

Reconstruct

Χρονικό όριο: 4 δευτερόλεπτα

Όριο μνήμης: 1024 MiB

Αυτό το πρόβλημα είναι διαδραστικό.

Η Μπίσσυ είναι έτοιμη να συμμετάσχει στο παιχνίδι πλοήγησης "Τελικός προορισμός" που οργανώνεται από εθελοντές, όπου οι αποστολές της EJOI θα ανταγωνίζονται για να επισκεφτούν διαφορετικές τοποθεσίες σε όλη την πόλη. Όμως, η Μπίσσυ ενδιαφέρεται περισσότερο για το πώς σχεδιάστηκε το παιχνίδι παρά για το να το κερδίσει.

Υπάρχουν N τοποθεσίες στον χάρτη και N ομάδες. Κάθε ομάδα πρέπει να επισκεφτεί όλες τις τοποθεσίες και να έχει μοναδική αρχική τοποθεσία — η ομάδα i ξεκινάει από την τοποθεσία i . Σε κάθε ομάδα δίνεται η δική της διαδρομή.

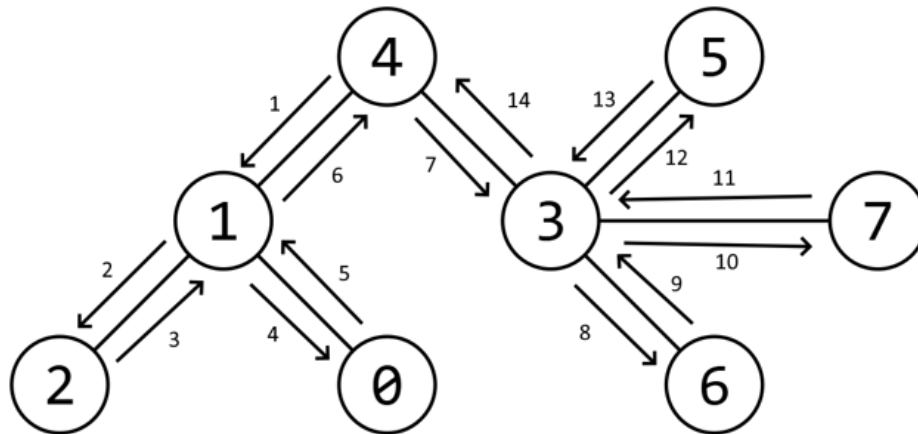
Η Μπίσσυ έχει επίσης κάποιες εσωτερικές πληροφορίες — οι διαδρομές είναι βασισμένες σε μια κρυμμένη δομή γρήγορων δημοσίων συγκοινωνιών. Πιο συγκεκριμένα, η επιτροπή του διαγωνισμού επέλεξε $N - 1$ γραμμές, καθεμία από τις οποίες συνδέει ένα ζεύγος από τοποθεσίες, έτσι ώστε κάθε τοποθεσία να είναι προσβάσιμη από οποιαδήποτε άλλη χρησιμοποιώντας αυτές τις γραμμές. Μία τέτοια δομή ονομάζεται επίσης δεντρική. Στη συνέχεια, για κάθε αρχική τοποθεσία i , η επιτροπή δημιούργησε έναν **αυθαίρετο περίπατο DFS** (ορίζεται παρακάτω) και ανέθεσε αυτόν τον περίπατο ως τη διαδρομή της ομάδας i .

Η Μπίσσυ θέλει να ανακαλύψει το δέντρο δημοσίων συγκοινωνιών που επέλεξε η επιτροπή. Μπορεί να κάνει μόνο γρήγορες ερωτήσεις της μορφής:

- Ποιος είναι ο j -οστός προορισμός της i -οστής ομάδας;

Βοηθήστε την Μπίσσυ γράφοντας ένα πρόγραμμα `find_tree` το οποίο βρίσκει το κρυμμένο δέντρο.

Ένας *περίπατος DFS* ενός δέντρου είναι η σειρά με την οποία μία διαδικασία αναζήτησης κατά βάθος επισκέπτεται για πρώτη φορά τις κορυφές του. Μία τέτοια διαδικασία ξεκινάει από μία δεδομένη κορυφή v και πηγαίνει αναδρομικά σε έναν από τους ανεπίσκεπτους γείτονές της μέχρι να μην υπάρχει τέτοιος γείτονας — μετά επιστρέφει στην προηγουμένως επισκεφθείσα κορυφή και συνεχίζει από εκεί. Παράδειγμα:



Εδώ $v = 4$ και τα βέλη κατά μήκος των ακμών αντιστοιχούν στα βήματα μίας αναζήτησης κατά βάθος. Ο περίπατος που παράγεται είναι $[4, 1, 2, 0, 3, 6, 7, 5]$. Να σημειωθεί ότι η σειρά με την οποία επισκεπτόμαστε τους γείτονες έχει σημασία. Ένας άλλος περίπατος θα μπορούσε να είναι $[4, 3, 5, 7, 6, 1, 0, 2]$.

