遺産相続 (Inheritance)

IOI 国の鉄道をすべて所有していた大富豪の JOI 氏が逝去した．鉄道は遺言に従って分割相続されることになった．

IOI 国には N 個の都市と，それらの間を結ぶ M 本の鉄道がある．都市には 1 から N までの番号が付けられ，鉄道には 1 から M までの番号が付けられている．鉄道 i は都市 A_i と都市 B_i の間を双方向に結び，また年間に C_i 円の収益を上げている．鉄道の利用客数や乗車料金は多様なため，$C_1, \ldots, C_M$ はそれぞれ相異なる．同じ二つの都市を結ぶ鉄道が複数あるかもしれない．

遺言には以下のように鉄道の分割相続の方法が記されていた．

- 鉄道は，JOI 氏の K 人の子供に相続させる．子供たちには年齢が高い順に 1 から K までの番号が付いている．
- それぞれの子供は，M 本の鉄道のうちのいくつか (0 本の場合もありうる) を相続する．
- はじめに M 本の鉄道の中から，子供 1 がいくつか選び，自分の相続分とする．次に残った鉄道の中から，子供 2 が自分の相続分を決める．以下同様にして，K 人の子供が順番に，自分の相続分を決めていく．
- どの子供も，すでに相続先が決まった鉄道は相続することができない．すなわち，子供 j の相続分の中に鉄道 i が含まれていたら，それより若い子供 k $(k > j)$ は鉄道 i を自分の相続分の中に含めることができない．
- どの子供も，自分の相続分を決める際，相続分がサイクルを含まないようにしなければならない．すなわち，鉄道 $i_1, i_2, \ldots, i_m$ $(i_1, i_2, \ldots, i_m$ は相異なる) を一回ずつ利用することである都市から出発して同じ都市に戻ってくることができるとき，どの子供も，鉄道 $i_1, i_2, \ldots, i_m$ をすべて一人で相続することはできない．
- 誰にも相続されずに残った鉄道は IOI 国に寄贈される．

どの子供も，父に似て貪欲であるため，相続する鉄道の年間収益の合計ができるだけ大きくなるように自分の相続分を選ぶ．どの子供についても，年間収益の合計が最大となるような相続分の選び方は，ただ一通りであることが証明できる．それぞれの鉄道が誰に相続されるかを求めよ．

課題

IOI 国の鉄道の情報と，JOI 氏の子供の人数が与えられたとき，それぞれの鉄道が誰に相続されるかを求めるプログラムを作成せよ．

入力

標準入力から以下の入力を読み込め．

- 1 行目には，3 個の整数 N, M, K が空白を区切りとして書かれている．これは IOI 国には N 個の都市と M 本の鉄道があり，JOI 氏には K 人の子供がいることを表す．
- 続く M 行のうちの i 行目 $(1 \leqq i \leqq M)$ には，整数 A_i, B_i, C_i が空白を区切りとして書かれている．これは鉄道 i は都市 A_i と都市 B_i を双方向に結び，年間収益が C_i 円であることを表す．

出力

出力は M 行からなる．i 行目 $(1 \leqq i \leqq M)$ には，鉄道 i を相続する子供の番号を出力せよ．IOI 国に寄贈される場合は 0 と出力せよ．

制限

すべての入力データは以下の条件を満たす．

- $2 \leqq N \leqq 1\,000$.
- $1 \leqq M \leqq 300\,000$.
- $1 \leqq K \leqq 10\,000$.
- $1 \leqq A_i \leqq N$, $1 \leqq B_i \leqq N$ $(1 \leqq i \leqq M)$.
- $A_i \neq B_i$ $(1 \leqq i \leqq M)$.
- $1 \leqq C_i \leqq 1\,000\,000\,000$ $(1 \leqq i \leqq M)$.
- $C_i \neq C_j$ $(1 \leqq i < j \leqq M)$.

