Ανάπτυξη Λογισμικού Πληροφορικής
Χειμερινό Εξάμηνο 2016-2017

“Εύρεση συντομότερων μονοπατιών σε γράφο”

Υπέυθυνος Καθηγητής	**Συνεργάτες Μαθήματος**
Ιωάννης Ιωαννίδης	Μαίλης Θεόφιλος
	Γιαννακόπουλος Αναστάσιος
	Γαβανάς Χρήστος
	Γαλούνη Κωνσταντίνα
	Τσαμπίκος Λιβισιανός

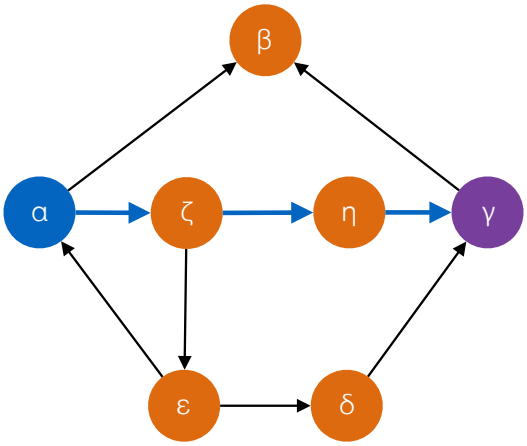
Γενική Περιγραφή

Η εργασία του μαθήματος “Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα” βασίζεται στον διαγωνισμό που πραγματοποιήθηκε στο Sigmod το έτος 2016. Για περισσότερες πληροφορίες ο χρήστης μπορεί να ανατρέξει στη σύνδεσμο για το διαγωνισμό του συνεδρίου. (http://dsg.uwaterloo.ca/sigmod16contest/).

Πρόβλημα του ελάχιστου μονοπατιού

Ορισμός. Στην θεωρία γράφων, δοθέντος ενός γράφου και δύο κόμβων αυτού, το *πρόβλημα του ελάχιστου μονοπατιού* συνίσταται στην εύρεση του μονοπατιού μεταξύ των δύο κόμβων με το ελάχιστο μήκος, όπου το μήκος (στην περίπτωσή μας) μετράται ως ο αριθμός των ακμών του μονοπατιού.

Παραδείγματα. Στον κατευθυνόμενο γράφο της Εικόνας 1, το ελάχιστο μονοπάτι από τον κόμβο α στον κόμβο γ είναι (α,ζ,η,γ).



Εικόνα 1. *Βλέπουμε έναν κατευθυνόμενο γράφο και το ελάχιστο μονοπάτι μεταξύ δύο κόμβων.*

Εφαρμογές. Το συγκεκριμένο πρόβλημα συνδυαστικής βελτιστοποίηση έχει μελετηθεί διεξοδικά με πολλές πρακτικές εφαρμογές: καθοδήγηση σε GPS, δρομολόγηση σε δίκτυα υπολογιστών, μηχανές αναζήτησης για την διασύνδεση μεταξύ ιστοσελίδων, ενώ τα δίκτυα κοινωνικής δικτύωσης εφαρμόζουν τους αλγορίθμους επίλυσης του προβλήματος σε γράφους που περιγράφουν τη σχέση μεταξύ διαφορετικών ατόμων.

Άσκηση

Σε αυτήν την άσκηση, στόχος είναι η απάντηση ερωτημάτων συντομότερου μονοπατιού σε ένα γράφους. Είσοδος στην εφαρμογή σας θα είναι ένας γράφος και μια σειρά ενεργειών και ερωτημάτων. Για το γράφο θα δημιουργήσετε και θα συντηρήσετε δομές για να επιταχύνετε τις επερωτήσεις σε αυτόν. Μετά την αποθήκευση του αρχικού γράφου, θα δοθεί φόρτος εργασίας που συνίσταται από μία σειρά εργασιών σε ριπές. Κάθε εργασία είναι είτε μία πράξη τροποποίησης του γράφου, είτε ένα ερώτημα συντομότερου μονοπατιού μεταξύ δύο κόμβων στον γράφο.

Η εφαρμογή σας αναμένεται να απαντάει με σωστό τρόπο όλα τα ερωτήματα θεωρώντας ότι οι εργασίες εκτελούνται με την σειρά που δίνονται. Οι γράφοι που θα χρησιμοποιηθούν είναι κατευθυνόμενοι, η είσοδος και η έξοδος θα πρέπει να έχουν μία προκαθορισμένη μορφή.

