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

Εαρινό 2021

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

Άσκηση 3

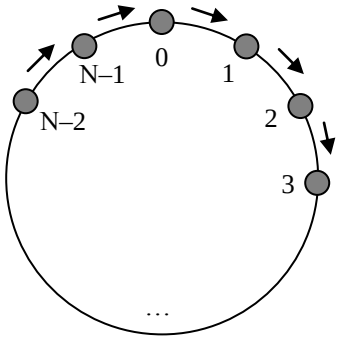
Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 18/7/2021, 23:59:59

Ταξινόμηση με ουρά και στοίβα, ξανά (0.25 βαθμοί)

Το πρόβλημα με την ταξινόμηση των αριθμών χρησιμοποιώντας μία ουρά και μία στοίβα είναι γνωστό από την προηγούμενη σειρά ασκήσεων της φετινής χρονιάς. Το ζητούμενο αυτής της άσκησης είναι να γραφεί η λύση του σε Java. Το πρόγραμμά σας θα πρέπει να έχει την ίδια συμπεριφορά με τα προγράμματα σε Python που παραδώσατε για τη δεύτερη σειρά ασκήσεων. Για τα παραδείγματα της εκφώνησης της δεύτερης σειράς, η έξοδος του προγράμματός σας πρέπει να είναι η εξής (προσέξτε ότι αυτό σημαίνει πως η κύρια κλάση σας — αυτή με τη μέθοδο main — πρέπει να ονομάζεται "`QSsort`", προσοχή στα μικρά και τα κεφαλαία γράμματα):

```
$ java QSsort qs1.txt
QQSQSSQQSS
$ java QSsort qs2.txt
QQQQSQSSSQSSQS
$ java QSsort qs3.txt
QQSQQSQSSS
$ java QSsort qs4.txt
QQQQSSSS
$ java QSsort qs5.txt
empty
```

Γύρω-γύρω όλοι (0.25+0.25+0.25 = 0.75 βαθμοί)



Κατά μήκος ενός κυκλικού δρόμου είναι τοποθετημένες N πόλεις, αριθμημένες από 0 έως και N−1, όπως στο διπλανό σχήμα. Τα οχήματα μπορούν να κινούνται μόνο δεξιόστροφα, δηλαδή από την πόλη 0 προς την πόλη 1, από την πόλη 1 προς την πόλη 2, κ.ο.κ., και κλείνοντας τον κύκλο από την πόλη N–1 προς την πόλη 0. Πάνω σε αυτό το δρόμο υπάρχουν K οχήματα, κάθε ένα από τα οποία τυχαίνει να βρίσκεται αρχικά σε μία από τις πόλεις. Σε κάθε πόλη μπορεί να βρίσκεται κανένα, ένα ή περισσότερα οχήματα.

Κάθε φορά που ένα όχημα κινείται από την πόλη όπου βρίσκεται στην επόμενη κατά μήκος του κύκλου, λέμε ότι γίνεται μία «κίνηση». Θεωρούμε ότι τα οχήματα δεν μπορούν να μετακινούνται ταυτόχρονα, δηλαδή ότι οι «κινήσεις» γίνονται κατά σειρά, η μία μετά την άλλη. Μία σειρά κινήσεων λέμε ότι είναι «νόμιμη» αν δεν κινείται το ίδιο όχημα δύο φορές διαδοχικά, χωρίς ενδιάμεσα να κινηθεί κάποιο άλλο όχημα.

Δεδομένων των N, K και των αρχικών θέσεων των οχημάτων, θέλουμε να βρούμε μία νόμιμη σειρά αποτελούμενη από τις λιγότερες δυνατές κινήσεις, έτσι ώστε στο τέλος της όλα τα οχήματα να βρίσκονται στην ίδια πόλη.

Η άσκηση σας ζητάει να γράψετε τρία προγράμματα (ένα σε ML, ένα σε Prolog και ένα σε Java) τα οποία να διαβάζουν την είσοδο όπως φαίνεται παρακάτω και να βρίσκουν τον ελάχιστο αριθμό κινήσεων M που χρειάζονται για να βρεθούν όλα τα οχήματα στην ίδια πόλη, όπως επίσης και τον αριθμό της πόλης C στην οποία θα καταλήξουν. Αν υπάρχουν περισσότερες νόμιμες σειρές

με τον ελάχιστο αριθμό κινήσεων, το πρόγραμμά σας πρέπει να βρίσκει την ελάχιστη δυνατή τιμή του C. Μπορείτε να υποθέσετε ότι η είσοδος θα είναι τέτοια ώστε το πρόβλημα να έχει λύση (προσέξτε ότι υπάρχουν περιπτώσεις για τις οποίες δεν υπάρχει λύση στο πρόβλημα).

Τα στοιχεία εισόδου θα διαβάζονται από ένα αρχείο με μορφή σαν και αυτή που φαίνεται στα παραδείγματα παρακάτω. Η πρώτη γραμμή του αρχείου θα περιέχει δύο ακέραιους αριθμούς χωρισμένους μεταξύ τους με ένα κενό διάστημα: το πλήθος των πόλεων **N** ($1 \le N \le 10.000$) και το πλήθος των οχημάτων **K** ($2 \le K \le 10.000$). Η δεύτερη γραμμή θα περιέχει K ακέραιους αριθμούς χωρισμένους ανά δύο με ένα κενό διάστημα, που θα παριστάνουν κατά σειρά τον αριθμό της πόλης όπου βρίσκεται αρχικά κάθε όχημα.