小課題

小課題 1 [15 点]

- $K \leqq 10$ を満たす．

小課題 2 [85 点]

追加の制限はない．

入出力例

入力例 1	出力例 1
3 5 2 1 2 3 1 2 1 2 3 4 2 3 6 1 3 2	1 0 2 1 2

- 子供 1 は，鉄道 1, 2, 3, 4, 5 のうちから鉄道 1, 4 を選んで相続する．このとき相続する鉄道の年間収益の合計が 3 + 6 = 9 円となり，これが最大値である．
- 子供 2 は，残った鉄道 2, 3, 5 のうちから鉄道 3, 5 を選んで相続する．このとき相続する鉄道の年間収益の合計が 4 + 2 = 6 円となり，これが最大値である．
- 残った鉄道 2 は IOI 国に寄贈される．

入力例 2	出力例 2
3 6 5 1 2 1 1 2 2 2 3 3 2 3 4 3 1 5 3 1 6	4 3 2 1 2 1

相続する鉄道の本数は子供によって異なっているかもしれない． 1 本も鉄道を相続しない子供がいるかもしれない．

記憶縛り (Limited Memory)

国際情報オリンピックの日本代表に選ばれた JOI ちゃんは，情報処理技術を高めるため，情報オリンピック日本委員会の K 理事長から以下のような課題を提示された．

K 理事長は，JOI ちゃんには分からないように，ある文字列 S を手帳に書いた．S は ‘<’, ‘>’, ‘[’, ‘]’ の 4 種類の文字からなる．K 理事長は JOI ちゃんに課題の内容と S の長さが書かれたプリントを渡した．

JOI ちゃんの課題は，S が**良い文字列**であるか否かを判定することである．ここで，良い文字列とは次のように定義される：

- 空文字列 (つまり，長さ 0 の文字列) は良い文字列である．
- x が良い文字列ならば，<x> (つまり，x を山括弧 <> で囲った文字列) は良い文字列である．
- x が良い文字列ならば，[x] (つまり，x を角括弧 [] で囲った文字列) は良い文字列である．
- x, y が良い文字列ならば，xy (つまり，x, y をこの順に連結した文字列) は良い文字列である．
- 以上で良い文字列と定義されるもののみが良い文字列である．

例えば，“<>[]” や “[<>]<>” は良い文字列であり，“><” や “[<]>” は良い文字列ではない．

JOI ちゃんは，毎日正午に高々 1 回，K 理事長に電話をすることができる．電話では，整数 I を指定することで，K 理事長から S の I 文字目を教えてもらうことができる．

さて，JOI ちゃんには，「この課題ではメモをとってはならない」という重大な制約が課されている．JOI ちゃんは毎晩 22 時に就寝し朝 6 時に起床するが，睡眠中に覚えていられるのは 22 ビットの情報のみである．より正確には，就寝前に 0 以上 $2^{22} - 1$ 以下の整数を 1 個記憶し，翌日は記憶した整数のみに基づいて課題に取り組まなければならない．ただし，S の長さはプリントに書いてあるので，いつでも参照することができる．

JOI ちゃんは，就寝前に整数を 1 個記憶する代わりに，S が良い文字列であるか否かを K 理事長に電子メールで回答してもよい．この場合課題は終了となり，正解か不正解かが判定される．ただし，課題の開始から 15 000 日以内に電子メールを送らなかった場合，不正解と判定される．

