ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εαρινό 2009

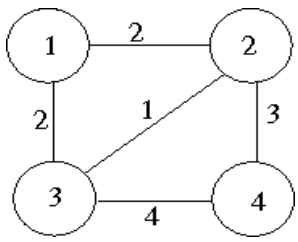
ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 18/6/2009, 12:30

Overbooked πτήσεις (0.25 + 0.25 = 0.5 βαθμοί)

Μια αεροπορική εταιρία έχει N (3 ≤ N ≤ 100,000) προορισμούς και M (2 ≤ M ≤ 200,000) πτήσεις μεταξύ τους. Οι πτήσεις της εταιρίας είναι τέτοιες ώστε όταν υπάρχει κάποια πτήση από τον προορισμό A σε κάποιον προορισμό B τότε υπάρχει και αντίστοιχη πτήση από τον προορισμό B στον A. Επίσης, μπορείτε να θεωρήσετε ότι αυτές οι δύο πτήσεις έχουν την ίδια χρονική διάρκεια. Κατά συνέπεια, ο γράφος των πτήσεων μπορεί να θεωρηθεί ως μη κατευθυνόμενος και φυσικά η ύπαρξη ακμών (δηλ. πτήσεων) από έναν κόμβο (δηλ. προορισμό) στον ίδιο κόμβο του γράφου δεν έχει κάποιο νόημα. Οι πελάτες της εταιρίας αγοράζουν εισιτήρια για να ταξιδέψουν από έναν αρχικό προορισμό X_1 σε κάποιον άλλο προορισμό X_n πιθανώς με κάποιες ενδιάμεσες στάσεις X_2 ... X_{n-1} κατά τη διαδρομή. Η εταιρία, για να ικανοποιήσει τους πελάτες της, τους δίνει πάντα εισιτήρια τέτοια ώστε ο συνολικός χρόνος που χρειάζεται για να ταξιδέψει κάποιος από τον προορισμό X_1 σε οποιονδήποτε άλλον προορισμό X_n να είναι ο ελάχιστος. Δυστυχώς, πολλές φορές συμβαίνει η τελευταία πτήση (X_{n-1} X_n) του κάθε προορισμού να είναι overbooked. Στις περιπτώσεις αυτές η εταιρία διατηρεί το δικαίωμα να αλλάζει (κατά την αναχώρηση) τις πτήσεις των πελατών της με τελικό προορισμό X_n σε κάποιο άλλο δρομολόγιο, αλλά για να μη τους δυσαρεστήσει πολύ τους αλλάζει πάντα το δρομολόγιο σε αυτό με την αμέσως επόμενη ελάχιστη συνολική διάρκεια ταξιδιού. Κατά πάσα πιθανότητα, το παρακάτω παράδειγμα θα σας διαφωτίσει.



Από - Προς	Καλύτερη Διαδρομή	Χρόνος	Τελευταία Πτήση
1 προς 2	1 → 2	2	1 → 2
1 προς 3	1 → 3	2	1 → 3
1 προς 4	1 → 2 → 4	5	2 → 4

Από - Προς	Καλύτερη Διαδρομή	Χρόνος	Overbooked Πτήση
1 προς 2	1 → 3 → 2	3	1 → 2
1 προς 3	1 → 2 → 3	3	1 → 3
1 προς 4	1 → 3 → 4	6	2 → 4

Η άσκηση ζητάει να γράψετε δύο προγράμματα (ένα σε ML και ένα σε Java) που να δέχονται ως είσοδο το γράφο με τις πτήσεις της εταιρίας και να επιστρέφουν ως έξοδο την ελάχιστη διάρκεια των διαδρομών από τον προορισμό 1 σε κάθε άλλο προορισμό όταν η τελευταία πτήση του ταξιδιού με την ελάχιστη διάρκεια χωρίς κανέναν περιορισμό είναι overbooked και κατά συνέπεια δε μπορεί να χρησιμοποιηθεί. Δηλαδή το πρόγραμμά σας πρέπει να επιστρέφει τις τιμές της στήλης “χρόνος” στο δεύτερο πίνακα παραπάνω. Παρακάτω, δείχνουμε από ένα παράδειγμα από τη χρήση του προγράμματός σας σε ML (δώστε στην αρχική σας συνάρτηση στο ML πρόγραμμά σας το παρακάτω όνομα) και σε Java. Οι δύο πρώτες παράμετροι στην ML και οι δύο πρώτοι αριθμοί του αρχείου εισόδου στη Java είναι οι N και M παραπάνω.