Τα προγράμματά σας, ανάλογα με τη γλώσσα υλοποίησής τους, θα πρέπει να συμπεριφέρονται όπως στα επόμενα παραδείγματα. Για όλες τις γλώσσες υλοποίησης εκτός της Prolog, τα αποτελέσματα εκτυπώνονται στην τυπική έξοδο (standard output).

Σε SML/NJ

```
- round "r1.txt";
6 3
val it = () : unit
- round "r2.txt";
3 0
val it = () : unit
```

Σε MLton ή σε OCaml

```
./round r1.txt
6 3
./round r2.txt
3 0
```

Σε Java

```
$ java Round r1.txt
6 3
$ java Round r2.txt
3 0
```

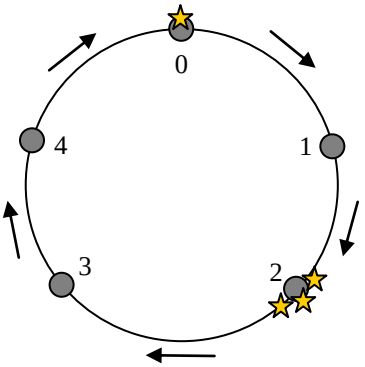
Σε Prolog

```
?- round('r1.txt', M, C), write(M), write(' '), writeln(C), fail.
6 3
false.
?- round('r2.txt', M, C), write(M), write(' '), writeln(C), fail.
3 0
false.
```

όπου τα αρχεία με τα δεδομένα εισόδου είναι τα εξής (η εντολή **cat** είναι εντολή του Unix):

```
$ cat r1.txt
5 4
2 0 2 2
```

```
$ cat r2.txt
3 4
2 0 2 2
```



Η αρχική θέση των οχημάτων για το πρώτο παράδειγμα φαίνεται στο διπλανό σχήμα. Όλα τα οχήματα μπορούν να βρεθούν μετά από 6 κινήσεις στην πόλη 3. Για να γίνει αυτό, το όχημα που βρίσκεται αρχικά στην πόλη 0 μπορεί να κινείται εναλλάξ με ένα από τα οχήματα που βρίσκονται αρχικά στην πόλη 2.

Περαιτέρω οδηγίες για τις ασκήσεις

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων. Μπορείτε αν θέλετε να σχηματίσετε διαφορετική ομάδα σε σχέση με την προηγούμενη σειρά ασκήσεων – οι ομάδες στο σύστημα υποβολής είναι έτσι και αλλιώς καινούργιες για κάθε σειρά ασκήσεων.
- Δεν επιτρέπεται να μοιράζεστε τα προγράμματά σας με συμφοιτητές εκτός της ομάδας σας ή να τα βάλετε σε μέρος που άλλοι μπορούν να τα βρουν (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοσελίδες συζητήσεων, …). Σε περίπτωση που παρατηρηθούν «περίεργες» ομοιότητες σε προγράμματα, ο βαθμός των εμπλεκόμενων φοιτητών σε *όλες τις σειρές ασκήσεων* γίνεται αυτόματα μηδέν ανεξάρτητα από το ποια ομάδα... «εμπνεύστηκε» από την άλλη.
- Μπορείτε να χρησιμοποιήσετε «βοηθητικό» κώδικα (π.χ. κάποιο κώδικα που διαχειρίζεται κάποια δομή δεδομένων) που βρήκατε στο διαδίκτυο στα προγράμματά σας, με την

προϋπόθεση ότι το πρόγραμμά σας περιέχει σε σχόλια την παραδοχή για την προέλευση αυτού του κώδικα και ένα σύνδεσμο σε αυτόν.

- Τα προγράμματα σε Python πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε Python 3.7.3. (Προσέξτε ότι η Python 2 είναι διαφορετική διάλεκτος της Python!)
- Τα προγράμματα σε Prolog πρέπει να είναι σε ένα αρχείο και να δουλεύουν σε κάποιο από τα παρακάτω συστήματα SWI Prolog (8.0.2), GNU Prolog (1.3.0) ή YAP (6.2.2).
- Ο κώδικας των προγραμμάτων σε Java μπορεί να βρίσκεται σε περισσότερα του ενός αρχείου αν θέλετε, αλλά θα πρέπει να μπορεί να μεταγλωττιστεί χωρίς προβλήματα με τον Java compiler με εντολές της μορφής: **`javac QSsort.java`** και **`javac Round.java`**. Μην ορίσετε δικά σας packages! Η υποβολή σας σε Java μπορεί είτε να είναι ένα μόνο **`.java`** αρχείο ή να αποτελείται από ένα **`.zip`** αρχείο ενός directory το οποίο περιέχει τα **`.java`** αρχεία της υποβολής σας (και μόνο αυτά – μην υποβάλετε **`.class`** αρχεία). Σε κάθε περίπτωση, η υποβολή σας πρέπει να έχει ένα αρχείο με τα ονόματα που φαίνονται παραπάνω σε αυτήν την παράγραφο.
- Η υποβολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του moodle, όπως και στην προηγούμενη άσκηση, και για να μπορέσετε να τις υποβάλλετε, τα μέλη της ομάδας σας (και οι δύο) θα πρέπει να έχουν ήδη λογαριασμό στο moodle. Θα υπάρξει σχετική ανακοίνωση μόλις το σύστημα υποβολής 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 υποβολής.