課題

JOI ちゃんの戦略を実装し，上記の課題で正解できるプログラムを作成せよ．

実装の詳細

あなたは，JOI ちゃんの戦略を実装した 1 個のプログラムを書かなければならない．プログラムは `Memory_lib.h` をインクルードすること．

プログラムは，以下の関数を実装しなければならない．

- `int Memory(int N, int M)`

  この関数は，S の長さおよび記憶した整数が与えられたときの，その日の JOI ちゃんの行動に対応する．

  - 引数 `N` は，文字列 S の長さ N である．
  - 引数 `M` は，前日の就寝前に記憶した内容を表す整数 M である．ただし，課題の開始時には $M = 0$ を記憶しているものとして扱う．
  - この関数の中では，以下で説明されるように関数 `Get` を高々 1 回呼び出すことができる．
  - この関数は，0 以上 $2^{22} - 1$ 以下の整数，または -1 または -2 を返さなければならない．この範囲の外の値を返した場合，**不正解 [1]** となる．
    - 戻り値が 0 以上 $2^{22} - 1$ 以下の整数の場合，就寝前にその値を記憶することを表す．
    - 戻り値が -1 の場合，「S は良い文字列である」と電子メールで回答することを表す．
    - 戻り値が -2 の場合，「S は良い文字列ではない」と電子メールで回答することを表す．
  - この関数は，引数 `N`, `M` の値および呼び出した `Get` の戻り値のみに依存した挙動を行うことが期待される．また，実際の採点プログラムではこの関数は $2^{22} \times 4$ 回呼び出される．詳細は，「採点の手順」と「重要な注意」を参照せよ．

プログラム中では以下の関数を呼び出すことができる．

- `char Get(int I)`

  この関数は，`Memory` の 1 回の呼び出しにつき高々 1 回しか呼び出すことができない．2 回以上呼び出した場合，**不正解 [2]** となる．

  引数 `I` は，$1 \leqq I \leqq N$ を満たす整数 I でなければならない．これを満たさない場合，**不正解 [3]** となる．

  この関数の戻り値は，文字列 S の I 文字目の文字を表す．

採点の手順

採点用の各入力データは複数のテストケースからなり，それらにおいて文字列 S の長さは同じ値 N である．

採点は以下の手順で行われる．不正解と判定された場合はその時点でプログラムは終了される．

(1) 引数 `N`, `M` の値および呼び出した `Get` の戻り値のすべての場合に対する `Memory` の挙動をまとめて調べる．すなわち，$0 \leqq M \leqq 2^{22} - 1$ を満たす整数 M それぞれについて，以下を行う：

(i) 文字 c を '<', '>', '[', ']' のそれぞれとして，以下を行う：

引数 `N` を N，引数 `M` を M として `Memory` を呼び出す．このとき，`Get` が呼び出されたならば，文字 c が返される．`Memory` が返した値を $m(M, c)$ とおく．

(ii) (i) での 4 回の `Memory` の呼び出しにおいて，`Get` を呼び出すか否かは同じでなければならない．さらに，`Get` を呼び出した場合は，4 回とも同じ整数 I を引数 `I` として呼び出さなければならず，`Get` を呼び出さなかった場合は，4 回とも同じ値を返さなければならない．これらを満たさない場合，**不正解 [4]** となる．`Get` を呼び出したときの引数 `I` を $i(M)$ とおく (ただし，`Get` を呼び出さなかった場合は $i(M) = 1$ とおく)．

(2) 各テストケースにおける文字列 S に対して，問題文で説明された課題のシミュレーションを行う．すなわち，以下を行う：

(i) $M = 0$ とする．

(ii) 以下を繰り返す：

(a) c を S の $i(M)$ 文字目の文字とする．

(b) M を $m(M, c)$ で置き換える．

(c) $M = -1$ または $M = -2$ ならば，(iii) へ進む．

(d) この項目に 15 000 回到達したら，**不正解 [5]** となる．

(iii) 以下のいずれかの場合，**不正解 [6]** となる：

- S が良い文字列だが，$M = -2$ である．
- S が良い文字列ではないが，$M = -1$ である．

(3) 正解となる．

重要な注意

- 実行時間計測・使用メモリ計測の対象となるのは，「採点の手順」における手順 (1) である．手順 (1) において `Memory` は $2^{22} \times 4$ 回呼び出されることに注意せよ．
- 手順 (1) における $2^{22} \times 4$ 回のすべての `Memory` の呼び出しにおいて，不正解 [1], [2], [3] となったり，実行時エラーを起こしたりしてはならないことに注意せよ．