Η δομή του δέντρου και οι N περίπατοι DFS αποφασίζονται πριν ξεκινήσει το πρόγραμμά σας και δεν αλλάζουν ως αποτέλεσμα των ερωτημάτων σας. Οι σειρές των γειτόνων των κορυφών που χρησιμοποιούνται κατά τη διάρκεια τέτοιων αναζητήσεων κατά βάθος μπορεί να είναι διαφορετικές.

Λεπτομέρειες υλοποίησης

Πρέπει να υλοποιήσετε την παρακάτω συνάρτηση:

```
std::vector<std::pair<int, int>> find_tree(int N)
```

- εδώ το N είναι το πλήθος τοποθεσιών,
- η επιστρεφόμενη τιμή είναι μία λίστα των ακμών του δέντρου — η σειρά των ακμών και των άκρων τους δεν έχει σημασία.

Για κάθε περίπτωση ελέγχου, η συνάρτηση μπορεί να κληθεί το πολύ T φορές.

Για να αλληλεπιδράσετε με τον κριτή, μπορείτε να κάνετε κλήσεις στην παρακάτω συνάρτηση:

```
int guess(int i, int j)
```

Επιστρέφει την j -οστή τοποθεσία στον i -οστό περίπατο του δέντρου. Να σημειωθεί ότι το `guess(i, 0)` επιστρέφει i . Μπορείτε να υποθέσετε ότι η συνάρτηση ανταποκρίνεται σε σταθερό χρόνο $O(1)$ για όλα τα υποπροβλήματα μέχρι και το 6, και σε λογαριθμικό χρόνο $O(\log N)$ για το υποπρόβλημα 7. Πρέπει να ισχύει ότι $0 \leq i, j \leq N - 1$, διαφορετικά η λύση σας θα λάβει την ένδειξη `Output isn't correct: Invalid call` (που σημαίνει Η έξοδος δεν είναι σωστή: Μη έγκυρη κλήση).

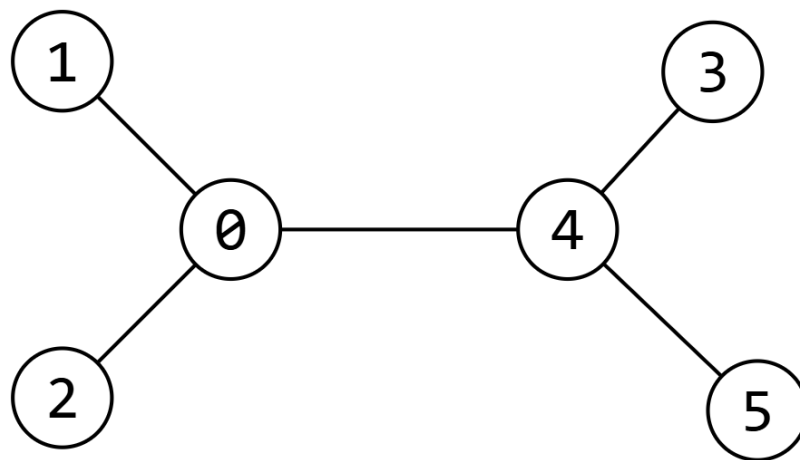
Περιορισμοί

- $2 \leq N \leq 2^{16} + 1$
- Το T ακολουθεί περιορισμούς που βασίζονται στο μέγιστο N (που συμβολίζεται ως N_{max}) μεταξύ όλων των κλήσεων συναρτήσεων σε μία περίπτωση ελέγχου:
 - αν $N_{max} \leq 9$, τότε $1 \leq T \leq 100$
 - αν $N_{max} \leq 2^{10} + 1$, τότε $1 \leq T \leq 10$
 - αν $N_{max} \leq 2^{16} + 1$, τότε $1 \leq T \leq 3$

Ο βαθμολογητής του συστήματος μπορεί να χρησιμοποιήσει μέχρι και 280 MiB, τα οποία προσμετρούνται ως μνήμη της λύσης σας.

