Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://helios.ntua.gr/course/view.php?id=869

Προγραμματιστικές Τεχνικές

Άσκηση 11 Μονοπάτια και κύκλοι Euler

Προθεσμία υποβολής στον grader: 11/6/2023

Δίνεται ένας μη κατευθυνόμενος γράφος N κορυφών (αριθμημένων από 0 έως $N-1$) και M ακμών. Θα είναι $2 \leq N \leq 10.000$ και $0 \leq M \leq 100.000$. Είναι πιθανό κάποια ακμή να υπάρχει πολλές φορές, όπως επίσης και να υπάρχουν κυκλικές ακμές — δηλαδή ακμές της μορφής (u, u). Θεωρήστε δεδομένο ότι ο γράφος θα είναι συνεκτικός, δηλαδή ότι θα είναι δυνατή η μετακίνηση από οποιαδήποτε κορυφή σε οποιαδήποτε άλλη, μέσω κάποιου μονοπατιού αποτελούμενου από ακμές.

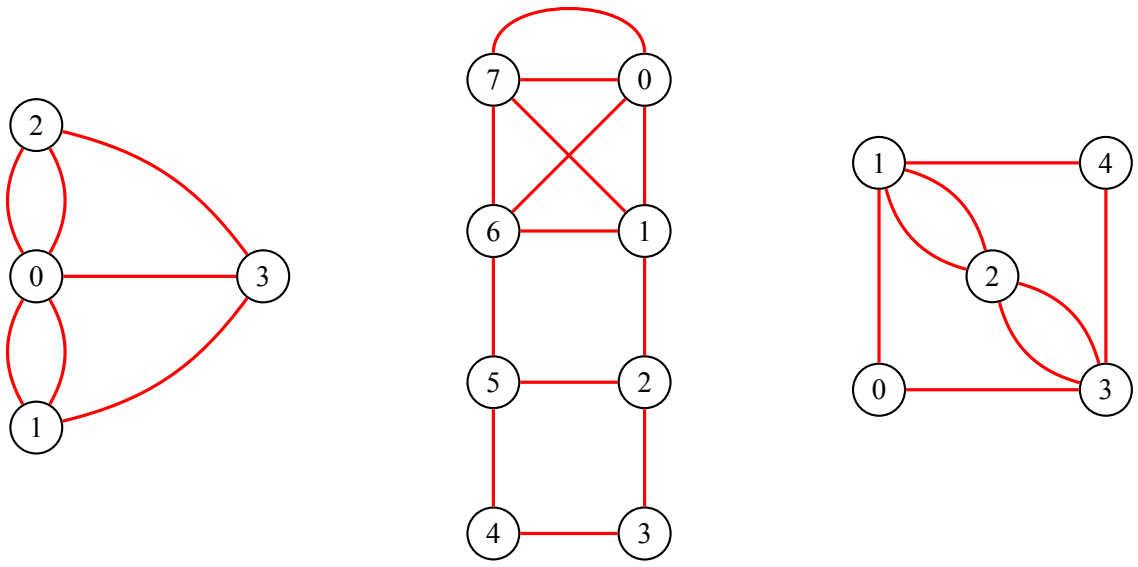
Γράψτε ένα πρόγραμμα που να διαβάζει τον γράφο από το standard input και να ελέγχει υπάρχει κύκλος ή μονοπάτι Euler στον γράφο. Συγκεκριμένα, το πρόγραμμά σας πρέπει να εκτυπώνει στο standard output μία γραμμή που να περιέχει ένα από τα εξής τρία:

- "`CYCLE`", αν υπάρχει κύκλος Euler.
- "`PATH u v`", αν δεν υπάρχει κύκλος Euler, υπάρχει όμως μονοπάτι Euler από την κορυφή u στην κορυφή v — φροντίστε να είναι $u < v$.
- "`IMPOSSIBLE`", αν δεν υπάρχει ούτε κύκλος ούτε μονοπάτι Euler.

