ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

ΑΛΓΟΡΙΘΜΟΙ

Διδάσκουσα: Κ. Παπακωνσταντινοπούλου

Σειρά Προγραμματιστικών Ασκήσεων – Ημ/νία παράδοσης 31/5/2018

Οι λύσεις των ασκήσεων θα πρέπει να υλοποιηθούν σε γλώσσα προγραμματισμού **Java** και τα **αρχεία** σας, μαζί με μια **αναφορά** όπου θα περιγράφεται η δουλειά σας, θα πρέπει να υποβληθούν στο **eClass** σε ένα συμπιεσμένο φάκελο .zip με όνομα τον αριθμό μητρώου σας, δηλαδή 3xxxxxx.zip. Η αναφορά θα πρέπει να περιέχει περιγραφή των αλγορίθμων που χρησιμοποιήσατε και της πολυπλοκότητάς τους, και παρουσίαση των αποτελεσμάτων σας.

Άσκηση 1: Διαίρει-και-Βασίλευε.

Δίνεται ταξινομημένος πίνακας ακεραίων A και ένα στοιχείο του x και ζητείται η συχνότητα εμφάνισης του x στον A. Ένας τρόπος να υπολογίσουμε τη συχνότητα είναι να διατρέξουμε τον πίνακα και μόλις συναντήσουμε για πρώτη φορά το x να αρχίσουμε να μετράμε τις εμφανίσεις του. Αυτός ο αλγόριθμος έχει πολυπλοκότητα $O(n)$ στη χειρότερη περίπτωση. Μπορούμε όμως να κάνουμε καλύτερα.

Να γράψετε πρόγραμμα Java που παίρνει ως είσοδο τα A (από αρχείο) και x υπολογίζει τη ζητούμενη συχνότητα σε χρόνο $O(\log n)$.

Για βοήθεια παρέχονται (στην ενότητα 'Εγγραφα' του eClass, στο φάκελο 'Auxiliary source files') συναρτήσεις για το διάβασμα ενός αρχείου που περιέχει ακεραίους χωρισμένους με ένα κενό μεταξύ τους και την αποθήκευσή του σε μια λίστα. Επίσης παρέχονται (στο φάκελο 'Datasets and Test files') ενδεικτικά αρχεία εισόδου που μπορείτε να χρησιμοποιήσετε για να δοκιμάζετε το πρόγραμμά σας. Προφανώς μπορείτε να δημιουργήσετε και να χρησιμοποιήσετε αντίστοιχα δικά σας αρχεία, προσέχοντας όμως να διατηρήσετε το ίδιο format.

Άσκηση 2: Δυναμικός Προγραμματισμός.

Έχετε να ταξιδέψετε με τρένο προς κάποιο μακρυνό προορισμό και θέλετε να το κάνετε ξοδεύοντας όσο το δυνατόν λιγότερα χρήματα. Το ταξίδι μπορείτε να το κάνετε είτε απευθείας, ή σταματώντας σε ενδιάμεσους σταθμούς.

Έστω ότι ξεκινάτε από το σταθμό 1, προορισμός σας είναι ο σταθμός N και υπάρχουν $N-2$ σταθμοί ενδιάμεσα. Να γράψετε πρόγραμμα Java που παίρνει ως είσοδο ένα $N \times N$ πίνακα ακεραίων C (από αρχείο), όπου:

$$C[i,j] = \begin{cases} 0, & \text{αν } i \geq j \\ c(i,j), & \text{αν } i < j \end{cases}$$

και $c(i,j)$ είναι το κόστος του εισιτηρίου για τη διαδρομή από το σταθμό i στο σταθμό j για κάθε $i,j \in \{1,2,\ldots,N\}$, και επιστρέφει τη διαδρομή με το ελάχιστο κόστος προς τον προορισμό. Συγκεκριμένα, θα πρέπει να εμφανίζει στην έξοδο το ελάχιστο δυνατό κόστος της διαδρομής από το σταθμό 1 στο σταθμό N καθώς και την ακολουθία σταθμών στους οποίους θα σταματήσετε σε αυτή τη διαδρομή (συμπεριλαμβανομένων των 1 και N).

Για βοήθεια παρέχονται (στην ενότητα 'Έγγραφα' του eClass, στο φάκελο 'Auxiliary source files') συναρτήσεις για το διάβασμα ενός αρχείου που περιέχει $N \times N$ πίνακα ακεραίων χωρισμένων με κενά μεταξύ τους και την αποθήκευσή του σε ένα πίνακα. Επίσης παρέχονται (στο φάκελο 'Datasets and Test files') ενδεικτικά αρχεία εισόδου που μπορείτε να χρησιμοποιήσετε για να δοκιμάζετε το πρόγραμμά σας. Προφανώς μπορείτε να δημιουργήσετε και να χρησιμοποιήσετε αντίστοιχα δικά σας αρχεία, προσέχοντας όμως να διατηρήσετε το ίδιο format.

Άσκηση 3: Αποστάσεις σε δίκτυο.