Περιγραφή παραδοτέων μαθήματος

Η εργασία θα χωριστεί σε τρία επίπεδα. Το κάθε επίπεδο θα υλοποιηθεί σε συγκεκριμένη χρονική περίοδο και θα βασίζεται στα προηγούμενα επίπεδα. Ο διαχωρισμός των επιπέδων στοχεύει: στην ανάπτυξη μιας απλής λύσης του προβλήματος και έπειτα στη σταδιακή βελτιστοποίηση των επιμέρους τμημάτων, βελτιώνοντας με αυτό τρόπο την αποδοτικότητα της εφαρμογής.

Στη συνέχεια παρουσιάζονται οι λειτουργίες που θα αναπτυχθούν στο πρώτο επίπεδο.

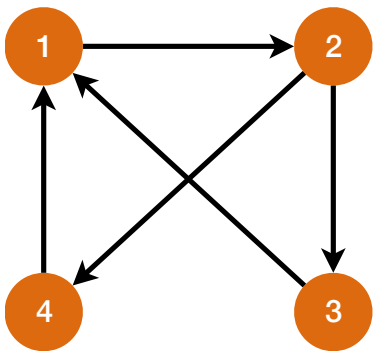
1. Είσοδος γράφου και δημιουργία δομών για την αποθήκευσή του.
2. Υλοποίηση αλγορίθμου υπολογισμού συντομότερου μονοπατιού.
3. Υλοποίηση test main για τον έλεγχο των δομών και της αναζήτησης.
4. Υλοποίηση αλγορίθμου επίλυσης του προβλήματος.

Α’ Μέρος της Άσκησης

1. Είσοδος και αποθήκευση γράφου

Αρχικά θα τροφοδοτηθεί ο γράφος στην είσοδο της εφαρμογής σας. Ο γράφος αναπαρίσταται ως λίστα από ακμές, και η κάθε ακμή συνίσταται από ένα ζεύγος ids για κόμβους (ο κόμβος από όπου ξεκινάει η ακμή και ο κόμβος στον οποίο τελειώνει) οι οποίοι αναπαρίστανται σαν φυσικοί αριθμοί. Το πρόγραμμα θα λάβει πολλαπλές γραμμές (μία ανά ακμή) οι οποίες περιέχουν ακριβώς 2 φυσικούς ακέραιους αριθμούς που διαχωρίζονται από έναν κενό χαρακτήρα. Ο πρώτος εκ τον δύο αριθμών είναι ο κόμβος αφετηρίας (source node) και ο δεύτερος ο κόμβος προορισμού (destination node). Ο αρχικός γράφος τελειώνει με μία γραμμή η οποία περιέχει μόνο τον χαρακτήρα ’S’.

```
1 2
2 3
3 1
4 1
2 4
S
```



Εικόνα 2. *Αναπαράσταση γράφου εισόδου*

Στο παραπάνω παράδειγμα, η είσοδος στο αριστερό μέρος περιγράφει τον γράφο στο δεξί μέρος. Η ακμή (1,2) ξεκινάει από τον κόμβο 1 και καταλήγει στον κόμβο 2. Τα IDs των κόμβων που εμφανίζονται στην είσοδο σε οποιαδήποτε σειρά, αλλά δεν έχουν κενά μεταξύ τους και θα ξεκινήσουν από το 0. Θεωρούμε ότι ο μέγιστος αριθμός που μπορεί να παρουσιαστεί σαν είσοδος είναι ο 2^{24}-1. Θεωρούμε ότι υπάρχει το πολύ μία ακμή ανά κατεύθυνση μεταξύ δύο κόμβων. Συνεπώς αν δοθεί δύο φορές μια είσοδος (1,3) μόνο μία ακμή θα πρέπει να περιλαμβάνεται στον γράφο. Αφού διαβαστεί ο αρχικός γράφος θα πρέπει να αναπαρασταθεί εσωτερικά στις δομές σας.