Παράδειγμα

Έστω ότι η κρυμμένη δομή των γραμμών γρήγορων δημοσίων συγκοινωνιών είναι η εξής:



Υποθέστε ότι για την αρχική τοποθεσία 0, ο περίπατος είναι $[0, 1, 2, 4, 3, 5]$. Τότε, τα παρακάτω είναι ένα παράδειγμα αλληλεπίδρασης:

Πρόγραμμα διαγωνιζόμενου	Πρόγραμμα κριτή
	<code>find_tree(6)</code>
<code>guess(0, 0)</code>	0
<code>guess(0, 1)</code>	1
<code>guess(0, 2)</code>	2
<code>guess(0, 3)</code>	4
<code>guess(0, 4)</code>	3
<code>guess(0, 5)</code>	5
<code>return {{0,1},{0,2},{4,0},{5,4},{3,4}};</code>	

Να σημειωθεί ότι η πληροφορία από τις απαντήσεις των ερωτημάτων δεν είναι επαρκής για να προσδιοριστεί μοναδικά το δέντρο, όμως αυτή είναι στην πραγματικότητα η απάντηση στην περίπτωση ελέγχου του υποπροβλήματος 0.

Υποπροβλήματα

Υποπρόβλημα	Πόντοι	N	Περαιτέρω περιορισμοί
0	0	-	Το παράδειγμα.
1	11	≤ 9	-
2	6	≤ 100	Κάθε κορυφή είναι συνδεδεμένη με το πολύ 2 άλλες.
3	13	≤ 100	Κάθε περίπατος δημιουργείται από μία αναζήτηση κατά βάθος η οποία σε κάθε βήμα δίνει προτεραιότητα στο να απομακρύνεται από την κορυφή 0. Αν υπάρχουν πολλές επιλογές για να απομακρυνθούμε από την κορυφή 0, μία από αυτές επιλέγεται αυθαίρετα.
4	11	≤ 100	Κάθε κορυφή διαφορετική της κορυφής 0 είναι συνδεδεμένη με το πολύ 2 άλλες.
5	10	≤ 100	-
6	31	$\leq 2^{10} + 1$	-
7	18	$\leq 2^{16} + 1$	-

Βαθμολόγηση

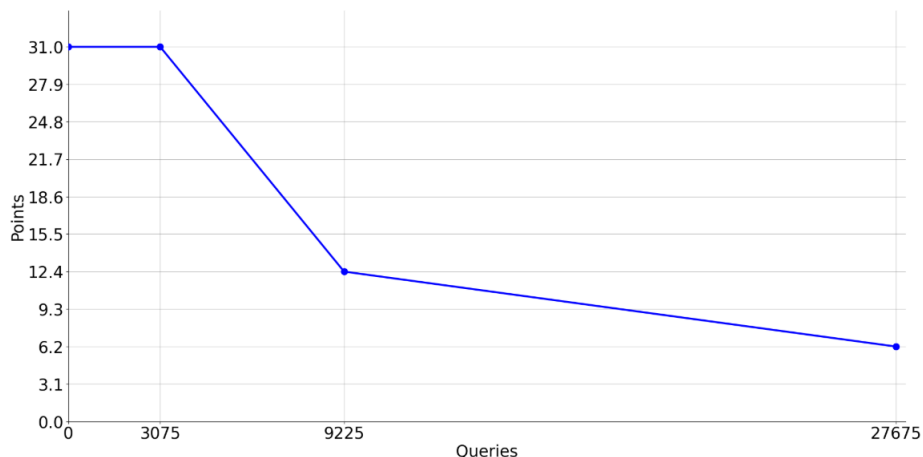
Για κάθε υποπρόβλημα μέχρι και το υποπρόβλημα 5, λαμβάνετε πλήρως τους πόντους του αν βρείτε με επιτυχία το δέντρο εντός των περιορισμών χρονικών ορίων. Για τα υποπροβλήματα 6 και 7, το ποσοστό των συνολικών πόντων S για μία δεδομένη περίπτωση ελέγχου εξαρτάται από το **μέγιστο** πλήθος ερωτημάτων που κάνετε σε κάποιο από τα σενάρια της, Q_{max} . Η βαθμολόγηση για αυτά τα υποπροβλήματα λειτουργεί ως εξής:

- αν $Q_{max} \leq L_1$, τότε $S = 1.0$;
- αν $L_1 < Q_{max} \leq L_2$, τότε $S = 0.4 + 0.6 \times \frac{L_2 - Q_{max}}{L_2 - L_1}$;
- αν $L_2 < Q_{max} \leq 3L_2$, τότε $S = 0.2 + 0.2 \times \frac{3L_2 - Q_{max}}{2L_2}$;
- αν $3L_2 < Q_{max}$, τότε $S = 0.2$;

