Task Navigation

⏳ 3 sec. 💾 512 MB

ملاحظة: الترجمة تشمل المسائل الفرعية، طريقة احتساب الدرجات، وتفاصيل كتابة الكود.

توجد **شبكة بسيطة غير متجهة متصلة على شكل صبار**[1] تحتوي على عدد N مكان و عدد M طريق. لكل مكان لون (مشار إليه بعدد صحيح غير سالب من 0 إلى 1499). في البداية، يكون لون جميع الاماكن 0. يستكشف روبوت حتمي عديم الذاكرة[2] الشبكة بالانتقال من مكان إلى آخر عبر الطرق. يجب عليه زيارة جميع الاماكن مرة واحدة على الأقل ثم إنهاء المهمة.

يبدأ الروبوت عند مكان ما، والذي قد يكون أيا من الاماكن في الشبكة. في كل خطوة، يرى لون مكانه الحالي وألوان جميع الاماكن المجاورة (بترتيب ثابت للمكان الحالي) (أي أن إعادة زيارة المكان ستمنح الروبوت نفس تسلسل الاماكن المجاورة، حتى لو كانت ألوانها مختلفة عن ذي قبل). يقوم الروبوت بأحد الإجراءين التاليين:

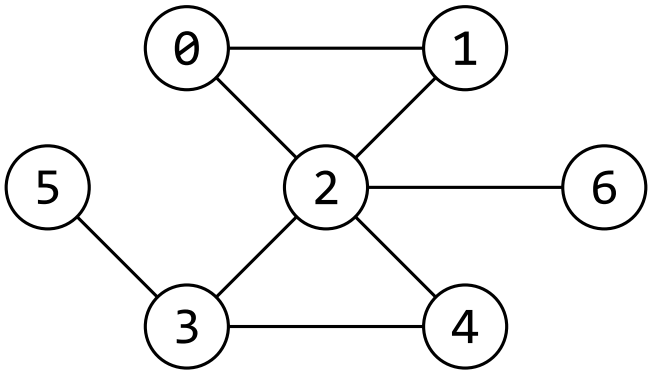
١. يقرر إنهاء المهمة.

٢. يختار لونا جديدا (أو ربما لونا مطابقا) للمكان الحالي، ومكان مجاور للتحرك اليه. يُحدد المكان المجاور برقم من 0 إلى $D-1$، حيث D هو عدد الاماكن المجاورة.

في الحالة الثانية، يعاد تلوين المكان الحالي (أو ربما يبقى لونه كما هو) وينتقل الروبوت إلى المكان المجاور المختار. يتكرر هذا حتى ينتهي الروبوت أو حتى يصل إلى حد التكرار. يفوز الروبوت إذا زار جميع العقد، ثم أنهى المهمة ضمن حد التكرار $L = 3000$ خطوة (وإلا سيخسر).

يجب عليك تصميم استراتيجية للروبوت لحل المسألة على أي شبكة من نوع الصبار. بالإضافة إلى ذلك، حاول تقليل عدد الألوان المختلفة التي يستخدمها الحل. هنا، يُحتسب اللون 0 دائمًا على أنه لون مستخدم.

[1] شبكة الصبار البسيطة غير الموجهة المتصلة هي شبكة بسيطة غير موجهة متصلة (يمكن الوصول إلى كل مكان من أي مكان آخر؛ الطرق ثنائية الاتجاه؛ لا تحتوي على طرق من مكان الى نفسه أو اكثر من طريق مباشر بين مكانين مختلفين)، حيث ينتمي كل طريق إلى دورة بسيطة واحدة على الأكثر (الدورة البسيطة هي دورة تحتوي على كل مكان مرة واحدة على الأكثر). الصورة أدناه مثال.



[2] الروبوت حتمي ولا يحتاج إلى ذاكرة، إذا كان عمله يعتمد فقط على مدخلاته الحالية (أي أنه لا يخزن أي بيانات من خطوة إلى أخرى)، ويختار دائما نفس العمل عند إعطائه نفس المدخلات.

Implementation details

The robot's strategy should be implemented as the following function:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

تستقبل الدالة لون المكان الحالي وألوان جميع الاماكن المجاورة (بالترتيب). يجب أن ترجع الدالة زوجا، يكون عنصره الأول هو اللون الجديد للمكان الحالي، وعنصره الثاني هو رقم المكان المجاور الذي يجب أن ينتقل إليه الروبوت. إذا انتهى الروبوت، فيجب أن ترجع الدالة الزوج $(-1 - 1)$.

ستُستدعى هذه الدالة بشكل متكرر لاختيار إجراءات الروبوت. ولأنها حتمية، فإذا استُدعيت دالة $navigate$ بالفعل مع بعض المدخلات، فلن تُستدعى بنفس هذه المدخلات مرة أخرى؛ بل ستُعاد قيمة إرجاعها السابقة. بالإضافة إلى ذلك، قد يحتوي كل اختبار على $T \leq 5$ اختبارات فرعية (شبكات مميزة و/أو مواقع بداية)، ويمكن تشغيلها بالتزامن (أي أن برنامجك قد يتلقى استدعاءات متناوبة لاختبارات فرعية مختلفة). أخيرًا، قد تحدث استدعاءات دالة التنقل في عمليات تنفيذ منفصلة لبرنامجك (ولكن قد تحدث أحيانًا في نفس التنفيذ). إجمالي عدد عمليات التنفيذ لبرنامجك هو $P = 100$. لذلك، يجب ألا يحاول برنامجك تمرير المعلومات بين استدعاءات مختلفة.

