Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/progtech/

Προγραμματιστικές Τεχνικές

Άσκηση 12 Μας τέλειωσε η βενζίνη

Προθεσμία υποβολής στον grader: 28/5/2021

Δίνεται το οδικό δίκτυο μίας χώρας. Αποτελείται από N πόλεις, αριθμημένες από 0 έως $N-1$, και M δρόμους. Κάθε δρόμος είναι διπλής κατεύθυνσης, συνδέει μεταξύ τους δύο (διαφορετικές) πόλεις και είναι γνωστό το μήκος του σε χιλιόμετρα.

Οδηγούμε ένα αυτοκίνητο και θέλουμε να ταξιδέψουμε από την πόλη A στην πόλη B. Το ντεπόζιτο του αυτοκινήτου μας έχει μια συγκεκριμένη χωρητικότητα και, αν το γεμίσουμε με βενζίνη, αυτή μας φτάνει για να κάνουμε C χιλιόμετρα. Δυστυχώς, βενζινάδικα υπάρχουν μόνο στις πόλεις — δεν υπάρχουν βενζινάδικα κατά μήκος των δρόμων. Στην αρχή του ταξιδιού μας, το ντεπόζιτο είναι άδειο (και άρα θα πρέπει να το γεμίσουμε στην πόλη A, πριν ξεκινήσουμε).

Ζητείται να γράψετε ένα πρόγραμμα που θα διαβάζει το οδικό δίκτυο και στη συνέχεια θα διαβάζει και θα απαντά μία σειρά ερωτημάτων. Για κάθε ερώτημα θα δίνονται οι τιμές των A, B και C και θα ζητούνται τα εξής:

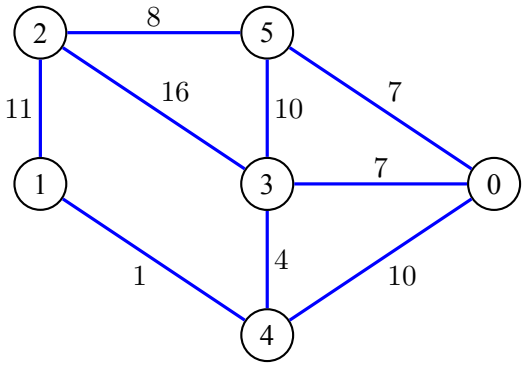
- Είναι δυνατό να φτάσει το αυτοκίνητο από την πόλη A στην πόλη B, αν η χωρητικότητα του ντεπόζιτου επαρκεί για C χιλιόμετρα;
- Αν ναι, ποια είναι μία δυνατή διαδρομή και πόσες φορές θα χρειαστεί να γεμίσουμε το ντεπόζιτο;

Είσοδος. Η πρώτη γραμμή της τυπικής εισόδου (`stdin`) θα περιέχει δύο ακέραιους αριθμούς N και M, χωρισμένους μεταξύ τους με ένα κενό διάστημα: το πλήθος των πόλεων και το πλήθος των δρόμων. Κάθε μία από τις επόμενες M γραμμές θα περιέχει τρεις ακέραιους αριθμούς, U, V και L, χωρισμένους ανά δύο με ένα κενό διάστημα, και θα παριστάνει ένα δρόμο που συνδέει τις πόλεις U και V και έχει μήκος L χιλιόμετρα. Η επόμενη γραμμή θα περιέχει έναν ακέραιο αριθμό Q: το πλήθος των ερωτημάτων. Κάθε μία από τις επόμενες Q γραμμές θα περιέχει τρεις ακέραιους αριθμούς, A, B και C, χωρισμένους ανά δύο με ένα κενό διάστημα, και θα παριστάνει ένα ερώτημα όπως περιγράφηκε παραπάνω.

Έξοδος. Το πρόγραμμά σας πρέπει να εκτυπώνει τα αποτελέσματά του στη τυπική έξοδο (`stdout`). Συνολικά πρέπει να εκτυπώνονται Q γραμμές, που κάθε μία θα περιέχει την απάντηση στο αντίστοιχο ερώτημα, κατά σειρά. Αν η απάντηση στο ερώτημα είναι ότι δεν είναι δυνατή η μετάβαση από την πόλη A στην πόλη B, τότε η γραμμή θα περιέχει τη λέξη "`IMPOSSIBLE`". Διαφορετικά, αν είναι δυνατή η μετάβαση, η γραμμή θα ξεκινά με τη λέξη "`POSSIBLE`" και θα περιέχει μία δυνατή διαδρομή (αν υπάρχουν περισσότερες δυνατές διαδρομές, μπορείτε να επιλέξετε οποιαδήποτε) και το ελάχιστο πλήθος των γεμισμάτων που απαιτούνται για την επιλεγείσα διαδρομή. Η διαδρομή θα δίνεται ως μία ακολουθία ακεραίων αριθμών, χωρισμένων ανά δύο με ένα κενό διάστημα, που θα αντιστοιχούν στις πόλεις από τις οποίες θα διέλθει το αυτοκίνητο. Η πρώτη πόλη της ακολουθίας πρέπει να είναι η A και η τελευταία η B. Δείτε το παράδειγμα που ακολουθεί (στην επόμενη σελίδα) για την ακριβή μορφή της γραμμής που πρέπει να εκτυπώνετε.