ML

```
- overbooked 4 5 [(1,2,2), (1,3,2), (3,4,4), (3,2,1), (2,4,3)];
val it = [3,3,6] : int list
```

Java

```
> java Overbooked.class
```

είσοδος

```
4 5
1 2 2
1 3 2
3 4 4
3 2 1
2 4 3
```

έξοδος

```
3
3
6
```

Μπορείτε να υποθέσετε ότι υπάρχουν πάντα τρόποι για ταξίδια από τον κόμβο 1 σε όλους τους άλλους κόμβους του γράφου τόσο ελάχιστου χρόνου όσο και (δεύτερου) ελάχιστου χρόνου όταν η τελευταία πτήση του ταξιδιού με τον ελάχιστο χρόνο είναι overbooked. Επίσης, μπορείτε να υποθέσετε ότι ο αρχικός γράφος θα είναι τέτοιος ώστε για όλα τα ταξίδια ελάχιστου χρόνου από τον κόμβο 1 στους άλλους προορισμούς η ακμή τελευταίας πτήσης (για κάθε ταξίδι) θα είναι μοναδική.

Ρίξε μια ζαριά καλή... (0.25 + 0.25 = 0.5 βαθμοί)

Όπως φυσικά ξέρετε τα περισσότερα παιχνίδια στρατηγικής είναι ισοδύναμα με αναζήτηση κάποιου κόμβου “στόχου” σε έναν κατευθυνόμενο γράφο του οποίου οι ακμές αντιστοιχούν στις επιτρεπτές κινήσεις από κόμβο σε κόμβο. Στα παιχνίδια στα οποία η τύχη δεν παίζει κάποιο ρόλο, θεωρητικά ο παίκτης μπορεί να σχεδιάσει από πριν κάποια βέλτιστη στρατηγική των “κινήσεών” του η οποία να τον οδηγεί στο επιθυμητό αποτέλεσμα. Όταν όμως υπεισέρχεται κάποιο στοιχείο τύχης στο παιχνίδι, όπως για παράδειγμα αν το πόσες κινήσεις θα κάνουμε κάθε φορά καθορίζεται από κάποιο ζάρι (π.χ. αν το ζάρι δείξει 2 θα πρέπει να μετακινηθούμε κατά δύο ακμές στο γράφο) η οποιαδήποτε στρατηγική μας μπορεί να χαλάσει. Τι γίνεται όμως αν ξέρουμε όλες τις “ζαριές” από πριν; Μπορούμε να σχεδιάσουμε κάποια στρατηγική που να μας οδηγεί στον κόμβο-στόχο με τις ελάχιστες δυνατές κινήσεις;

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε ML και ένα σε Java) τα οποία να παίρνουν ως είσοδο το γράφο των δυνατών κινήσεων μεταξύ κόμβων και μια πεπερασμένη ακολουθία από ζαριές (ακεραίους από 1 έως 6) και να επιστρέφουν ως έξοδο τον ελάχιστο δυνατό αριθμό κινήσεων που πρέπει να κάνουμε για να βρεθούμε από τον αρχικό κόμβο του γράφου (τον κόμβο 1) στον κόμβο-στόχο. Προσέξτε τα εξής:

1. η ακολουθία των ζαριών είναι μεν πεπερασμένη αλλά μπορούμε να τη χρησιμοποιήσουμε ως κυκλική: αν έχει εξαντληθεί και δεν έχουμε καταφέρει να βρεθούμε στον κόμβο-στόχο τότε μπορούμε να την επαναχρησιμοποιήσουμε από την αρχή της από τον κόμβο που βρισκόμαστε
2. ο γράφος είναι κατευθυνόμενος, μπορεί να έχει κύκλους (π.χ. ακμές από έναν κόμβο σε κάποιον άλλο και ξανά στον πρώτο), και φυσικά κατά τη διάρκεια των κινήσεών μας μπορούμε να βρεθούμε σε κάποιον κόμβο περισσότερες από μία φορά
3. μπορεί να μην είναι δυνατό να πάμε από τον αρχικό κόμβο στον κόμβο στόχο ακόμα και αν χρησιμοποιήσουμε την ακολουθία με τις ζαριές άπειρες φορές. (Στην περίπτωση αυτή το πρόγραμμά σας πρέπει να επιστρέφει -1.)