Δομή αναπαράστασης γράφου

Για την αναπαράσταση του γράφου θα χρησιμοποιήσετε λίστες γειτνίασης, για κάθε κόμβο θα διατηρείται μια λίστα που θα περιέχει τους κόμβους προορισμού των εξερχόμενων ακμών και μια λίστα με τις εισερχόμενες. Κάθε κόμβος αναγνωρίζεται από ένα id και κάθε ακμή χαρακτηρίζεται από το id του κόμβου προέλευσης και το id του κόμβου προορισμού.

Για την αποθήκευση κάθε λίστας γειτνίασης του γράφου στη μνήμη θα χρειαστούν δυο δομές:

- **ο buffer:** αποθηκεύει τις λίστες γειτνίασης για κάθε κόμβο του γράφου
- **το ευρετήριο κόμβων:** μας επιτρέπει να ανακτήσουμε τις πληροφορίες για τον κόμβο που θέλουμε από τον buffer.

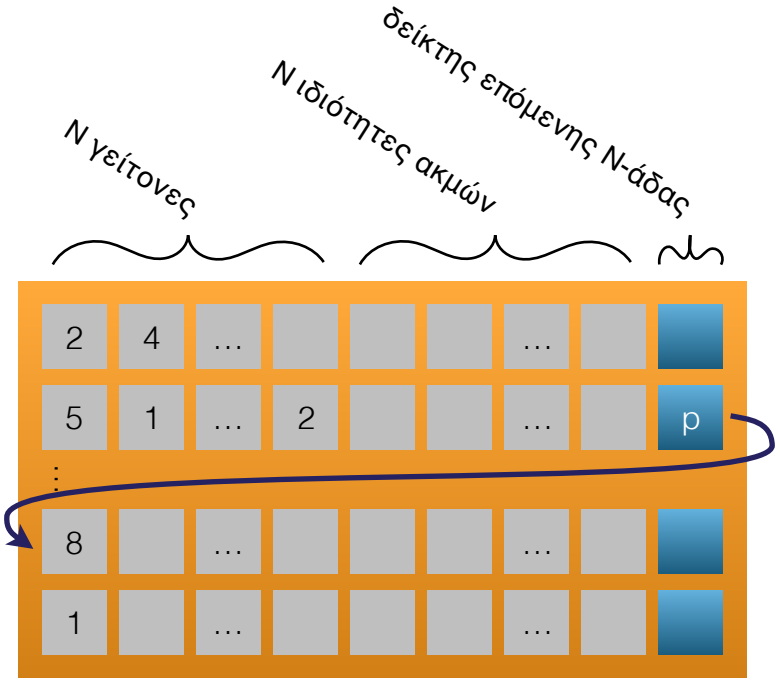
Buffer

Η δομή αυτή (Σχήμα 3) αποθηκεύει για κάθε κόμβο τους γείτονες του. Κάθε κόμβος του γράφου θα έχει την δική του λίστα γειτνίασης, κάθε κόμβος της λίστας θα περιέχει ένα πίνακα σταθερού μεγέθους με τους γείτονες και τις ιδιότητες των ακμών. Το μέγεθος του πίνακα θα πρέπει να είναι τέτοιο ώστε να ελαχιστοποιούμε τις μετακινήσεις στο χώρο για να προσπελάσουμε τους γείτονες ενός κόμβου αλλά και την σπατάλη χώρου

```
struct list_node{

      uint32_t  neighbor[N];       //the ids of the neighbor nodes

      uint32_t  edgeProperty[N];   //property for each edge

      ptr  nextListNode;          //

};
```

Η δομή αυτή θα δεσμεύει ένα συνεχόμενο χώρο μνήμης και θα αποθηκεύει τις λίστες γειτνίασης σε αυτόν. Με αυτό τον τρόπο μπορούμε να εκμεταλλευτούμε τις ιδιότητες της τοπικής χωρητικότητας κατά τους υπολογισμούς των επερωτήσεων.

Καθώς το μέγεθος του γράφου δεν είναι γνωστό από την αρχή η δομή του Buffer θα πρέπει να διπλασιάζεται.