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として，grader-simple と grader-strict の 2 つが提供される．grader-simple のファイルは `grader-simple.c` または `grader-simple.cpp` であり，grader-strict のファイルは `grader-strict.c` または `grader-strict.cpp` である．例えば，作成したプログラムを `Memory.c` または `Memory.cpp` とするとき，作成したプログラムを grader-simple や grader-strict でテストするには，次のようにコマンドを実行する．

- C の場合

```
gcc -O2 -o grader-simple grader-simple.c Memory.c -lm
gcc -O2 -o grader-strict grader-strict.c Memory.c -lm
```

- C++ の場合

```
g++ -std=c++11 -O2 -o grader-simple grader-simple.cpp Memory.cpp
g++ -std=c++11 -O2 -o grader-strict grader-strict.cpp Memory.cpp
```

コンパイルが成功すれば，`grader-simple` あるいは `grader-strict` 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，grader-simple あるいは grader-strict とは異なることに注意すること．grader-simple や grader-strict は単一のプロセスとして起動する．これらのプログラムは，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概要

grader-simple は,「採点の手順」 (1) のように始めにまとめて `Memory` を呼び出すのではなく，あなたのプログラムと交互にやりとりを行うことによって問題文で説明された課題のシミュレーションを行う.「やりとりの例」を参照せよ．

grader-strict は,「採点の手順」と同様に `Memory` を呼び出す．

以下の点は実際の採点プログラムの挙動とは異なるので注意せよ：

- grader-simple は,「採点の手順」と `Memory` の呼び出し方が異なるので，不正解 [4] を判定しない．
- **grader-simple** や **grader-strict** は，不正解 **[6]** の判定を行わず，代わりに M の値を出力する．

採点プログラムのサンプルの入力

grader-simple や grader-strict は標準入力から以下の入力を読み込む.

- 1 行目には整数 N, Q が空白を区切りとして書かれている．N は文字列 S の長さを表し，Q ($0 \leqq Q \leqq 2^{31} - 1$) はテストケースの個数を表す.
- 続く Q 行の各行には，各テストケースにおける文字列 S (長さ N) が書かれている.

採点プログラムのサンプルの出力

grader-simple や grader-strict 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).

- 「採点の手順」(2)(iii) における M の値を (各テストケースごとに) 1 行に出力する.
- 不正解 [1], [2], [3], [4], [5] のいずれかと判定された場合，不正解の種類が “`Wrong Answer [1]`” のように出力される．その時点でプログラムは終了し，それ以降の出力は行われない.

制限

すべての入力データは以下の条件を満たす.

- $1 \leqq (S$ の長さ$) \leqq 100$.
- S の各文字は '<', '>', '[', ']' のいずれかである.

小課題

小課題 1 [10 点]

- $(S$ の長さ$) \leqq 8$ を満たす.

小課題 2 [10 点]

- $(S$ の長さ$) \leqq 14$ を満たす.

小課題 3 [5 点]

- $(S$ の長さ$) \leqq 24$ を満たす.

小課題 4 [5 点]

- $(S$ の長さ$) \leqq 30$ を満たす.

小課題 5 [10 点]

- S の各文字は '<', '>' のいずれかである.

小課題 6 [60 点]

追加の制限はない.