Εξήγηση παραδείγματος. Υπάρχουν 6 πόλεις, αριθμημένες από το 0 έως και το 5, και 9 δρόμοι. Ο πρώτος δρόμος συνδέει τις πόλεις 5 και 3 και έχει μήκος 10 χιλιόμετρα. Δίνονται 5 ερωτήματα, τα οποία μπορούν να απαντηθούν κατά σειρά ως εξής:

Παράδειγμα:



Είσοδος:

```
6 9
5 3 10
1 2 11
4 3 4
5 2 8
2 3 16
5 0 7
3 0 7
4 0 10
1 4 1
5
1 2 17
4 5 1
2 3 14
4 2 7
1 5 7
```

Έξοδος:

```
POSSIBLE: 1 fill(s), 1 2
IMPOSSIBLE
POSSIBLE: 2 fill(s), 2 5 3
IMPOSSIBLE
POSSIBLE: 3 fill(s), 1 4 3 0 5
```

1. Από την πόλη $A = 1$ στην πόλη $B = 2$ με χωρητικότητα $C = 17$. Αυτό είναι εύκολο γιατί οι δύο πόλεις απέχουν 11, επομένως μία δυνατή διαδρομή είναι η `1 2` και απαιτεί ένα γέμισμα, στην αρχή, στην πόλη 1.

2. Από την πόλη $A = 4$ στην πόλη $B = 5$ με χωρητικότητα $C = 1$. Αυτό είναι αδύνατο γιατί με $C = 1$ το αυτοκίνητο μπορεί να ακολουθήσει μόνο δρόμους μέγιστου μήκους 1 χιλιομέτρου.

3. Από την πόλη $A = 2$ στην πόλη $B = 3$ με χωρητικότητα $C = 14$. Ο απευθείας δρόμος έχει μήκος 16 χιλιόμετρα, άρα δεν μπορεί να χρησιμοποιηθεί. Υπάρχουν όμως πολλές δυνατές διαδρομές. Μία είναι η `2 5 3`, με 2 γεμίσματα, ένα στην πόλη 2 και ένα στην πόλη 5. Μερικές άλλες σωστές απαντήσεις σε αυτό το ερώτημα είναι οι ακόλουθες. Προσέξτε ότι το πλήθος των απαιτούμενων ελάχιστων γεμισμάτων αλλάζει ανάλογα με τη διαδρομή.

```
POSSIBLE: 2 fill(s), 2 5 0 3
POSSIBLE: 3 fill(s), 2 5 0 4 3
POSSIBLE: 2 fill(s), 2 1 4 3
POSSIBLE: 4 fill(s), 2 1 4 0 5 3
```

4. Από την πόλη $A = 4$ στην πόλη $B = 2$ με χωρητικότητα $C = 7$. Αυτό είναι αδύνατο γιατί όλες οι δρόμοι που οδηγούν στην πόλη 2 έχουν μήκος μεγαλύτερο των 7 χιλιομέτρων.

5. Από την πόλη $A = 1$ στην πόλη $B = 5$ με χωρητικότητα $C = 7$. Αυτό γίνεται ακολουθώντας τους μοναδικούς διαθέσιμους δρόμους με μήκος που δεν υπερβαίνει τα 7 χιλιόμετρα και γεμίζοντας στις πόλεις 1, 3 και 0.

Άσκηση 13 Δυαδικά Δένδρα

Προθεσμία υποβολής στον grader: 28/5/2021

Υλοποιήστε την κλάση `lexicon` για την αναπαράσταση λεξιλογίων κειμένων με τη μορφή δυαδικών δένδρων αναζήτησης. Η κλάση θα πρέπει να υποστηρίζει τις παρακάτω λειτουργίες, τις οποίες πρέπει να υλοποίησετε:

```
class lexicon {
public:
  lexicon();
  ~lexicon();

  void insert(const string &s);
  int  lookup(const string &s) const;
  int  depth(const string &s) const;
  void replace(const string &s1, const string &s2);

  friend ostream & operator << (ostream &out, const lexicon &l);
};
```

Θεωρήστε δεδομένο ότι οι λέξεις που θα αποθηκεύονται σε ένα λεξικό δεν θα είναι κενές και θα περιέχουν μόνο μικρά λατινικά γράμματα. Κάθε λέξη θα αντιστοιχεί σε έναν κόμβο στο δένδρο. Ο κόμβος θα περιέχει τη λέξη και τη συχνότητα εμφάνισής της στο λεξικό (δηλαδή έναν μετρητή που θα θυμάται πόσες φορές έχει εισαχθεί αυτή η λέξη). Το αριστερό παιδί του κόμβου θα περιέχει αλφαβητικά (λεξικογραφικά) μικρότερες λέξεις, ενώ το δεξιό παιδί αλφαβητικά (λεξικογραφικά) μεγαλύτερες. Καμία λέξη δεν θα πρέπει να εμφανίζεται σε περισσότερους από έναν κόμβους στο δένδρο. Επίσης, το δένδρο θα πρέπει να είναι ένα απλό BST: δεν απαιτείται και δεν πρέπει να είναι ισοζυγισμένο (π.χ., AVL).

