

Task 1. Monopoly

Η Deni απολαμβάνει πολύ να παίζει το παιχνίδι monopoly. Αλλά η διάρκεια ενός παιχνιδιού είναι πολύ μεγάλη - από 5 έως 7 ώρες. Έτσι, η Deni αρχίζει να σκέφτεται να αλλάξει τους κλασικούς κανόνες. Οι παίκτες στη monopoly συνήθως κινούνται προς μία κατεύθυνση από θέση σε θέση και μετά από κάποιο χρονικό διάστημα επιστρέφουν στην αρχή, ξεκινούν ξανά και ούτω καθεξής. Στη νέα έκδοση, η κίνηση θα είναι και πάλι από θέση σε θέση, αλλά από την τρέχουσα θέση θα μπορούν να υπάρχουν πολλές επιλογές για την επόμενη κίνηση. Η Deni θέλει να βρει κατευθυνόμενες συνδέσεις μεταξύ των θέσεων, τέτοιες ώστε ένας παίκτης να μην μπορεί ποτέ να επιστρέψει σε μια θέση, όπου έχει βρεθεί, ανεξάρτητα από το πώς κινείται (φυσικά, ακολουθώντας τους κανόνες). Με αυτόν τον τρόπο το παιχνίδι δεν θα διαρκεί τόσο πολύ.

Έχει ήδη αρχίσει να φτιάχνει τον νέο πίνακα - έχει επιλέξει τον αριθμό των θέσεων N (οι θέσεις αριθμούνται από το 1 έως το N) και έχει φτιάξει μια λίστα με M συνδέσεις (κάθε σύνδεση έχει μια κατεύθυνση και δεν υπάρχει καμία σύνδεση που να συνδέει μια θέση με τον εαυτό της). Αν η θέση i συνδέεται με τη θέση j , τότε δεν υπάρχει καμία άμεση σύνδεση προς την αντίθετη κατεύθυνση, δηλαδή από τη θέση j στη θέση i , και επίσης, δεν υπάρχουν άλλες άμεσες συνδέσεις από τη θέση i στη θέση j . Η Deni νόμιζε ότι ήταν έτοιμη με τον νέο πίνακα, αλλά ξαφνικά παρατήρησε ότι η συνθήκη που θέλει (όταν μετακινείσαι από θέση σε θέση, χρησιμοποιώντας τις συνδέσεις, δεν μπορείς να επιστρέψεις σε μια θέση που έχεις επισκεφτεί προηγουμένως) δεν ισχύει για τη λίστα των συνδέσεών της. Αρχικά, σκέφτηκε να αφαιρέσει κάποιες από τις άμεσες συνδέσεις αλλά αυτό θα έχει ως αποτέλεσμα να ξαναγράψει τη λίστα, η οποία μπορεί να είναι πραγματικά μεγάλη. Γι' αυτό η Deni αποφάσισε να αντιστρέψει την κατεύθυνση ορισμένων από τις κατευθυνόμενες συνδέσεις.

Πρόβλημα

Παίζετε τακτικά monopoly με την Deni. Γι' αυτό θέλετε να τη βοηθήσετε γράφοντας το πρόγραμμα **monopoly**, το οποίο πρέπει να της πει ποιες συνδέσεις πρέπει να αντιστραφούν ώστε να ισχύει η ζητούμενη συνθήκη. Το πρόγραμμα πρέπει να περιέχει τη συνάρτηση `find_reverse` η οποία θα μεταγλωττιστεί με το πρόγραμμα της κριτικής επιτροπής.

Λεπτομέρειες υλοποίησης

Η συνάρτηση `find_reverse` πρέπει να έχει το ακόλουθο πρότυπο:

```
std::string find_reverse (int N, int M, int connections[][2]);
```

Καλείται μόνο μία φορά από το πρόγραμμα της κριτικής επιτροπής με τρεις παραμέτρους: N – ο αριθμός των θέσεων στον νέο πίνακα, M – ο αριθμός των κατευθυνόμενων συνδέσεων στη λίστα της Deni και ο πίνακας `connections` που έχει M γραμμές, καθεμία από τις οποίες αποτελείται από δύο αριθμούς x και y – τις θέσεις αρχής και τέλους για την κατευθυνόμενη σύνδεση. Η συνάρτηση αυτή θα πρέπει να επιστρέφει μια δυαδική συμβολοσειρά μήκους M – με τη σειρά του πίνακα `connections`, για κάθε σύνδεση θα πρέπει να βάζετε '1' στη συμβολοσειρά, αν η σύνδεση πρέπει να αντιστραφεί και '0', διαφορετικά. Εάν υπάρχουν περισσότερες από μία λύσεις, μπορείτε να επιστρέψετε οποιαδήποτε από αυτές.

