# 迷い猫 (Stray Cat)

蟻の Anthony が住む JOI 市には $N$ 個の町があり，0 から $N-1$ までの番号が付いている．Anthony の家は町 0 にある．また，$M$ 本の道があり，0 から $M-1$ までの番号が付いている．道 $i$ $(0 \leqq i \leqq M-1)$ は町 $U_i$ と町 $V_i$ を双方向に結んでいる．複数の道が同じ町の組を結んでいることはない．どの町からどの町へも，いくつかの道を通ることで移動することができる．

Anthony の友達である，猫の Catherine が JOI 市に遊びに来ることになった．Catherine は道の情報を知らない上にすぐ道に迷ってしまうので，Anthony はあらかじめ道に印を付けておくことにした．Anthony は $A$ 種類の印を付けることができ，0 から $A-1$ までの番号が付いている．

さて，Catherine は JOI 市のどこかの町に辿り着いた．Catherine は町 0 以外の町にいるとき，

直前に通った道 (存在すれば) 以外に，今いる町にどの印が付いている道が何本出ているか

を把握でき，次にどの道を進むか選ぶことができる．**直前に通った道以外は，付いている印の種類でしか区別できない．**進む道を上手く選んでいくことで，町 0 にあまり時間をかけずに到着したい．具体的には，初めに着いた町から町 0 まで行くために通る必要がある道の個数の最小値を $d$ とするとき．道を高々 $d+B$ 回進んで町 0 に到着したい．

道の情報が与えられたとき，道に印を付ける Anthony の戦略を実装したプログラムおよび，把握した情報が与えられたとき，進む道を選んでいく Catherine 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 2 つのファイルを提出しなければならない．

1 つ目のファイルは `Anthony.cpp` という名前である．このファイルは Anthony の行動を実装したファイルであり，以下の関数を実装していなければならない．そのファイルは `Anthony.h` をインクルードすること．

- `std::vector<int> Mark(int N, int M, int A, int B,`
  `std::vector<int> U, std::vector<int> V)`

  この関数は，最初に 1 回だけ実行される．
  - 引数 `N` は，町の個数 $N$ である．
  - 引数 `M` は，道の本数 $M$ である．
  - 引数 `A` は，印の種類数 $A$ である．
  - 引数 `B` は，道を進む回数の猶予を表す値 $B$ である．
  - 引数 `U`, `V` は，長さ $M$ の配列であり，`U[i]`，`V[i]` は道 $i$ が結んでいる町 $U_i$, $V_i$ を表す $(0 \leqq i \leqq$

$M-1$).
  - 戻り値 `x` は，長さ $M$ の配列でなければならない．長さが $M$ でない場合，**不正解 [1]** と判定される．`x[i]` ($0 \leqq i \leqq M-1$) は，道 $i$ に付ける印を表す．$0 \leqq$ `x[i]` $\leqq A-1$ でなければならない．$0 \leqq$ `x[i]` $\leqq A-1$ でない場合，**不正解 [2]** と判定される．

2 つ目のファイルは `Catherine.cpp` という名前である．このファイルは Catherine の行動を実装したファイルであり，以下の関数を実装していなければならない．そのファイルは `Catherine.h` をインクルードすること．

- `void Init(int A, int B)`
  この関数は，最初に 1 回だけ実行される．
  - 引数 `A` は，印の種類数 $A$ である．
  - 引数 `B` は，道を進む回数の猶予を表す値 $B$ である．
- `int Move(std::vector<int> y)`
  この関数は，Catherine が町 0 以外の町に着くたびに実行される．
  - 引数 `y` は，長さ $A$ の配列であり，Catherine が今いる町から出ている道のうち，直前に通った道 (存在すれば) 以外に，印 $j$ ($0 \leqq j \leqq A-1$) がついている道が `y[j]` 本あることを表す．
  - 戻り値 `z` は，$-1 \leqq$ `z` $\leqq A-1$ でなければならない．$-1 \leqq$ `z` $\leqq A-1$ でない場合，**不正解 [3]** と判定される．`z` $= -1$ は直前に通った道を引き返すことを表し，$0 \leqq$ `z` $\leqq A-1$ は直前に通った道以外の印 `z` が付いた道を進むことを表す．関数 `Move` が初めて呼び出されたときに `z` $= -1$ である場合，**不正解 [4]** と判定される．$0 \leqq$ `z` $\leqq A-1$ かつ `y[z]` $= 0$ である場合，**不正解 [5]** と判定される．

