



宇宙怪盗 (Space Thief)

あなたは JOI 銀河で怪盗として活躍している。

JOI 銀河には 0 から $N-1$ までの番号が付けられている N 個の星がある。また、 0 から $M-1$ までの番号が付けられている M 個のワープ装置がある。ワープ装置 i ($0 \leq i \leq M-1$) は星 U_i と星 V_i の間を双方向に結んでいる。どの 2 つの星の間も、いくつかのワープ装置を用いることで移動できる。

JOI 銀河のある星には鍵が隠されており、別の星には宝箱が隠されている。あなたのミッションは、鍵と宝箱が隠されている星の番号を特定することである。そのために、以下の質問を 300 回まで行うことができる。

- それぞれのワープ装置 i ($0 \leq i \leq M-1$) について、星 U_i から星 V_i へ一方方向のみに結ぶようにするか星 V_i から星 U_i へ一方方向のみに結ぶようにするかを選択する。
- そのもとで、いくつかのワープ装置を用いて鍵のある星から宝箱のある星へ移動できるかを尋ねる。

あなたは質問を行うことで、鍵が隠された星の番号である A と宝箱が隠された星の番号である B を特定したい。ミッションで高い評価を得るために、質問の回数はできるだけ少なくしたい。

銀河の情報が与えられたとき、質問を行うことで、鍵が隠された星の番号 A と宝箱が隠された星の番号 B を特定するあなたの戦略を実装したプログラムを作成せよ。

実装の詳細

あなたは 1 つのファイルを提出しなければならない。

あなたの提出するファイルは `thief.cpp` という名前である。`thief.cpp` は `#include` プリプロセッサ指令によって `thief.h` を読み込み、以下の関数を実装していなければならない。

- `void solve(int N, int M, std::vector<int> U, std::vector<int> V)`

この関数は各テストケースにおいて 1 回だけ呼び出される。

- 引数 N は星の個数 N である。
- 引数 M はワープ装置の個数 M である。
- 引数 U, V は長さ N の配列であり、 $0 \leq i \leq M-1$ について、 $U[i], V[i]$ はワープ装置 i が双方向に結ぶ星の番号 U_i, V_i である。



`thief.cpp` 内では以下の関数を呼び出すことができる。

★ `int query(std::vector<int> x)`

この関数を用いて、質問を行う。

- ◇ 引数 `x` は長さ M の配列である。 $0 \leq i \leq M-1$ について、 `x[i]` が 0 のときワープ装置 i について、星 U_i から星 V_i へ一方向に結ぶようにすることを、1 のときワープ装置 i について、星 V_i から星 U_i へ一方向に結ぶようにすることを表す。
- ◇ 戻り値は 0 または 1 である。戻り値が 0 のときいくつかのワープ装置を用いて星 A から星 B へ移動できないことを、1 のとき星 A から星 B へ移動できることを表す。
- ◇ 引数 `x` の長さは M でなければならない。これが満たされていない場合、不正解 [1] と判定される。
- ◇ 引数 `x` の各要素は、0 または 1 でなければならない。これが満たされていない場合、不正解 [2] と判定される。
- ◇ 関数 `query` を 300 回を超えて呼び出してはならない。300 回を超えて呼び出した場合、不正解 [3] と判定される。

★ `void answer(int A, int B)`

この関数を用いて、鍵が隠された星の番号と宝箱が隠された星の番号を解答する。

- ◇ 引数 `A` は、0 以上 $N-1$ 以下の整数である。鍵が隠された星の番号 A を表す。
- ◇ 引数 `A` は、0 以上 $N-1$ 以下でなければならない。これが満たされていない場合、不正解 [4] と判定される。
- ◇ 引数 `B` は、0 以上 $N-1$ 以下の整数である。宝箱が隠された星の番号 B を表す。
- ◇ 引数 `B` は、0 以上 $N-1$ 以下でなければならない。これが満たされていない場合、不正解 [5] と判定される。
- ◇ 誤った星の番号を解答した場合、不正解 [6] と判定される。
- ◇ 関数 `answer` はちょうど 1 回呼び出さなければならない。関数 `answer` を 2 回以上呼び出した場合、不正解 [7] と判定される。関数 `solve` の実行の終了時に関数 `answer` が 1 回も呼び出されていない場合、不正解 [8] と判定される。