Η είσοδος θα περιέχει τα εξής. Η πρώτη γραμμή θα περιέχει δύο αριθμούς, χωρισμένους μεταξύ τους με ένα κενό διάστημα: το πλήθος των κορυφών N και το πλήθος των ακμών M. Οι επόμενες M γραμμές της εισόδου θα περιγράφουν τις ακμές του γράφου. Κάθε μία θα περιέχει δύο αριθμούς, χωρισμένους μεταξύ τους με ένα κενό διάστημα: η γραμμή που περιέχει τους αριθμούς u και v θα παριστάνει μία ακμή μεταξύ των κορυφών u και v του γράφου.

Μπορείτε να προσαρμόσετε κατάλληλα τους γνωστούς τρόπους αναπαράστασης γράφων ώστε να καλύπτουν την περίπτωση γράφων με πολλαπλές και με κυκλικές ακμές.

Παραδείγματα γράφων:



Είσοδος:

```
4 7
0 1
0 1
0 2
0 2
3 0
3 1
3 2
```

```
8 13
0 7
0 7
0 1
0 6
1 7
7 6
6 1
1 2
6 5
2 5
2 3
3 4
5 4
```

```
5 8
0 1
1 2
0 3
1 4
1 2
2 3
3 2
4 3
```

Έξοδος:

```
IMPOSSIBLE
```

```
PATH 2 5
```

```
CYCLE
```

Άσκηση 12 Κύκλος σε κατευθυνόμενο γράφο

Προθεσμία υποβολής στον grader: 11/6/2023

Υλοποιήστε την κλάση `Graph` για την αναπαράσταση κατευθυνόμενων γράφων. Χάριν ευκολίας, θα θεωρήσουμε ότι ένας γράφος περιέχει V κορυφές αριθμημένες από 0 έως $V-1$. Η κλάση σας πρέπει να υποστηρίζει τις εξής λειτουργίες, τις οποίες πρέπει να υλοποιήσετε:

```
class Graph {
 public:
  Graph(int V);
  ~Graph();
  void addEdge(int u, int v);
};
```

Θεωρήστε ότι είναι πιθανό να κληθεί η `addEdge` για την προσθήκη περισσότερων ακμών με τα ίδια άκρα. Με εξαίρεση την πρώτη, τις υπόλοιπες μπορείτε να τις αγνοείτε. Θεωρήστε επίσης ότι δε θα υπάρχουν κυκλικές ακμές – δηλαδή ακμές της μορφής (u, u).

Στη συνέχεια, προσθέστε την παρακάτω μέθοδο στην κλάση σας:

```
  bool cycle(vector<int> &path) const;
```

Η μέθοδος αυτή πρέπει να επιστρέφει **`true`** αν υπάρχει κύκλος στο γράφο, διαφορετικά **`false`**. Στην πρώτη περίπτωση, η μέθοδος πρέπει να προσθέτει στο `path` τους κόμβους που απαρτίζουν τον κύκλο. Δηλαδή, αν στο γράφο υπάρχουν οι ακμές $1 \to 4$, $4 \to 2$, $2 \to 7$ και $7 \to 1$, τότε η μέθοδός σας μπορεί να προσθέσει στο `path` τις τιμές $1, 4, 2, 7$ με αυτή τη σειρά. Εξίσου καλά μπορεί να προσθέσει τις τιμές $2, 7, 1, 4$, ή οποιονδήποτε άλλον έγκυρο κύκλο βρει.

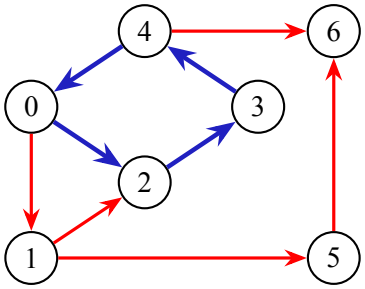
Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τη δήλωση της κλάσης `Graph` και τις υλοποιήσεις των μεθόδων της.

