l-i+2}$ βλέπουμε ότι αν δεν είναι το σημείο ελαχιστοποίησης η διάμεσος, τότε αριστερά(αντίστοιχα δεξιά) θα υπάρχουν λιγότερα από τα μισά σημεία και δεξιά(αντίστοιχα αριστερά) περισσότερα από τα μισά, άρα κουνώντας στο σημείο αμέσως δεξιότερα(αντίστοιχα αριστερότερα) το συνολικό άθροισμα θα μειωθεί κατά τουλάχιστον την διανυόμενη απόσταση, άτοπο. □

Τώρα είμαστε έτοιμοι να εφαρμόσουμε **δυναμικό προγραμματισμό**. Ορίζουμε ως $dp[i][j]$, τη βέλτιστη λύση για να εξυπηρετήσουμε τους i πρώτους επαναστάτες με τα j πρώτα φορτηγά. Προφανώς, $dp[0][0] = 0$ και $dp[i][0] = +\infty$ για $i > 0$. Για να υπολογίσουμε το $dp[i][j]$ θα κοιτάξουμε ποιοι επαναστάτες θα μπουν στο φορτηγάκι j, άρα κοιτάμε τα διαστήματα $[l, j]$ για όλα τα πιθανά $1 \leq l \leq j$. Για σταθερό l, η βέλτιστη λύση θα είναι $dp[l-1][j-1] + cost(f_j, \{r_l, .., r_i\})$, όπου $cost(f_j, \{r_l, .., r_i\})$ είναι το άθροισμα των αποστάσεων των σημείων από τη διάμεσο όπως προαναφέρθηκε. Υπολογίζοντας το παραπάνω χρειάζεται $O(l)$ χρόνο και για l σημεία ο συνολικός χρόνος φτάνει στο $O(n^3 \cdot k)$. Καθώς το l αλλάζει όμως, μπορούμε με έξυπνο τρόπο (πώς;) να υπολογίσουμε καθένα από τα παραπάνω κόστη σε $O(1)$ από το προηγούμενο, ρίχνοντας έτσι το χρόνο στο $O(n^2 \cdot k)$.