Aesthetic Islands

يخطط منظمو الأولمبياد الإفريقي للإعلام الآلي لتنظيم رحلة لبحيرة كيفو, والتي تعرف بجزرها الجميلة الكثيرة. للتخطيط كما ينبغي, عليهم معرفة عدد الجزر في البحيرة بالضبط. لحسن الحظ, لقد تحصلوا للتو على صورة بالقمر الصناعي لخريطة البحيرة.

تتمثل الخريطة في شبكة حجمها $N \times M$. أسطر الخريطة مرقمة من 0 إلى $N-1$ من الأعلى إلى الأسفل, وأعمدة الخريطة مرقمة من 0 إلى $M-1$ من اليسار إلى اليمين. الخلية في السطر i والعمود j ($0 \le i < N, 0 \le j < M$) يرمز لها ب (i, j). كل خلية تحتوي على 1 إذا كانت براً و 0 إذا كانت ماءً.

خليتي بر تعتبران متجاورتين إذا كانا يشتركان في جانب. بصفة أخرى, خليتي بر تعتبران متجاورتين إذا كانت إحداهما مباشرة فوق, تحت, على يمين أو على يسار الأخرى. الخلايا التي تتلامس في الركن فقط لا تعتبر متجاورة.

نسمي جزيرة مجموعة أعظمية من خلايا البر بحيث يمكن ربط كل زوج من الخلايا في المجموعة بتسلسل من الخلايا البرية المتجاورة. هنا, أعظمية يقصد بها أنه لا يمكن إضافة خلية برية أخرى إلى المجموعة بحيث يبقى هذا الشرط محققاً.

على سبيل المثال, في الخريطة المجاورة:

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

توجد 5 جزر, لأن التلامس بالركن لا يجعل خلايا البر متجاورة.

في الخريطة المجاورة:

$$\begin{matrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{matrix}$$

توجد 3 جزر.

نعلم كذلك أن بحيرة في بحيرة كيفو, كل الجزر جميلة. تسمى جزيرة بالجميلة إذا كان من أجل أي خلايا برية تنتمي للجزيرة وتقع على نفس السطر أو العمود, كل خلية بينها هي أيضا خلايا برية. بصفة أخرى, في كل جزيرة, كل سطر وكل عمود تشغله الجزيرة يحتوي على سلسلة وحيدة متواصلة من الخلايا البرية.

على سبيل المثال, الخرائط التالية هي خرائط ممكنة لبحيرة كيفو (أي تحتوي فقط على الجزر الجميلة):

$$\begin{matrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{matrix}$$

الخريطة الثانية تحتوي على 4 جزر, وكل منها جميلة.

ولكن, الخرائط التالية ليست خرائط ممكنة لبحيرة كيفو:

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{matrix}$$

المنظمون لديهم حاسوب خاص برقاقة ذاكرة تحتوي على K بيت (bits), مرقمة من 0 إلى $K-1$. الخريطة مخزنة بالفعل في هاته الذاكرة. بالتحديد, قيمة الخلية (i,j) مخزنة في البيت رقم $i \cdot M + j$ من الذاكرة.

لا يمكن للمنظمين قراءة أو تعديل الذاكرة مباشرة, وبالتالي لا يعرفون كيف تبدو الخريطة. بدلاً من ذلك, بإمكانهم تقديم برنامج إلى الحاسوب.

برنامجك لا يعرف القيم المخزنة في الذاكرة. بل يعرف فقط N, M و K. مهمته وصف سلسلة ثابتة من العمليات (bitwise operations) التي يقوم الحاسوب بتنفيذها على الذاكرة التي تحتوي على الخريطة.

بعد انتهاء برنامجك من التنفيذ, العمليات التي تم وصفها من طرفه سيتم تنفيذها على محتويات الذاكرة المجهولة. في النهاية, على برنامجك اختيار بعض من الbits المشغولة. هاته الbits, والتي يتم قراءتها من أصغر بيت (least significant bit) إلى أكبر بيت (most significant bit), عليها تمثيل عدد الجزر في نظام العد الثنائي (binary) بعد تنفيذ برنامجك على الخريطة الخفية المخزنة في الذاكرة.

عليك تنفيذ برنامج يصف العمليات التي على الحاسوب تنفيذها على الذاكرة.

في البداية, الbits $0, 1, \ldots, N \cdot M - 1$ تحتوي على الخريطة. هاته الbits هي بالفعل مشغولة ولا يمكن إعادة كتابتها. كل الbits الباقية هي في البداية خالية.

يمتلك الحاسوب 6 أنواع من التعليمات التي يمكن استعمالها في برنامجك للقيام بعمليات على bits الذاكرة. كل تعليمة هي عملية (bitwise operation) على واحدة أو اثنتين من الbits, والنتيجة تخزن في bit ذاكرة جديد.

كلما تشكل تعليمة bit جديد, يخزنها الحاسوب في أول bit خالية متاحة في الذاكرة. هذا الbit يصبح مشغولاً ويمكن القيام بعمليات عليه لاحقاً. كل bit ذاكرة يمكن الكتابة عليه مرة واحدة على الأكثر.

التعليمات ال6 المتاحة يتم وصفها أدناه. بعد الحصول على كل تعليمة, يقوم الحاسوب بإنشاء bit جديد C $(0 \leq C < K)$, والذي هو أول bit ذاكرة خالٍ. من أجل bit رقمه X, ليكن $memory[X]$ يرمز إلى القيمة المخزنة في الbit X من الذاكرة.