Δίνεται ένα σύνολο δεδομένων που παριστάνει ένα μη κατευθυνόμενο δίκτυο φιλοσόφων, με συνδέσεις μεταξύ φιλοσόφων των οποίων το έργο σχετίζεται. Το δίκτυο παριστάνεται με ζεύγη κόμβων. Το αρχείο με τα δεδομένα βρίσκεται στη διεύθυνση: `https://eclass.aueb.gr/modules/document/file.php/INF161/Datasets/philosophy_edgelist.txt` και μπορείτε να το ανοίξετε αφού κάνετε login στο eClass.

Να γράψετε πρόγραμμα Java που παίρνει ως είσοδο δύο φιλοσόφους και βρίσκει το συντομότερο τρόπο με τον οποίο συνδέονται στο δίκτυο. Αν όντως συνδέονται, να εμφανίζει το αντίστοιχο μονοπάτι μεταξύ τους, αλλιώς να εμφανίζει κατάλληλο μήνυμα.

(Προτείνεται για την αναπαράσταση του γράφου να χρησιμοποιήσετε hash map με κλειδί το όνομα του κόμβου (τύπου String) και τιμή τη λίστα των γειτόνων του κόμβου αυτού (τύπου List<String>).)

Άσκηση 4: NP-δύσκολα προβλήματα.

Θεωρήστε ένα online κοινωνικό δίκτυο (social network), αποτελούμενο από χρήστες και συνδέσεις μεταξύ τους που αναπαριστούν φιλίες, στο οποίο κάθε χρήστης ενημερώνεται για το περιεχόμενο που αναρτά κάποιος φίλος του (μπορείτε να σκεφτείτε σαν παράδειγμα το facebook).

Μια εταιρεία κινητής τηλεφωνίας θέλει να διαφημίσει το καινούριο μοντέλο τηλεφώνου της, και αποφασίζει να το κάνει με τον εξής τρόπο: Θα δώσει το τηλέφωνο δωρεάν σε ορισμένους χρήστες, και αυτοί θα αναρτήσουν στο προφίλ τους τη διαφήμιση του μοντέλου αυτού. Με το που αναρτά κάποιος χρήστης τη διαφήμιση, όλοι οι φίλοι του ενημερώνονται για αυτή. Στόχος της εταιρείας είναι να φτάσει η διαφήμιση σε όλους τους χρήστες με το ελάχιστο δυνατό κόστος, δηλαδή δίνοντας το ελάχιστο δυνατό πλήθος τηλεφώνων. Υποθέτουμε ότι ο στόχος της ενημέρωσης κάθε χρήστη επιτυγχάνεται την πρώτη φορά που θα φτάσει η διαφήμιση σε αυτόν, και είναι αδιάφορο το πόσες άλλες φορές θα φτάσει.

Εδώ μπορούμε να σκεφτούμε δύο απλούς αλγόριθμους επίλυσης του προβλήματος:

1. Εξαντλητική αναζήτηση: Ξεκινάμε με $k = 1$ και ελέγχουμε όλα τα δυνατά υποσύνολα μεγέθους k για να βρούμε κάποιο που αποτελεί λύση του προβλήματός μας. Αν δε βρεθεί λύση, αυξάνουμε το k κατά 1 και επαναλαμβάνουμε.

2. Άπληστη προσέγγιση: Επιλέγουμε το χρήστη v που έχει το μεγαλύτερο βαθμό, τον εισάγουμε στο σύνολο επιλεγμένων χρηστών, και αφαιρούμε από το δίκτυο όλες τις ακμές που προσπίπτουν στον v. Επαναλαμβάνουμε μέχρι το δίκτυο να μείνει χωρίς ακμές.

Να γράψετε πρόγραμμα Java που παίρνει ως είσοδο ένα δίκτυο σε μορφή λίστας ακμών (παρέχονται test data στην ενότητα 'Έγγραφα' του eClass, στα οποία οι ακμές δίνονται σαν ζεύγη ακεραίων χωρισμένων με tab μεταξύ τους) και βρίσκει το σύνολο χρηστών στους οποίους πρέπει να δοθεί δωρεάν το κινητό έτσι ώστε να διαδοθεί η διαφήμιση σε όλο το δίκτυο, χρησιμοποιώντας κάθε έναν από τους παραπάνω αλγόριθμους (άρα σε κάθε εκτέλεση θα παίρνουμε *δύο* αποτελέσματα).

Να εκτελέσετε το πρόγραμμά σας για δίκτυα διαφόρων μεγεθών, να μετρήσετε το χρόνο εκτέλεσης κάθε αλγορίθμου και να δώσετε σε μια γραφική παράσταση τους χρόνους εκτέλεσης ως συνάρτηση του πλήθους χρηστών του δικτύου. Τι τάξης μεγέθους είναι ο χρόνος εκτέλεσης κάθε αλγορίθμου ως προς το πλήθος των χρηστών και τι παρατηρείτε όσον αφορά τις λύσεις που δίνουν οι δύο αλγόριθμοι; Είναι το ίδιο καλές; Σχετίζονται οι χρόνοι εκτέλεσης με την ποιότητα των λύσεων; Σχολιάστε/δικαιολογήστε σύντομα την απάντησή σας.

(Προτείνεται για την αναπαράσταση του γράφου να χρησιμοποιήσετε hash map με κλειδί το όνομα του κόμβου (τύπου String) και τιμή τη λίστα των γειτόνων του κόμβου αυτού (τύπου List<String>).)