やりとりの例

grader-simple 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	呼び出しの例			
	呼び出し	戻り値	呼び出し	戻り値
4 1 <>[]	Memory(4, 0)			
			Get(1)	
				<
		2015		
	Memory(4, 2015)			
			Get(3)	
				[
		3		
	Memory(4, 3)			
			Get(2)	
				>
		23		
	Memory(4, 23)			
			Get(4)	
				]
		4194303		
	Memory(4, 4194303)			
			Get(3)	
				[
		-1		

上のやりとりが行われたとき，grader-simple は −1 を出力する．

防壁 (Walls)

あなたは，JOI 社が発売したテレビゲームソフトを手に入れた．なかなか良くできたゲームであり，それなりに楽しみながら毎日プレイしていた．

ある日，ゲーマーの中で「レーザー」と呼称されるステージが出現した．どうやらそのステージは極めて難しく，優れたゲーマーでさえもほんのわずかな確率でしかクリアできない物であるらしい．何度もそのステージに挑戦する中であなたは，とても高速な判断を行うことでクリアできる可能性が有ることに気づき，プログラムを作成して対処出来るのではないかと考えた．

レーザーステージは，N 個の防壁が配置された場所が舞台となっている．舞台は長方形であり，1×1 の正方形のマスに分かれている．各マスは 0 以上の整数 x, y によって (x, y) と表される．$(0, 0)$ は左下隅のマスであり，(x, y) は $(0, 0)$ から右に x マス，上に y マス進んだマスを表す．

ステージが始まると敵が出現し攻撃を行う．M 回の攻撃が順番に行われる．j 回目の攻撃で，敵はマス $(P_j, N+1)$ からマス $(P_j, 0)$ に向かってレーザーをまっすぐに発射する．

各防壁は y 座標が同じ連続したいくつかのマスに置かれる．防壁 i $(1 \leqq i \leqq N)$ は左右の幅 $B_i - A_i + 1$，上下の幅 1 の長方形であり，ステージ開始時にはマス (A_i, i) からマス (B_i, i) までを占める．あなたは，敵の最初の攻撃の直前と，敵の攻撃と攻撃の合間に何度でも防壁を左右に動かすことができる．1 回の移動で 1 つの防壁を右に 1 マス動かすか，または左に 1 マス動かすことができる．

レーザーは防壁にぶつかると威力が弱くなる．レーザーをすべての防壁にぶつけるように防壁を動かすことで，レーザーの威力を最小にしたい．

あなたは，このステージにおいて，防壁の移動回数をできるだけ少なくしたい．

課題

ステージ開始時の各防壁の位置と，それぞれの敵の攻撃の位置が与えられる．すべてのレーザーをすべての防壁にぶつけるように防壁を動かすとき，それぞれの防壁の移動回数の最小値を求めよ．

入力

標準入力から以下のデータを読み込め．

- 1 行目には，整数 N, M が空白を区切りとして書かれている．これは，このステージには防壁が N 個あり，敵の攻撃が M 回行われることを表す．
- 続く N 行のうちの i 行目 $(1 \leqq i \leqq N)$ には，整数 A_i, B_i が空白を区切りとして書かれている．これは，ステージ開始時において，防壁 i がマス (A_i, i) からマス (B_i, i) までの位置に置かれていることを表す．
- 続く M 行のうちの j 行目 $(1 \leqq j \leqq M)$ には，整数 P_j が書かれている．これは，j 回目の攻撃において，敵はマス $(P_j, N+1)$ からマス $(P_j, 0)$ に向かってレーザーをまっすぐに発射することを表す．

出力

標準出力に N 行出力せよ．i 行目 $(1 \leqq i \leqq N)$ には，防壁 i の移動回数の最小値を出力せよ．

制限

すべての入力データは以下の条件を満たす．

- $1 \leqq N \leqq 200\,000$.
- $1 \leqq M \leqq 200\,000$.
- $0 \leqq A_i \leqq B_i \leqq 1\,000\,000\,000$ $(1 \leqq i \leqq N)$.
- $0 \leqq P_j \leqq 1\,000\,000\,000$ $(1 \leqq j \leqq M)$.