Κάποια από τα αποτελέσματα του προγράμματος σε ML δείχνουν ως εξής:

```
- zaria 3 [(1,2),(2,1),(2,3),(3,2)] [3,5];
val it = 8 : int
- zaria 4 [(1,2),(2,3),(3,4)] [1];
val it = 3 : int
- zaria 3 [(1,2),(2,3)] [4,2,6];
val it = -1 : int
```

Εξηγούμε το πρώτο από αυτά. Το πρώτο του όρισμα είναι ο κόμβος-στόχος. Το επόμενο όρισμα περιέχει τις ακμές του γράφου. Συγκεκριμένα, υπάρχουν ακμές από τον κόμβο 1 στον κόμβο 2, από τον 2 στον 1, από τον 2 στον 3, και από τον 3 στον 2. Το τελευταίο όρισμα είναι η ακολουθία με τις ζαριές. Για να βρεθούμε από τον αρχικό κόμβο 1 στον κόμβο-στόχο 3 πρέπει να χρησιμοποιήσουμε τόσο το ζάρι 3 όσο και το 5. Ένας τρόπος να κινηθούμε είναι ο εξής: 1 → 2 → 1 → 2 (με το ζάρι 3) και στη συνέχεια 2 → 3 → 2 → 3 → 2 → 3 (με το ζάρι 5).

Το πρόγραμμά σας σε Java θα πρέπει να διαβάζει την είσοδο από το stdin ως εξής: Η πρώτη γραμμή περιέχει τον κόμβο-στόχο N και τον αριθμό των ακμών M. Οι επόμενες M γραμμές περιέχουν ζεύγη ακμών. Ακολουθεί ο αριθμός των ζαριών και μια γραμμή με όλες τις ζαριές. Όλα αυτά φαίνονται στο παρακάτω αρχείο:

```
3 4
1 2
2 1
2 3
3 2
2
3 5
```

Περαιτέρω οδηγίες για την άσκηση

- Μπορείτε να δουλέψετε σε ομάδες το πολύ 2 ατόμων (αλλά μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με την προηγούμενη άσκηση – οι ομάδες θα είναι έτσι και αλλιώς καινούργιες)
- Δεν επιτρέπεται να μοιράζεστε ασκήσεις με άλλους συμφοιτητές σας ή να βάλετε τις ασκήσεις σας σε μέρος που άλλοι μπορούν να τις βρουν εύκολα (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοτόπους συζητήσεων, …)
- Τα προγράμματα σε ML πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε SML/NJ v110.67 ή πιο πρόσφατη.
- Τα προγράμματα σε Java μπορεί να είναι σε περισσότερα του ενός αρχείου αλλά θα πρέπει να είναι τέτοια ώστε να μπορούν να μεταγλωττιστούν χωρίς πρόβλημα με τον Java compiler από το command line με εντολές της μορφής `javac Overbooked.java` ή `javac Zaria.java`
- Η αποστολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle, όπως και στην προηγούμενη άσκηση. Θα υπάρξει σχετική ανακοίνωση μόλις το submission site καταστεί ενεργό. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο εκτός από τους αριθμούς που ζητούνται διότι δε θα γίνουν δεκτά από το σύστημα.
- Οι ασκήσεις αυτής της σειράς θα έχουν περιορισμό χρόνου εκτέλεσης ο οποίος θα καθοριστεί με επόμενη ανακοίνωση. Όμως, θα είναι κατά πολύ λιγότερος των 2 ωρών ανά test case της προηγούμενης άσκησης...