Constraints

- $3 \leq N \leq 1000$
- $0 \leq \text{Color} < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$

Scoring

النسبة S من النقاط التي تحسب لك تعتمد على C -- أكبر عدد ألوان مختلفة يستعملها حلك (شاملا ال 0) في أي تجربة في المسألة الجزئية الحالية او أي مسألة جزئية ملزومة:

- إذا فشل حلك في أي تجربة، إذن $S = 0$.
- إذا $C \leq 4$، إذن $S = 1.0$.
- إذا $4 < C \leq 8$، إذن $S = 1.0 - 0.6\frac{C-4}{4}$.
- إذا $8 < C \leq 21$، إذن $S = 0.4\frac{8}{C}$.
- إذا $C > 21$، إذن $S = 0.15$.

Subtasks

Subtask	Points	Required subtasks	N	Additional constraints
0	0	–	≤ 300	المثال المعطى.
1	6	–	≤ 300	الشبكة عبارة عن دائرة.[1] [1]
2	7	–	≤ 300	الشبكة عبارة عن نجمة.[2]
3	9	–	≤ 300	الشبكة عبارة عن طريق.[3]
4	16	$2-3$	≤ 300	الشبكة عبارة عن شجرة.[4]
5	27	–	≤ 300	جميع الاماكن لديها ثلاثة طرق متصلة بها على الاكثر والنقطة التي سيبدا بها الروبوت لديه طريق وحيد متصل به.
6	28	$0-5$	≤ 300	–
7	7	$0-6$	–	–

[1] الدائرة هي شبكة بحيث طرقها تربط: $(i, (i+1) \bmod N)$ for $0 \leq i < N$.
[2] النجمة هي شبكة بحيث طرقها تربط: $(0, i)$ for $1 \leq i < N$.
[3] الطريق هو شبكة بحيث طرقها تربط: $(i, i+1)$ for $0 \leq i < N-1$.
[4] الشجرة هي شبكة بحيث يوجد طريق بسيط وحيد بين اي مكانين.

Example

Consider the sample graph from the image in the statement, which has $N = 7$, $M = 8$ and edges $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ and $(2, 6)$. Additionally, since the orders of the elements in the nodes' adjacency lists are relevant, we give them in this table:

Node	Adjacent nodes
0	$2, 1$
1	$2, 0$
2	$0, 3, 4, 6, 1$
3	$4, 5, 2$
4	$2, 3$
5	3
6	2

Suppose the robot starts at node 5. Then the following is one possible (unsuccessful) sequence of interactions:

#	Colors	Node	Call to `navigate`	Return value
1	$0, 0, 0, 0, 0, 0, 0$	5	`navigate(0, {0})`	`{1, 0}`
2	$0, 0, 0, 0, 0, 1, 0$	3	`navigate(0, {0, 1, 0})`	`{4, 2}`
3	$0, 0, 0, 4, 0, 1, 0$	2	`navigate(0, {0, 4, 0, 0, 0})`	`{0, 3}`
4	$0, 0, 0, 4, 0, 1, 0$	6	[1]`navigate(0, {0})`	`{1, 0}`
5	$0, 0, 0, 4, 0, 1, 1$	2	`navigate(0, {0, 4, 0, 1, 0})`	`{8, 0}`
6	$0, 0, 8, 4, 0, 1, 1$	0	`navigate(0, {8, 0})`	`{3, 0}`
7	$3, 0, 8, 4, 0, 1, 1$	2	`navigate(8, {3, 4, 0, 1, 0})`	`{2, 2}`
8	$3, 0, 2, 4, 0, 1, 1$	4	`navigate(0, {2, 4})`	`{1, 1}`
9	$3, 0, 2, 4, 1, 1, 1$	3	`navigate(4, {1, 1, 2})`	`{-1, -1}`

Here the robot used a total of 6 distinct colors: 0, 1, 2, 3, 4 and 8 (note that 0 would have counted as used even if the robot never returned color 0, since all nodes start in color 0). The robot ran for 9 iterations before terminating. However, it failed since it terminated without having visited node 1.

[1]Note the call to `navigate` at iteration 4 would not actually happen. This is because it is equivalent to the call at iteration 1, so the grader would simply reuse the return value of your function from that call. However, this still counts as an iteration of the robot.

Sample grader

The sample grader does not run multiple executions of your program, so all calls to `navigate` will be in the same execution of your program.

The input format is the following: First T (the number of subtests) is read. Then for each subtest:

- line 1: two integers – N and M;
- line $2+i$ (for $0 \leq i < M$): two integers – A_i and B_i, which are the two nodes that edge i connects ($0 \leq A_i, B_i < N$).

The sample grader will then print out the number of distinct colors your solution used and the number of iterations it needed before it terminated. Alternatively, it will print out an error message, if your solution failed.

By default, the sample grader prints detailed information on what the robot sees and does at each iteration. You can disable this, by changing the value of `DEBUG` from `true` to `false`.