小課題

小課題 1 [10 点]

- $N = 1$ を満たす．

小課題 2 [45 点]

- $A_i = 0$ $(1 \leqq i \leqq N)$ を満たす．

小課題 3 [45 点]

追加の制限はない．

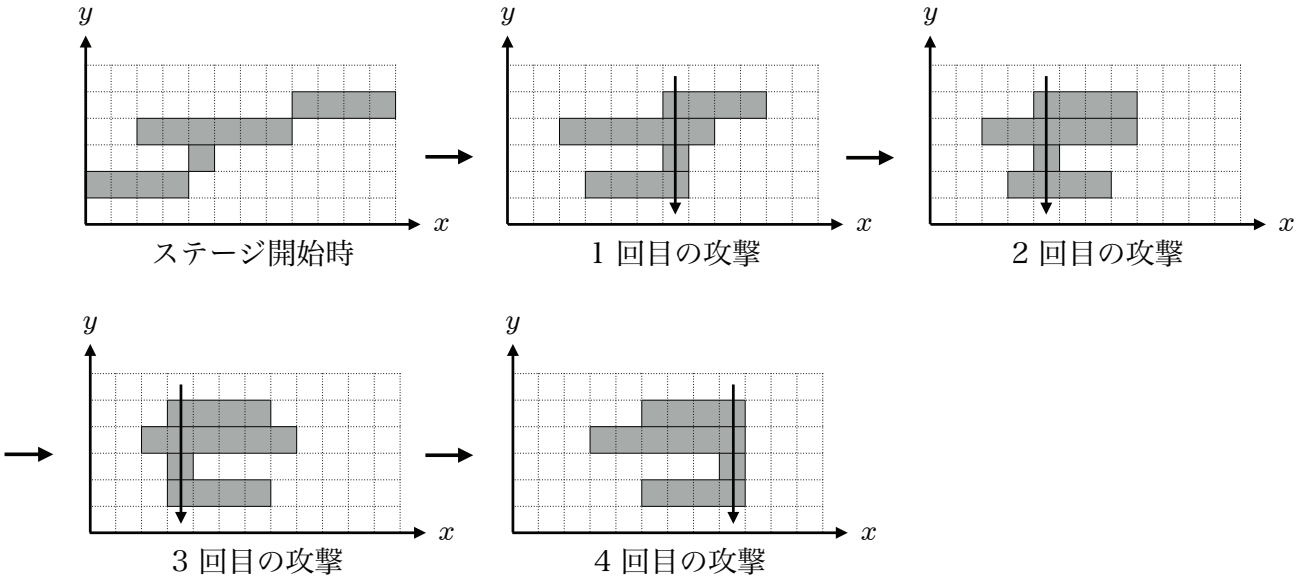
入出力例

入力例 1	出力例 1
4 4 0 3 4 4 2 7 8 11 6 4 3 8	5 10 1 7

この入力において，防壁の移動回数を最小にする動かし方の 1 つは以下の通りである：

- 1 回目の攻撃の前に，防壁 1 を右に 3 回動かし，防壁 2 を右に 2 回動かし，防壁 3 は動かさず，防壁 4 を左に 2 回動かす．
- 2 回目の攻撃の前に，防壁 1 は動かさず，防壁 2 を左に 2 回動かし，防壁 3 は動かさず，防壁 4 を左に 2 回動かす．
- 3 回目の攻撃の前に，防壁 1 は動かさず，防壁 2 を左に 1 回動かし，防壁 3 は動かさず，防壁 4 を左に 1 回動かす．
- 4 回目の攻撃の前に，防壁 1 を右に 2 回動かし，防壁 2 を右に 5 回動かし，防壁 3 を右に 1 回動かし，防壁 4 を右に 2 回動かす．

この動かし方において，防壁 1 を 5 回，防壁 2 を 10 回，防壁 3 を 1 回，防壁 4 を 7 回動かすことになる．



入力例 2	出力例 2
7 11 12 39 22 23 5 38 6 47 10 43 0 50 18 46 38 19 15 1 12 29 29 0 6 40 6	34 178 13 6 18 0 36