重要な注意

- 内部での使用のために他の関数を実装したり、グローバル変数を宣言するのは自由である。
- あなたの提出したプログラムは、標準入力・標準出力、あるいは他のファイルといかなる方法でもやりとりしてはならない。ただし、標準エラー出力にデバッグ情報等を出力することは許される。



コンパイル・実行の方法

作成したプログラムをテストするための、採点プログラムのサンプルが、コンテストサイトからダウンロードできるアーカイブの中に含まれている。このアーカイブには、提出しなければならないファイルのサンプルも含まれている。

採点プログラムのサンプルは1つのファイルからなる。そのファイルは `grader.cpp` である。作成したプログラムをテストするには、これらのファイル `grader.cpp`, `thief.cpp`, `thief.h` を同じディレクトリに置き、次のようにコマンドを実行する。

```
g++ -std=gnu++20 -O2 -o grader grader.cpp thief.cpp
```

なお、アーカイブの中に含まれている `compile.sh` というファイルを代わりに実行してもよい。その場合、次のようにコマンドを実行する。

```
./compile.sh
```

コンパイルが成功すれば、`grader` という実行ファイルが生成される。

実際の採点プログラムは、採点プログラムのサンプルとは異なることに注意すること。採点プログラムのサンプルは単一のプロセスとして起動する。このプログラムは、標準入力から入力を読み込み、標準出力に結果を出力する。

採点プログラムのサンプルの入力

採点プログラムのサンプルは標準入力から以下の形式で入力を読み込む。

```
N M A B
U0 V0
U1 V1
⋮
UM-1 VM-1
```



採点プログラムのサンプルの出力

採点プログラムのサンプルは標準出力へ以下の情報を出力する (引用符は実際には出力されない).

- 正解の場合, 関数 `query` の呼び出し回数が “Accepted: 25” のように出力される.
- 不正解の場合, 不正解の種類が “Wrong Answer [4]” のように出力される.

実行するプログラムが複数の不正解の条件を満たした場合, 表示される不正解の種類はそれらのうち 1 つのみである. 採点プログラムのサンプルは, 不正解の条件を満たした場合, 途中で実行を打ち切ることができる.

採点に関する注意

いくつかのテストケースについて, 実際の採点プログラムは適応的 (adaptive) である. これは, 採点プログラムは初めから固定された答えを持たず, それ以前の `query` 関数の呼び出しに応じて採点プログラムが応答するということである. ただし, すべての応答に矛盾しないような答えが少なくとも 1 つ存在するということが保証される.

制約

すべての入力データは以下の条件を満たす.

- $2 \leq N \leq 10\,000$.
- $1 \leq M \leq 15\,000$.
- $0 \leq A \leq N - 1$.
- $0 \leq B \leq N - 1$.
- $A \neq B$.
- $0 \leq U_i < V_i \leq N - 1$ ($0 \leq i \leq M - 1$).
- $(U_i, V_i) \neq (U_j, V_j)$ ($0 \leq i < j \leq M - 1$).
- どの 2 つの星の間も, いくつかのワープ装置を用いることで移動できる.