Catherine がある町から直前に通った道以外を進むとき，実際に進む道は，関数 `Move` の戻り値で指定された印が付いた道のうち 1 本が採点プログラムによって選ばれる．**この選択は無作為とは限らない．**

Catherine が道を $d+B$ 回進んだ (すなわち，関数 `Move` が $d+B$ 回実行された) 時点で町 0 に到着していない場合，**不正解 [6]** 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．ただし，提出された 2 つのプログラムは，採点プログラムとまとめてリンクされて 1 つの実行ファイルになるので，各ファイル内のすべてのグローバル変数と内部関数を無名名前空間内で宣言して，他のファイルとの干渉を避ける必要がある．採点時には，このプログラムは Anthony 側，Catherine 側として 2 個のプロセスとして実行されるので，Anthony 側と Catherine 側でプログラム中のグローバル変数を共有することはできない．

- 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，`grader.cpp`, `Anthony.cpp`, `Catherine.cpp`, `Anthony.h`, `Catherine.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++14 -O2 -o grader grader.cpp Anthony.cpp Catherine.cpp
```

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．

$N\ M\ A\ B\ S$
$U_0\ V_0$
$\vdots$
$U_{M-1}\ V_{M-1}$

$S$ は Catherine が最初に着いた町の番号を表す．

### 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)．

- 不正解 [1], [2], [3], [4], [5] のいずれかの場合，不正解の種類が “`Wrong Answer [1]`” のように出力される．
- $N+B$ 回の移動で町 0 に到着していない場合，“`Wrong Answer; Number of moves > N + B`” と出

力される．

- その他の場合，移動回数 (関数 `Move` の呼び出し回数) が “`Number of moves = 4`” のように出力される．正解か不正解 [6] かは判定されないことに注意せよ．

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においては，Catherine がある町から直前に通った道以外を進むとき，実際に進む道は，関数 `Move` の戻り値で指定された印が付いた道のうち 1 本が，実行ごとに結果が変わらない疑似乱数を用いて一様に無作為に選ばれる．シードの値を変更したい場合は，

```
./grader 2020
```

のように 1 個目の引数に整数値を与えて実行せよ．

## 制約

- $2 \leqq N \leqq 20\,000$.
- $1 \leqq M \leqq 20\,000$.
- $1 \leqq S \leqq N - 1$ ($S$ は Catherine が最初に着いた町の番号を表す).
- $0 \leqq U_i < V_i \leqq N - 1$ $(0 \leqq i \leqq M - 1)$.
- $(U_i, V_i) \neq (U_j, V_j)$ $(0 \leqq i < j \leqq M - 1)$.
- どの町からどの町へも，いくつかの道を通ることで移動することができる．

## 小課題

1. (2 点) $A = 4$，$B = 0$，$M = N - 1$.
2. (2 点) $A = 4$，$B = 0$.
3. (2 点) $A = 3$，$B = 0$，$M = N - 1$.
4. (9 点) $A = 3$，$B = 0$.
5. (5 点) $A = 2$，$B = 2N$，$M = N - 1$，$6 \leqq N \leqq 500$.
6. (71 点) $A = 2$，$B = 12$，$M = N - 1$.
7. (9 点) $A = 2$，$B = 6$，$M = N - 1$.

## 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 |
| --- |
| 7 6 2 6 1<br>0 2<br>0 4<br>1 2<br>1 3<br>1 5<br>4 6 |

| Anthony | | Catherine | |
| --- | --- | --- | --- |
| 呼び出し | 戻り値 | 呼び出し | 戻り値 |
| Mark(7,6,2,6,[0,0,1,1,1,4],[2,4,2,3,5,6]) | [1,0,0,1,0,1] | | |
| | | Init(2,6) | |
| | | Move([2,1]) | 0 |
| | | Move([0,0]) | -1 |
| | | Move([1,1]) | 0 |
| | | Move([0,1]) | 1 |

この例では，Catherine は町 1，町 5，町 1，町 2，町 0 の順に移動している．$d = 2$ であり，移動回数は 4 である．

この入力例は小課題 7 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ファイルのうち，sample-02.txt は小課題 4 の制約を満たす．