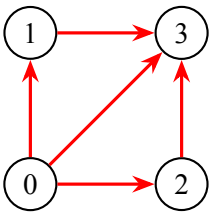
Μπορείτε να δοκιμάσετε την υλοποίησή σας με προγράμματα όπως το εξής (όπως ήδη γνωρίζετε από προηγούμενες ασκήσεις, μπορείτε να προσθέσετε τη συνάρτηση `main` ανάμεσα σε "`#ifndef CONTEST`" και "`#endif`", αν θέλετε να υποβάλλετε τον κώδικά σας στον grader χωρίς να τη σβήνετε):

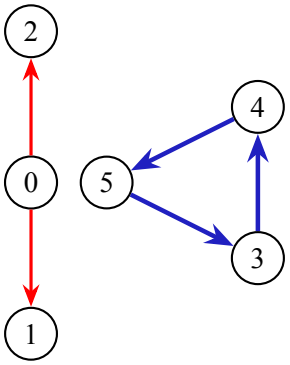
```
int main() {
  int V, E;
  cin >> V >> E;
  Graph g(V);
  for (int i = 0; i < E; ++i) { int u, v; cin >> u >> v; g.addEdge(u, v); }
  vector<int> path;
  bool c = g.cycle(path);
  if (c) {
    cout << "CYCLE: ";
    for (int i = 0; i < path.size(); ++i)
      cout << path[i] << (i == path.size()-1 ? "\n" : " ");
  } else {
    cout << "NO CYCLE" << endl;
  }
}
```

Η εκτέλεση του παραπάνω προγράμματος θα πρέπει να έχει την εξής συμπεριφορά, ανάλογα με την είσοδο που θα του δίνεται και που θα περιγράφει το γράφο:

Παραδείγματα γράφων:







Είσοδος:

```
7 9
0 1
0 2
1 2
2 3
3 4
4 0
4 6
1 5
5 6
```

```
4 5
0 1
0 2
1 3
2 3
0 3
```

```
6 5
0 1
0 2
3 4
4 5
5 3
```

Έξοδος:

```
CYCLE: 0 2 3 4
```

```
NO CYCLE
```

```
CYCLE: 3 4 5
```

Άσκηση 13 Ενώστε τα χωριά

Προθεσμία υποβολής στον grader: 11/6/2023

Μετά το 144ο μνημόνιο, οι περισσότεροι δρόμοι που συνέδεαν τα χωριά της Ελλάδας μεταξύ τους έχουν καταστραφεί. Η χώρα έχει N χωριά, αριθμημένα από 1 έως και N, και M βατούς δρόμους διπλής κατεύθυνσης που καθένας συνδέει δύο διαφορετικά χωριά. Δεν υπάρχουν περισσότεροι από ένας δρόμοι με άκρα στα ίδια χωριά. Αυτοί οι δρόμοι δεν εξασφαλίζουν την δυνατότητα οδικής σύνδεσης δυο οποιωνδήποτε χωριών. Αντίθετα, τα χωριά σχηματίζουν ανεξάρτητες ομάδες, που σε κάθε ομάδα δύο οποιαδήποτε χωριά συνδέονται οδικώς μεταξύ τους. Όμως, δεν υπάρχει οδική σύνδεση ανάμεσα σε χωριά που ανήκουν σε διαφορετικές ομάδες.

Το κράτος, με σύνθημα "έρχεται η επανασύνδεση", σχεδιάζει την ανασυγκρότηση του οδικού δικτύου. Όμως, λόγω περιορισμένων πόρων έχει την δυνατότητα κατασκευής το πολύ K νέων δρόμων, καθένας από τους οποίους θα συνδέει δυο χωριά για τα οποία δεν υπάρχει οδική σύνδεση. Μετά την κατασκευή αυτών των νέων δρόμων, αν έχει γίνει σωστή σχεδίαση, το πλήθος των ομάδων χωριών θα μειωθεί γιατί περισσότερα χωριά θα συνδέονται οδικώς μεταξύ τους.

Η άσκηση σας ζητάει να γράψετε ένα πρόγραμμα που να βρίσκει το ελάχιστο δυνατό πλήθος ανεξάρτητων ομάδων που μπορεί να μείνει μετά την κατασκευή των νέων δρόμων. Θα πρέπει να διαβάζει τους υπάρχοντες δρόμους από το standard input και να εκτυπώνει στο standard output μία γραμμή που να περιέχει έναν μόνο ακέραιο αριθμό: το ελάχιστο δυνατό πλήθος ομάδων.