Εικόνα 3. *Δομή Buffer*

Λειτουργίες δομής Buffer

```
Buffer* createBuffer();

ptr allocNewNode(Buffer*);

struct list_node* getListNode(ptr);


OK_SUCCESS destroyBuffer(Buffer*);
```

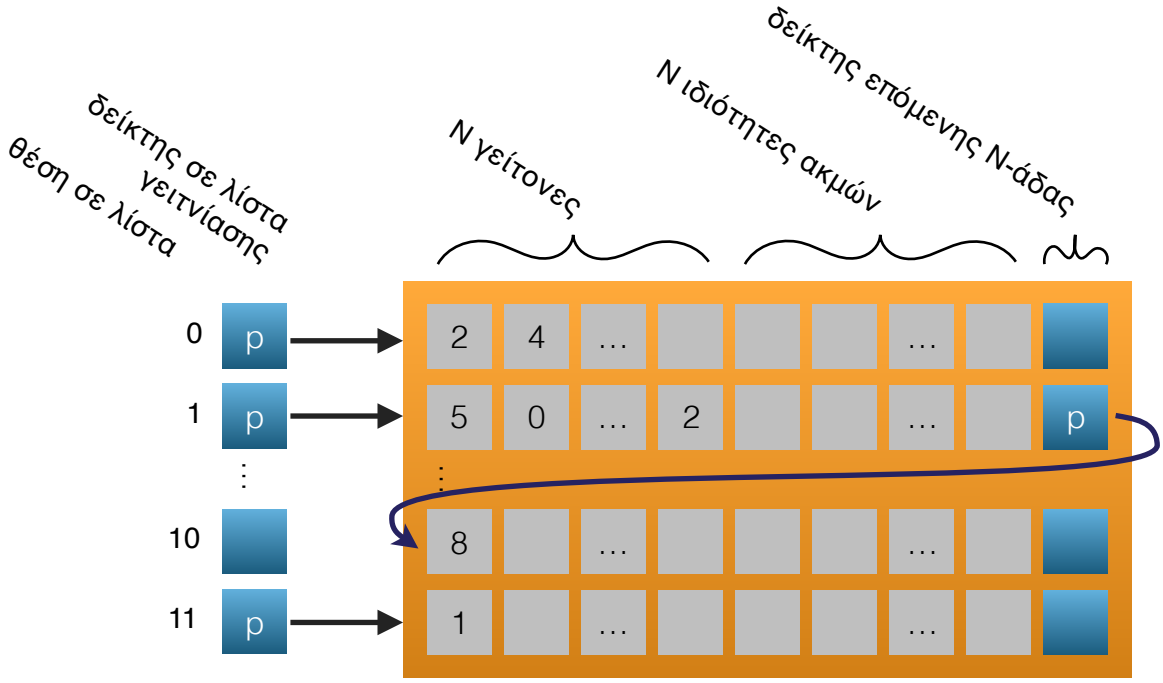
Ευρετήριο κόμβων

Το ευρετήριο των κόμβων επιτρέπει την ανάκτηση των γειτόνων από την δομή του buffer, για κάθε κόμβο το ευρετήριο διατηρεί ένα δείκτη προς την αρχή της λίστας γειτνίασης του (Εικόνα 4).

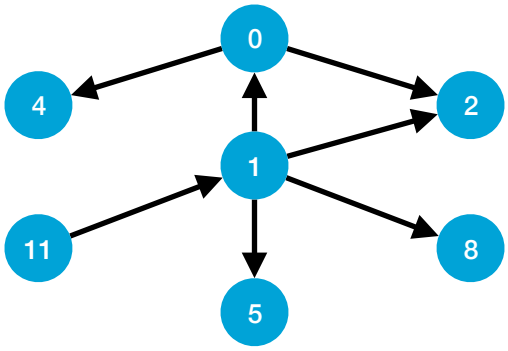
Καθώς ο γράφος είναι δυναμικός θα πρέπει και η δομή αυτή να μπορεί να μεγαλώσει.

Λειτουργίες ευρετηρίου κόμβων

```
NodeIndex* createNodeIndex();

OK_SUCCESS insertNode(NodeIndex*, uint32_t nodeId,.. );

ptr getListHead( NodeIndex*, uint32_t nodeId );

OK_SUCCESS destroyNodeIndex( NodeIndex* );
```