Πρέπει να υποβάλετε στο σύστημα το αρχείο **monopoly.cpp**, το οποίο περιέχει την υλοποίηση της συνάρτησης `find_reverse`. Το αρχείο μπορεί να περιέχει άλλες συναρτήσεις και κώδικα, που απαιτούνται για το πρόγραμμά σας, αλλά **δεν πρέπει να περιέχει** την κύρια συνάρτηση – `main`. Επίσης, δεν πρέπει να προσπαθήσετε

να διαβάσετε από την τυπική είσοδο (standard input) ούτε να προσπαθήσετε να γράψετε στην τυπική έξοδο (standard output)!

Περιορισμοί

- ♣ $3 \leq N \leq 5 \times 10^5$
- ♣ $3 \leq M \leq 1.5 \times 10^6$

Υποπροβλήματα

Υποπρόβλημα	Πόντοι	N	M	Πρόσθετοι περιορισμοί
1	0	–	–	Το παράδειγμα
2	15	≤ 7	≤ 21	–
3	40	$\leq 10^3$	$\leq 5 \times 10^3$	–
4	25	$\leq 10^5$	$\leq 5 \times 10^5$	–
5	20	$\leq 5 \times 10^5$	$\leq 1.5 \times 10^6$	–

Για να λάβετε τους πόντους για ένα συγκεκριμένο υποπρόβλημα, η λύση σας πρέπει να περάσει όλα τα tests του υποπροβλήματος.

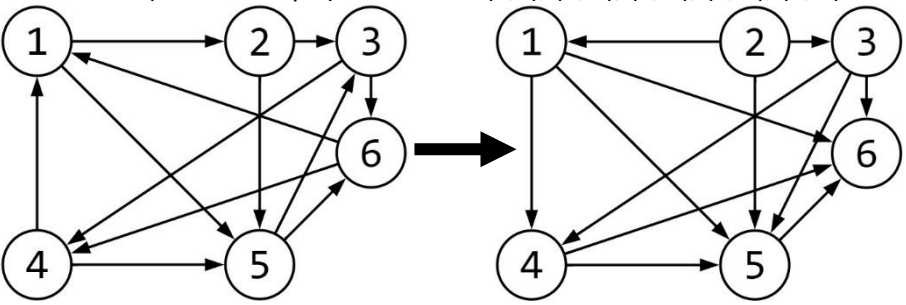
Τοπικές δοκιμές

Για τοπικές δοκιμές σας δίνεται το αρχείο **Lgrader.cpp**. Πρέπει να το τοποθετήσετε στον ίδιο φάκελο με το πρόγραμμα **monopoly.cpp** και να μεταγλωττίσετε μαζί τα αρχεία **Lgrader.cpp** και **monopoly.cpp**. Με αυτόν τον τρόπο, θα φτιάξετε ένα πρόγραμμα για να ελέγξετε την ορθότητα της συνάρτησής σας. Το πρόγραμμα θα απαιτεί την ακόλουθη ακολουθία δεδομένων από την τυπική είσοδο:

- στην πρώτη γραμμή: δύο ακέραιοι αριθμοί – ο αριθμός των θέσεων N και ο αριθμός των συνδέσεων M του νέου πίνακα.
- στις τελευταίες M γραμμές: σε κάθε γραμμή πρέπει να υπάρχουν δύο ακέραιοι x και y – οι θέσεις αρχής και τέλους για κάθε κατευθυνόμενη σύνδεση.

Η έξοδος του προγράμματος θα είναι η δυαδική συμβολοσειρά που βρήκατε.

Παράδειγμα τοπικής δοκιμής

Είσοδος	Έξοδος	Επεξήγηση
6 12 1 2 1 5 2 3 2 5 3 4 3 6 4 1 4 5 5 3 5 6 6 1 6 4	100000101011	<p>Οι συνδέσεις που αντιστρέφονται είναι: (1,2), (4, 1), (5, 3), (6, 1), (6, 4).</p>  <p>Το παραπάνω σχήμα δείχνει τις διαδρομές, πρώτον, με τις κατευθυνόμενες συνδέσεις από τη λίστα της Deni και, δεύτερον, μετά την αντιστροφή της κατεύθυνσης των συνδέσεων με το αντίστοιχο '1' στην έξοδο. Είναι σαφές ότι στην αρχή χρησιμοποιώντας τις κατευθυνόμενες συνδέσεις (1, 2), (2, 3), (3, 4) και (4, 1) μπορούμε να ξεκινήσουμε από τη θέση 1 και κινούμενοι κατά μήκος αυτών των κατευθυνόμενων συνδέσεων, θα επιστρέψουμε σε αυτή τη θέση. Μπορεί κανείς να δει ότι η συνθήκη της Deni ισχύει στον πίνακα που προκύπτει. Σημειώστε ότι υπάρχουν και άλλες έγκυρες λύσεις:</p> <p>(4, 1), (5, 3), (6, 1), (6, 4) (1, 5), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (5, 6)</p>