Η είσοδος θα περιέχει τα εξής. Η πρώτη γραμμή περιέχει τρεις ακέραιους αριθμούς N, M, K, χωρισμένους ανά δύο με ένα κενό διάστημα: το πλήθος N ($1 \leq N \leq 1.000.000$) των χωριών, το πλήθος M ($1 \leq M \leq 2.000.000$) των υπαρχόντων δρόμων, και το πλήθος K ($0 \leq K \leq 1.000.000$) των νέων δρόμων που πρόκειται να κατασκευαστούν. Κάθε μία από τις επόμενες M γραμμές περιέχει δύο ακέραιους αριθμούς A και B, χωρισμένους με ένα κενό διάστημα, όπου $1 \leq A, B \leq N$ και $A \neq B$. Αυτό σημαίνει ότι υπάρχει ήδη, πριν την κατασκευή νέων δρόμων, ένας βατός δρόμος που συνδέει τα χωριά A και B. Θεωρήστε ότι τα χωριά είναι αριθμημένα από 1 μέχρι και N.

Παραδείγματα:

Είσοδος:

```
7 2 2
1 2
6 5
```

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

```
4 3 0
3 2
1 4
1 3
```

Έξοδος:

```
3
```

```
1
```

```
1
```

Για το 1ο παράδειγμα, αρχικά υπάρχουν πέντε ομάδες χωριών: $\{1, 2\}$, $\{3\}$, $\{4\}$, $\{5, 6\}$ και $\{7\}$. Αν κατασκευαστούν δύο νέοι δρόμοι, π.χ. μεταξύ των χωριών $(3, 7)$ και των χωριών $(3, 4)$, τότε θα προκύψουν μόνο τρεις ομάδες: $\{1, 2\}$, $\{3, 4, 7\}$ και $\{5, 6\}$. Αυτός είναι και ο ελάχιστος αριθμός ομάδων που μπορούν να προκύψουν μετά την κατασκευή δύο δρόμων. Στο 2ο παράδειγμα, αρκεί να κατασκευαστεί ένας νέος δρόμος για να μείνει μόνο μία ομάδα που να περιέχει όλα τα χωριά. Στο 3ο παράδειγμα, υπάρχει εξ αρχής μόνο μία ομάδα χωριών και αυτή θα παραμείνει, παρόλο που το κράτος δεν μπορεί να κατασκευάσει κανέναν δρόμο ($K = 0$).

Σημείωση: Για την άσκηση αυτή, θα σας είναι χρήσιμη η δομή union-find. Ξεκινήστε με N ομάδες αποτελούμενες από 1 χωριό κάθε μία και ενώστε, προσθέοντας έναν-έναν τους υπάρχοντες δρόμους. Μετρήστε πόσες ομάδες υπάρχουν αρχικά. Μετά, σκεφτείτε σαν μηχανικοί!

Άσκηση 14 Μας τέλειωσε η βενζίνη

Προθεσμία υποβολής στον grader: 11/6/2023

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;

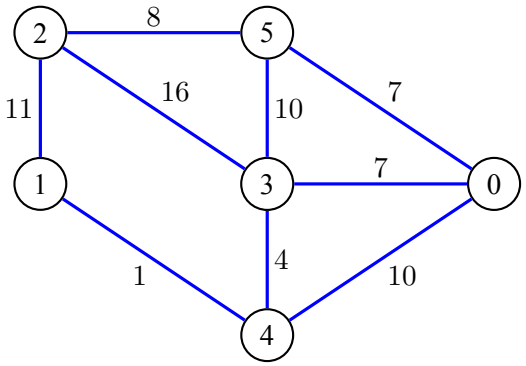
Είσοδος. Η πρώτη γραμμή της τυπικής εισόδου (`stdin`) θα περιέχει δύο ακέραιους αριθμούς N και M, χωρισμένους μεταξύ τους με ένα κενό διάστημα: το πλήθος των πόλεων και το πλήθος των δρόμων. Κάθε μία από τις επόμενες M γραμμές θα περιέχει τρεις ακέραιους αριθμούς, U, V και L, χωρισμένους ανά δύο με ένα 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:

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.

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
```

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.