Εικόνα 4. *Συνολική δομή αποθήκευσης γράφου.*



Εικόνα 5. *Γράφος που αντιστοιχεί στην δομή αποθήκευσης της εικόνας 4.*

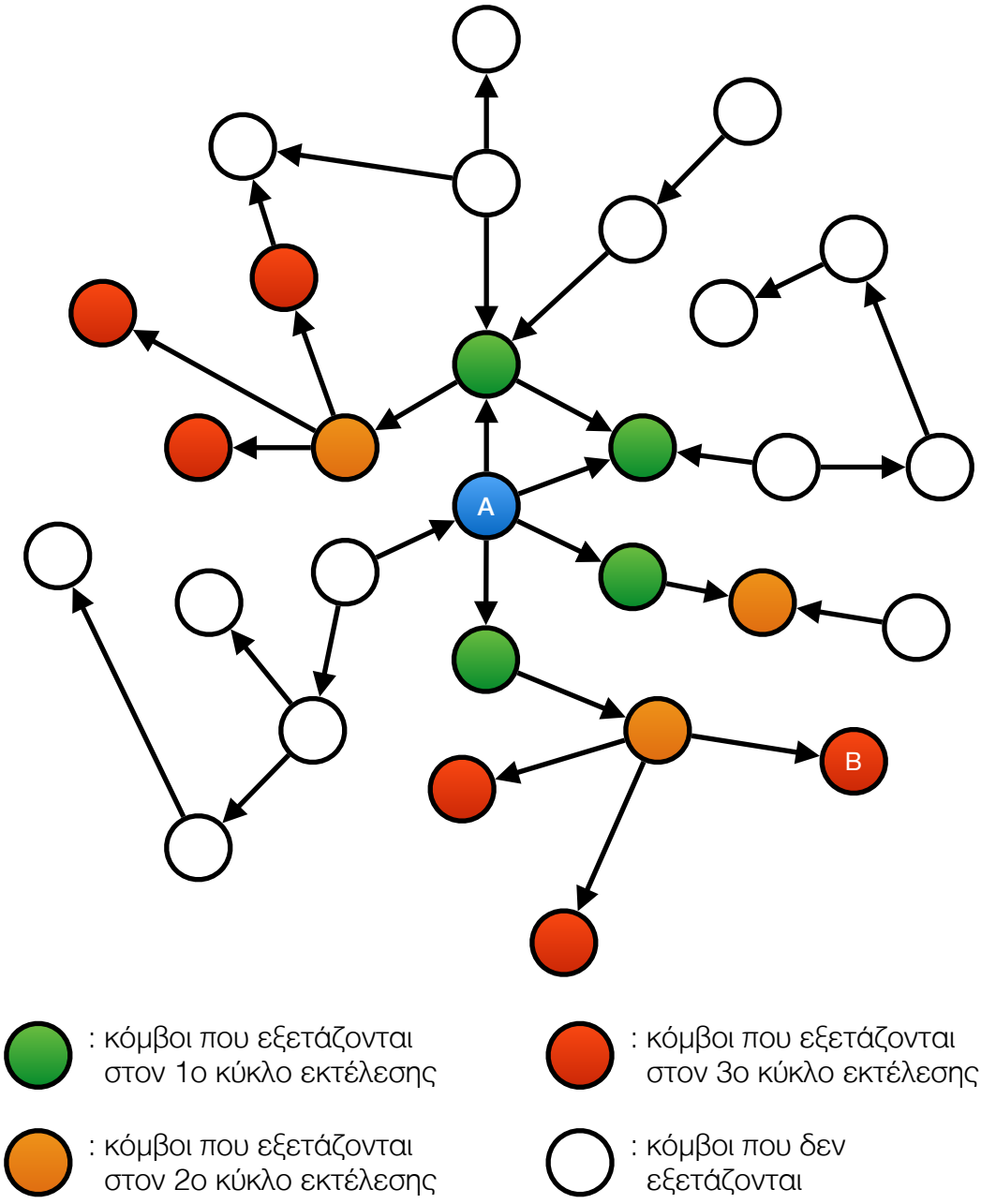
Θα χρειαστείτε δυο ζευγάρια buffer-ευρετηρίου για την αποθήκευση του γράφου, ένα για την αποθήκευση των εξερχόμενων ακμών και ένα για την αποθήκευση των εισερχόμενων ακμών.

2. Υλοποίηση αλγόριθμου εύρεσης συντομότερου μονοπατιού

Για την εύρεση του συντομότερου μονοπατιού θα χρησιμοποιηθεί ο αλγόριθμος Bidirectional Breadth-First-Search.

