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

Εαρινό 2011

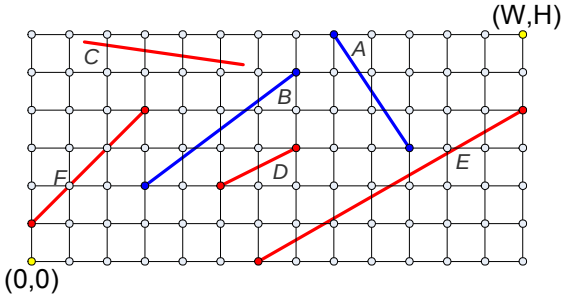
ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 2

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 16/7/2011, 12:30

Βάζε γραμμές (0.25 + 0.25 = 0.5 βαθμοί)

Έστω ένα ορθογώνιο πλέγμα διαστάσεων **W**×**H** όπως αυτό που φαίνεται στο διπλανό σχήμα. Κάθε κόμβος του πλέγματος έχει συντεταγμένες (*x*,*y*), όπου *x* και *y* φυσικοί αριθμοί με 0 ≤ *x* ≤ **W**, 0 ≤ *y* ≤ **H**, και επίσης **W** ≤ 1000, **H** ≤ 1000.



Θέλουμε να βρούμε πόσα ευθύγραμμα τμήματα μπορούμε να τοποθετήσουμε μέσα στο πλέγμα τέτοια ώστε:

- τα άκρα τους να βρίσκονται σε κόμβους του πλέγματος,
- εκτός από τα άκρα τους, δε διέρχονται από κανέναν άλλο κόμβο του πλέγματος, και
- το μήκος τους να είναι στο κλειστό διάστημα μεταξύ $\mathbf{L}_{min}$ και $\mathbf{L}_{max}$, όπου $\mathbf{L}_{min}$ και $\mathbf{L}_{max}$ φυσικοί αριθμοί και $1 \le \mathbf{L}_{min} \le \mathbf{L}_{max} \le 1500$.

Ας υποθέσουμε ότι $\mathbf{L}_{min} = 3$, $\mathbf{L}_{max} = 5$ και ότι έχουμε τοποθετήσει κάποια ευθύγραμμα τμήματα. Από τα ευθύγραμμα τμήματα που φαίνονται στο σχήμα, τα A και B (ζωγραφισμένα μπλε) έχουν μήκη 3.606 και 5.0 πρέπει να προσμετρηθούν. Αντίθετα τα υπόλοιπα (ζωγραφισμένα κόκκινα) δεν πρέπει να προσμετρηθούν. Το C δεν έχει τα άκρα του σε κόμβους του πλέγματος. Το D είναι πολύ μικρό (έχει μήκος 2.236). Το E είναι πολύ μεγάλο (έχει μήκος 8.062). Τέλος, το F διέρχεται από δύο άλλους κόμβους εκτός των άκρων του.

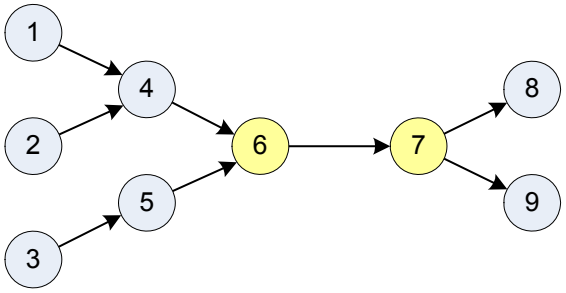
Αυτό που ζητάει η άσκηση είναι να γραφούν δύο προγράμματα (ένα σε ML και ένα σε Java) τα οποία να παίρνουν ως είσοδο τα **W**, **H**, $\mathbf{L}_{min}$ και $\mathbf{L}_{max}$ (όλα ακέραιοι οι οποίοι χωράνε σε 32 bit) και να βρίσκουν το ζητούμενο πλήθος των ευθυγράμμων τμημάτων. Προσέξτε ότι το αποτέλεσμα μπορεί να μη χωράει σε ακέραιο των 32 bit, σίγουρα όμως χωράει σε 64 bit.