小課題

1. (7 点) $M = N - 1$, $U_i = i$, $V_i = i + 1$ ($0 \leq i \leq M - 1$).
2. (13 点) $M = N - 1$, $U_i = 0$, $V_i = i + 1$ ($0 \leq i \leq M - 1$).
3. (2 点) $M = N - 1$, $N \leq 8$.
4. (8 点) $M = N - 1$, $N \leq 50$.
5. (5 点) $M = N - 1$, $N \leq 150$.
6. (5 点) $M = N - 1$, $N \leq 250$.
7. (40 点) $M = N - 1$. この小課題では、以下に従い得点が決定される.
 - 小課題 7 に対応するテストケースについて、1 つでも不正解があった場合、この小課題の得点は 0 点となる.
 - そうでない場合、この小課題のすべてのテストケースにおける、関数 `query` の呼び出し回数の最大値を T とする. このとき、小課題の得点は以下のように決定される.
 - $120 < T$ のとき、20 点.
 - $70 < T \leq 120$ のとき、30 点.
 - $T \leq 70$ のとき、40 点.
8. (20 点) 追加の制約はない. この小課題では、以下に従い得点が決定される.
 - 小課題 8 に対応するテストケースについて、1 つでも不正解があった場合、この小課題の得点は 0 点となる.
 - そうでない場合、この小課題のすべてのテストケースにおける、関数 `query` の呼び出し回数の最大値を T とする. このとき、小課題の得点は以下のように決定される.
 - $120 < T$ のとき、10 点.
 - $70 < T \leq 120$ のとき、15 点.
 - $T \leq 70$ のとき、20 点.

小課題 1, 2, 3, 4, 5, 6 の得点は関数 `query` の呼び出し回数によらない (300 回以下であればよい) が、70 回より多い場合はコンテストサイトにおいて「出力は部分的に正しい」と表記されることがある.



やりとりの例

採点プログラムのサンプルが読み込む入力の例と、それに対応する関数の呼び出しの例を以下に示す。

入力例 1	関数の呼び出しの例		
	呼び出し	呼び出し	戻り値
5 4 0 4	solve(5, 4, [0, 0, 1, 1], [1, 3, 2, 4])		
0 1		query([0, 1, 0, 0])	1
0 3		query([1, 1, 1, 0])	0
1 2		query([0, 0, 1, 0])	1
1 4		query([0, 0, 1, 1])	0
		answer(0, 4)	

1 回目の **query** 関数の呼び出しに対応する選択は以下のようになる。

- ワープ装置 0 について、星 0 から星 1 へ方向に結ぶようにする。
- ワープ装置 1 について、星 3 から星 0 へ方向に結ぶようにする。
- ワープ装置 2 について、星 1 から星 2 へ方向に結ぶようにする。
- ワープ装置 3 について、星 1 から星 4 へ方向に結ぶようにする。

この選択のもとで、星 0 から星 4 へはワープ装置 0, 3 をこの順に用いることで移動できるため、戻り値は 1 となる。

2 回目の **query** 関数の呼び出しに対応する選択は以下のようになる。

- ワープ装置 0 について、星 1 から星 0 へ方向に結ぶようにする。
- ワープ装置 1 について、星 3 から星 0 へ方向に結ぶようにする。
- ワープ装置 2 について、星 2 から星 1 へ方向に結ぶようにする。
- ワープ装置 3 について、星 1 から星 4 へ方向に結ぶようにする。

この選択のもとで、星 0 から星 4 へはいくつかのワープ装置を用いて移動できないため、戻り値は 0 となる。



3 回目の `query` 関数の呼び出しに対応する選択は以下のようになる。

- ワープ装置 0 について、星 0 から星 1 へ方向に結ぶようにする。
- ワープ装置 1 について、星 0 から星 3 へ方向に結ぶようにする。
- ワープ装置 2 について、星 2 から星 1 へ方向に結ぶようにする。
- ワープ装置 3 について、星 1 から星 4 へ方向に結ぶようにする。

この選択のもとで、星 0 から星 4 へはいくつかのワープ装置を用いて移動できるため、戻り値は 1 となる。

4 回目の `query` 関数の呼び出しに対応する選択のもとで、星 0 から星 4 へはいくつかのワープ装置を用いて移動できないため、戻り値は 0 となる。

`answer` 関数の呼び出しでは星 A が星 0 であり、星 B が星 4 であると解答したことを表している。

この入力例は小課題 3, 4, 5, 6, 7, 8 の制約を満たす。コンテストサイトからダウンロードできるアーカイブの中に含まれているファイルのうち、`sample-01-in.txt` は入力例 1 に対応する。

なお、コンテストサイトからダウンロードできるアーカイブの中に含まれている、提出しなければならないファイルのサンプルにおける関数の呼び出しは、これらの関数の呼び出しと同じである。