Ο αλγόριθμος Breadth-First-Search προκειμένου να βρει το συντομότερο μονοπάτι μεταξύ του κόμβου Α και του κόμβου Β λειτουργεί ως εξής:

- Στον πρώτο κύκλο εκτέλεσης εξετάζει όλους τους κόμβους που έχουν μία ακμή από τον Α προς αυτούς. Αν ο Β ανήκει σε αυτούς η διαδικασία τερματίζεται.
- Στον δεύτερο κύκλο θα εξετάσει όλους τους κόμβους από τους οποίους υπάρχει μία ακμή από κάποιον διάδοχο του Α προς αυτούς. Δηλαδή όλους τους κόμβους στους οποίους μπορώ να φτάσω με αφετηρία το Α μέσω δύο ακμών. Η διαδικασία επαναλαμβάνεται για μεγαλύτερες διαδρομές μέχρι να βρεθεί ο κόμβος Β.



- Στην περίπτωση γράφου ο οποίος περιέχει κύκλους, θα πρέπει ο αλγόριθμος να αποτρέπει την επίσκεψη του ίδιου κόμβου περισσότερες της μίας φορές.

Στην παρακάτω Εικόνα βλέπουμε ένα παράδειγμα αναζήτησης με Breadth-First-Search από

τον κόμβο Α στον κόμβο Β. Χρησιμοποιούμε διαφορετικό φόρμα για τους κόμβους που θα επισκεφτεί ο αλγόριθμός μας στον 1ο, 2ο κύκλο κοκ…

Για τον Bidirectional BFS δυο BFS αναζητήσεις ξεκινάνε ταυτόχρονα. Μια απο το κόμβο προορισμού και μια απο τον κόμβο προέλευσης. Η αναζήτηση σταματάει όταν οι δύο αναζητήσεις συναντηθούν.

3. Υλοποίηση tests για τον έλεγχο των δομών δεδομένων και της αναζήτησης

Θα πρέπει να φτιάξετε τρόπους ελέγχου των βασικών λειτουργιών που αναπτύξατε τόσο για την δομή αποθήκευσης του γράφου και για την αναζήτηση συντομότερου μονοπατιού. Ουσιαστικά, θα πρέπει για συγκεκριμένη είσοδο το πρόγραμμα να παράγει την αναμενόμενη έξοδο. Καλό είναι το πρόγραμμα σας να προκαλεί δυναμικές μεταβολές στις δομές σας (π.χ. διπλασιασμός) προκειμένου να εξασφαλίσετε ότι λειτουργούν σωστά. Ένας προτεινόμενος τρόπος για να ελέγχετε τις δομές σας είναι με τη χρήση unit tests (https://en.wikipedia.org/wiki/Unit_testing).

4. Υλοποίηση αλγορίθμου επίλυσης του προβλήματος

Μετά την αποθήκευση του γράφου στην είσοδο της εφαρμογής σας θα δίνονται ριπές εργασιών. Κάθε ριπή περιλαμβάνει μία σειρά εργασιών, μια εργασία ανά γραμμή. Το τέλος της ριπής το σηματοδοτεί μια γραμμή που περιέχει μόνο το χαρακτήρα ’F’.

Κάθε εργασία θα πρέπει να εκτελείται χωρίς να την επηρεάζουν οι επόμενες εργασίες της ριπής που ανήκει. Τα αποτελέσματα των ερωτημάτων που περιέχονται μέσα σε μια ριπή πρέπει να υπολογίζονται και να εμφανίζονται πριν την εκτέλεση των εργασιών της επόμενης ριπής. Μετά το τέλος κάθε πακέτου πράξεων, αναμένεται να εμφανίσετε στην έξοδο της εφαρμογής σας το αποτέλεσμα των εργασιών, πριν διαβάσετε την επόμενη ριπή.

Κάθε εργασία αναπαρίσταται από έναν χαρακτήρα (‘Q' ή 'A') ο οποίος ορίζει και τον τύπο της εργασίας, ακολουθούμενος από ένα κενό και δύο integer αριθμούς που διαχωρίζονται με ένα κενό. Οι δύο αριθμοί αντιστοιχούν σε IDs κόμβων.