Παρακάτω δείχνουμε κάποιες πιθανές κλήσεις των προγραμμάτων σε ML και Java.

```
- lines 2 1 2 3;
val it = 2 : int
- lines 5 3 3 4;
val it = 48 : int
- lines 3 2 1 4;
val it = 49 : int
```

```
> java Lines 2 1 2 3
2
> java Lines 5 3 3 4
48
> java Lines 3 2 1 4
49
```

Γράφο, γράφο τον καημό μου... (0.25 + 0.25 = 0.5 βαθμοί)



Δίνεται ένας κατευθυνόμενος γράφος, όπως αυτός του διπλανού σχήματος. Οι κορυφές του είναι αριθμημένες με τους φυσικούς αριθμούς από το 1 έως το **N** (όπου 2 ≤ **N** ≤ 100,000) και το πλήθος των ακμών είναι **N**–1. Ο γράφος αποτελείται ουσιαστικά από δύο δυαδικά δέντρα. Στο πρώτο δέντρο, οι ακμές κατευθύνονται από τα φύλλα προς τη ρίζα, ενώ στο δεύτερο κατευθύνονται από τη ρίζα προς τα φύλλα. Επίσης, μία ακμή κατευθύνεται από τη ρίζα του πρώτου δέντρου στη ρίζα του δεύτερου δέντρου.

Ας ονομάσουμε τα φύλλα του πρώτου δέντρου «αφετηρίες» και τα φύλλα του δεύτερου δέντρου «προορισμούς». Προφανώς, για κάθε αφετηρία και για κάθε προορισμό υπάρχει ακριβώς ένα μονοπάτι που τα συνδέει.

Θέλουμε να βρούμε όλους τους κόμβους που δεν είναι αφετηρίες και που βρίσκονται σε όλα τα δυνατά μονοπάτια που συνδέουν αφετηρίες με προορισμούς. Στο παράδειγμα του σχήματος, οι κόμβοι αυτοί είναι ο 6 και ο 7.[1] Γράψτε προγράμματα (ένα σε ML και ένα σε Java) που κάνουν αυτή τη δουλειά.

Η είσοδος πρέπει να διαβάζεται από ένα αρχείο το οποίο έχει την εξής μορφή: η πρώτη γραμμή του περιέχει το πλήθος των κόμβων **N**. Κάθε μία από τις επόμενες **N**–1 γραμμές περιέχει δύο φυσικούς αριθμούς **A** και **B**, στο διάστημα από 1 έως και **N**. Αυτό σημαίνει ότι υπάρχει μία ακμή που κατευθύνεται από τον κόμβο **A** προς τον κόμβο **B**.

Η έξοδος θα πρέπει να περιέχει όλους τους ζητούμενους κόμβους, σε αύξουσα σειρά.

Έστω ότι το αρχείο **`input.txt`** έχει το παρακάτω περιεχόμενο (αντιστοιχεί στο σχήμα):

```
9
1 4
3 5
2 4
5 6
6 7
7 8
4 6
7 9
```

Στη Java, έξοδος του προγράμματος θα πρέπει να είναι:

```
6 7
```

Δηλαδή στην έξοδο οι αριθμοί τυπώνονται σε μία γραμμή και διαχωρίζονται μεταξύ τους με ένα κενό. Στο τέλος της εξόδου δεν υπάρχει κάποιος κενός χαρακτήρας.

Σε ένα σύστημα Unix, η χρήση του προγράμματος σε Java δείχνει ως εξής:

```
> javac Grafo.java
> java Grafo < input.txt
6 7
```

Προσέξτε ότι στη Java η είσοδος, παρ' όλο που βρίσκεται σε αρχείο, διαβάζεται από το stdin.

Το πρόγραμμα σε ML επιστρέφει τα αποτελέσματα σε μια λίστα και δείχνει ως εξής:

```
- grafo "input.txt";
val it = [6,7] : int list
```

[1] Με γινόμενο 42...

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 στο σύστημα υποβολής 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 επίσης να είναι σε ένα αρχείο και να δουλεύουν σε SML/NJ (v110.67 ή πιο πρόσφατη έκδοση) ή σε Mlton (20070826 ή πιο πρόσφατη). Το σύστημα ηλεκτρονικής υποβολής επιτρέπει να επιλέξετε μεταξύ αυτών των υλοποιήσεων της ML.
- Ο κώδικας των προγράμματα σε Java μπορεί να βρίσκεται σε περισσότερα του ενός αρχείου αλλά έτσι ώστε να μπορεί να μεταγλωττιστεί χωρίς πρόβλημα με τον Java compiler από το command line με εντολές της μορφής **`javac Lines.java`** ή **`javac Grafo.java`**
- Η υποβολή 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 τη ζητούμενη διότι δε θα γίνουν δεκτά από το σύστημα στο οποίο θα υποβληθούν.