Η μέθοδος `insert(s)` θα εισάγει τη λέξη `s` στο δένδρο.

Η μέθοδος `lookup(s)` θα αναζητά τη λέξη `s` στο δένδρο και θα επιστρέφει τη συχνότητα εμφάνισής. Το αποτέλεσμα θα είναι 0 (μηδέν) αν η λέξη δεν υπάρχει στο δένδρο.

Η μέθοδος `depth(s)` θα αναζητά τη λέξη `s` στο δένδρο, όπως και η `lookup`. Αν δεν υπάρχει, θα επιστρέφει -1. Αν όμως υπάρχει, θα επιστρέφει το *βάθος* στο οποίο βρίσκεται ο κόμβος που την περιέχει στο δένδρο, δηλαδή το μήκος του μονοπατιού από τη ρίζα έως αυτόν τον κόμβο. Θεωρούμε ότι η ρίζα του δένδρου βρίσκεται σε βάθος 0 (μηδέν).

Η μέθοδος `replace(s1, s2)` Θα αντικαθιστά όλες τις εμφανίσεις της λέξης `s1` με ισάριθμες εμφανίσεις της λέξης `s2`. Αν η `s1` δεν υπάρχει στο δένδρο, τότε δε θα γίνεται τίποτα. Αν υπάρχει και το πλήθος εμφανίσεών της είναι $k > 0$, τότε η `s1` θα διαγράφεται και θα αναζητάται η λέξη `s2`. Αν αυτή δεν υπάρχει, θα εισάγεται με συχνότητα εμφάνισης k. Αν όμως υπάρχει, τότε η συχνότητα εμφάνισής της θα ενημερώνεται κατάλληλα (θα αυξάνει κατά k).

Κατά τη διαγραφή μίας λέξης από το λεξικό (που γίνεται έμμεσα με τη μέθοδο `replace`), αν διαγράφεται κόμβος που έχει δύο μη κενά παιδιά, τότε ο κόμβος που διαγράφεται αντικαθίσταται από αυτόν που περιέχει την αμέσως μικρότερη λέξη. Αν διαγράφεται κόμβος που έχει ένα μη κενό παιδί, τότε αντικαθίσταται από το παιδί του.

Η εκτύπωση των λεξικών πρέπει να γίνεται σε αλφαβητική σειρά, με τις λέξεις να ακολουθούνται από τη συχνότητα εμφάνισής τους.

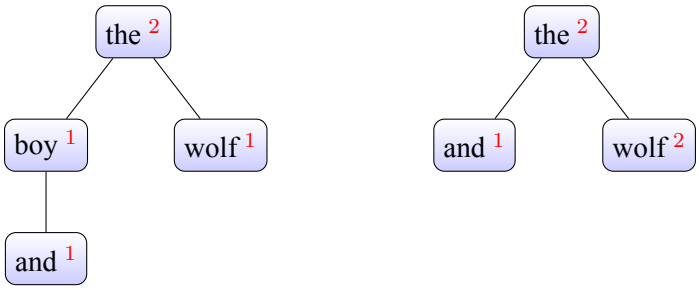
Μπορείτε να δοκιμάσετε την υλοποίησή σας με προγράμματα όπως το εξής:

```
int main() {
  lexicon l;
  l.insert("the");
  l.insert("boy");
  l.insert("and");
  l.insert("the");
  l.insert("wolf");
  cout << "The word 'the' is found " << l.lookup("the") << " time(s)" << endl;
  cout << "The word 'and' is found at depth " << l.depth("and") << endl;
  cout << l;
  l.replace("boy", "wolf");
  cout << "After replacement:\n";
  cout << l;
  cout << "Now the word 'and' is found at depth " << l.depth("and") << endl;
}
```

Η εκτέλεσή του θα πρέπει να εμφανίζει:

```
The word 'the' is found 2 time(s)
The word 'and' is found at depth 2
and 1
boy 1
the 2
wolf 1
After replacement:
and 1
the 2
wolf 2
Now the word 'and' is found at depth 1
```

Η μορφή του δένδρου πριν (αριστερά) και μετά (δεξιά) την κλήση της `replace` δίνεται παρακάτω.



Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τη δήλωση της κλάσης `lexicon` και τις υλοποιήσεις των μεθόδων της.