Οι δυο τύποι εργασιών είναι:

‘Q’/ερώτημα: Αυτή η εργασία θέτει ένα ερώτημα ελαχίστου μονοπατιού από τον αρχικό στον τελικό κόμβο. Για παράδειγμα το ερώτημα “Q 1 2” θα πρέπει να μας δώσει σαν απάντηση το μήκος του ελαχίστου μονοπατιού μεταξύ των κόμβων 1 και 2. Η απάντηση του ερωτήματος θα πρέπει να εμφανιστεί σαν έξοδος με τη μορφή μίας μονής γραμμής που περιέχει σε δεκαδική αναπαράσταση την απόσταση μεταξύ των δύο κόμβων, δηλαδή, τον ελάχιστο αριθμό ακμών που χρειάζονται για να φτάσουμε από τον αρχικό στον τελικό κόμβο. Σε περίπτωση που δεν υπάρχει μονοπάτι μεταξύ των κόμβων ή κάποιος από τους

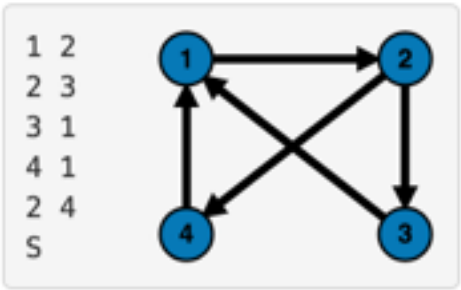
κόμβους δεν εμφανίζεται στον γράφο, η απάντηση πρέπει να είναι -1. Η απόσταση μεταξύ ενός κόμβου και του εαυτού του είναι πάντα 0.

'A'/προσθήκη: Αυτή η εργασία απαιτεί από τον χρήστη να τροποποιήσει τον γράφο του προσθέτοντας μία επιπλέον ακμή από τον πρώτο κόμβο προς τον δεύτερο. Έτσι για παράδειγμα η πράξη “A 2 3” θα προσθέσει μία νέα ακμή από τον κόμβο 2 στον κόμβο 3. Σε περίπτωση που υπάρχει ήδη η συγκεκριμένη ακμή, ο γράφος μας παραμένει αναλλοίωτος. Σε περίπτωση που ένας από τους κόμβους δεν υπάρχει στον γράφο θα πρέπει να προστεθεί. Η συγκεκριμένη εργασία δεν παράγει κάποια έξοδο.

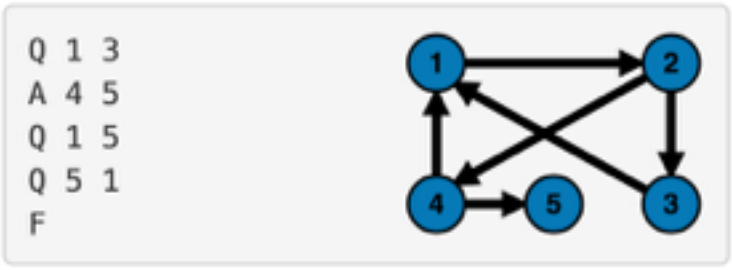
Για ερωτήματα (Q) θα πρέπει η έξοδος να περιέχει τον ίδιο αριθμό γραμμών με τα ερωτήματα στη είσοδο.

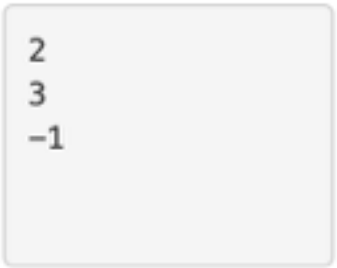
Ακολουθεί ένα παράδειγμα από μια ριπή εργασιών και της αναμενόμενης εξόδου:

Με αρχικό γράφο τον:



Εκτελούμε τις ενέργειες τις παρακάτω ριπής που φαίνεται στα αριστερά παρακάτω εικόνας, στη μέση φαίνεται ο γράφος μετά την εκτέλεση των εισαγωγών και στα δεξιά η έξοδος που θα πρέπει παράξει η εφαρμογή σας.





Η ριπή του παραδείγματος έχει 3 ερωτήσεις άρα και 3 γραμμές εξόδου.

Παρατηρήσεις

- Τα πρότυπα των συναρτήσεων δίνονται σε C, στην περίπτωση υλοποίησης σε κάποια αντικειμενοστραφή γλώσσα το πρώτο όρισμα παραλείπετε.
- Επιπλέον ορίσματα μπορούν να προστεθούν στα πρότυπα των συναρτήσεων.