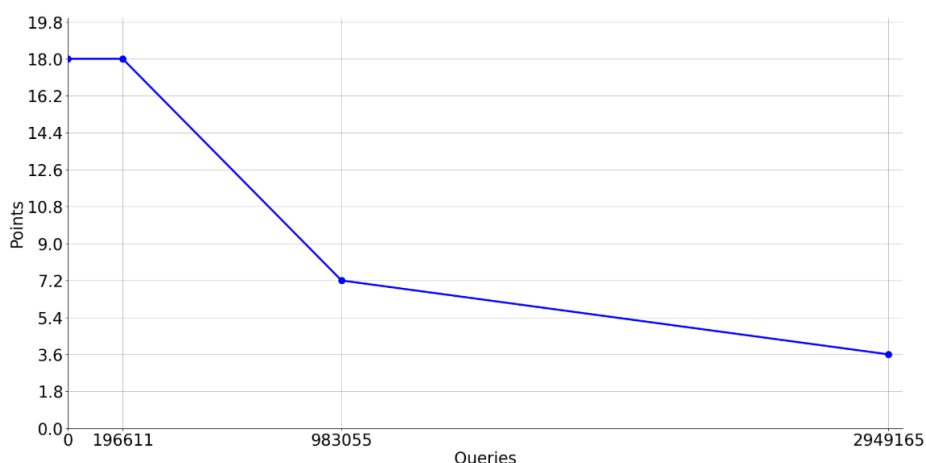
όπου $L_1 = 3 \times (2^{10} + 1) = 3075$ και $L_2 = 9 \times (2^{10} + 1) = 9225$ για το υποπρόβλημα 6, και $L_1 = 3 \times (2^{16} + 1) = 196611$ και $L_2 = 15 \times (2^{16} + 1) = 983055$ για το υποπρόβλημα 7. Το ποσοστό πόντων που λαμβάνετε από ολόκληρο το υποπρόβλημα είναι το ελάχιστο S μεταξύ όλων των περιπτώσεων ελέγχου του.

Τα παρακάτω γραφήματα δείχνουν την συνάρτηση βαθμολόγησης για τα τελευταία δύο υποπροβλήματα:

Υποπρόβλημα 6



Υποπρόβλημα 7



Ενδεικτικός βαθμολογητής

Υπάρχουν δύο ενδεικτικοί βαθμολογητές που παρέχονται.

Για τοπικές δοκιμές: το `Lgrader.cpp` μπορεί να μεταγλωττιστεί μαζί με το πρόγραμμά σας για να παρέχει ένα εργαλείο για δοκιμές. Το πρόγραμμα διαβάζει το πλήθος T των σεναρίων και για καθένα αναμένει το πλήθος των τοποθεσιών N ακολουθούμενο από $N - 1$ γραμμές, καθεμία με ένα ζεύγος ακεραίων που περιγράφει μία άμεση γραμμή. Έπειτα, N γραμμές από N ακεραίους ακολουθούν – οι περίπατοι DFS. Να σημειωθεί ότι ο περίπατος i πρέπει να ξεκινάει από την κορυφή i . Μπορείτε να αφήσετε τον βαθμολογητή να παράγει περίπατους για εσάς θέτοντας την σταθερά `AUTO_GENERATE` σε `true`. Ως έξοδο ο βαθμολογητής αναφέρει είτε το μήνυμα σφάλματος αν το αποτέλεσμα είναι λανθασμένο, είτε το πλήθος ερωτημάτων που γίνονται για κάθε σενάριο και το συνολικό μέγιστο. Επιπλέον, να σημειωθεί ότι ο βαθμολογητής δε δουλεύει

για επαρκώς μεγάλα N (πιο συγκεκριμένα για τους περιορισμούς του υποπροβλήματος 7), είστε ελεύθεροι να χρησιμοποιήσετε την φαντασία σας για αυτήν την περίπτωση.

Για δοκιμές χρήστη στο σύστημα: το `stub.cpp` μπορεί να χρησιμοποιηθεί σε συνδυασμό με το πρόγραμμά σας για τις δοκιμές χρήστη (user tests) στο σύστημα. Η μορφή εισόδου είναι η ίδια με αυτήν του άλλου βαθμολογητή. Να σημειωθεί ότι για τις δοκιμές χρήστη δεν υπάρχει ενσωματωμένη επιλογή για αυτόματη παραγωγή περιπάτων.