- $and(A,B)$: تقوم بحساب bitwise-AND للbits التي رقمها A و B وتخزنها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 1$ إذا كان كلا $memory[A]$ و $memory[B]$ يساويان 1, وتجعل $memory[C] := 0$ في الحالات الأخرى. هاته الدالة تقوم بإرجاع C.
- $or(A,B)$: تقوم بحساب bitwise-OR للbits التي رقمها A و B وتخزنها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 1$ إذا كان على الأقل واحد من $memory[A]$ و $memory[B]$ يساوي 1, وتجعل $memory[C] := 0$ في الحالات الأخرى. هاته الدالة تقوم بإرجاع C.
- $xor(A,B)$: تقوم بحساب bitwise-XOR للbits التي رقمها A و B وتخزنها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 1$ إذا كانت بالضبط واحدة من $memory[A]$ و $memory[B]$ تساوي 1, وتجعل $memory[C] := 0$

في الحالات الأخرى. هاته الدالة تقوم بإرجاع C.

- $not(A)$: تقوم بحساب bitwise-NOT للbit التي رقمها A وتخزنها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 1 - memory[A]$. هاته الدالة تقوم بإرجاع C.
- $nand(A, B)$: تقوم بحساب bitwise-NAND للbits التي رقمها A و B وتقوم بتخزينها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 0$ إذا كان كلا $memory[A]$ و $memory[B]$ يساويان 1, وتجعل $memory[C] := 1$ في الحالات الأخرى. هاته الدالة تقوم بإرجاع C.
- $nor(A, B)$: تقوم بحساب bitwise-NOR للbits التي رقمها A و B وتخزنها في bit C من الذاكرة. بصفة أخرى, تجعل $memory[C] := 0$ إذا كان على الأقل واحد من$memory[A]$ و $memory[B]$ يساوي 1, وتجعل $memory[C] := 1$ في الحالات الأخرى. هاته الدالة تقوم بإرجاع C.

يقوم الحاسوب بتنفيذ العمليات التي تم وصفها بالترتيب التي يحدده برنامجك. في النهاية, بعض من bits الذاكرة المختارة من طرف برنامجك يجب أن تحتوي على تمثيل عدد الجزر في الخريطة في النظام الثنائي (binary).

Implementation Details

C++/Python Interface

من أجل الحلول بلغة C++/Python , عليك تنفيذ الإجراء التالي:

```
int32[] construct_instructions(int32 N, int32 M, int32 K)
```

- N: عدد الأسطر
- M: عدد الأعمدة
- K: عدد الbits في رقاقة الذاكرة
- يتم استدعاء هذا الإجراء مرة واحدة بالضبط وعليه أن يشكل سلسلة من التعليمات لتنفيذ المهمة المطلوبة.
- على هذا الإجراء إرجاع مصفوفة V تحتوي على المواضع في الذاكرة التي تحتوي على الإجابة, من أصغر واحدة (least significant bit) إلى أكبر واحدة (most significant bit). بصفة أخرى, ليكن l طول V. عدد الجزر في الخريطة عليه أن يساوي $2^0 \cdot memory[V[0]] + 2^1 \cdot memory[V[1]] + \ldots + 2^{l-1} \cdot memory[V[l-1]]$.
- كل موضع bit في V عليه أن يشير إلى bit مشغول.
- طول V يجب أن لا يتجاوز 20.

على هذا الإجراء استدعاء الإجراءات التالية لتشكيل سلسلة التعليمات:

```
int32 append_and(int32 A, int32 B)
int32 append_or(int32 A, int32 B)
int32 append_xor(int32 A, int32 B)
int32 append_not(int32 A)
int32 append_nand(int32 A, int32 B)
int32 append_nor(int32 A, int32 B)
```

- كل إجراء يقوم بإضافة تعليمة and, or, xor, not, $nand$ أو nor إلى البرنامج, على التوالي.

- من أجل `append_not`, A عليه أن يكون bit مشغول. من أجل الإجراءات الخمسة الأخرى, كلا A و B عليها أن تكون مشغولة.
- كل إجراء يقوم بإرجاع موضع الذاكرة الذي سيتم تخزين الbit الناتج فيه.

بإمكانك أيضاً استدعاء هذا الإجراء للمساعدة في تجريب الحل الخاص بك:

```
void append_print(string label, int32[] A)
```

- كل موضع في A إلى `append_print` عليه أن يشير إلى bit مشغول.
- أي استدعاء لهذا الإجراء سيتم تجاهله عند تصحيح الحل الخاص بك ولن يتم اعتباره على أنه تعليمة.
- في مصحح العينات, يقوم هذا الإجراء بإضافة عملية $print(label, A)$ إلى البرنامج.
- عندما يواجه مصحح العينات عملية $print(label, A)$ عند تنفيذ برنامج, يقوم بكتابة $label$ متبوع بمحتويات مواضع الbits ل A في سطر وحيد بنفس الترتيب التي تظهر به في A (أنظر إلى قسم "Sample Grader" من أجل التفاصيل).

C Interface

For C submissions, include the provided `islands_c.h` header and implement:

```
int construct_instructions(int N, int M, int K, int answer_bits[]);
```

The grader supplies `answer_bits`, an array with capacity `ISLANDS_MAX_ANSWER_BITS` (20). Write the indices of V, from least significant bit to most significant bit, into `answer_bits[0]`, `answer_bits[1]`, and so on, and return the number l of indices written. Thus, $V[i] = answer_bits[i]$ for $0 \le i < l$. The returned value must satisfy $0 \le l \le 20$. Do not write beyond the capacity of `answer_bits` and do not free this grader-owned array.

In C, the exact signature of the debugging procedure is:

```
void append_print(const char *label, const int A[], int length);
```

Here, `label` must point to a null-terminated string, and `length` is the number of elements in `A`. `A` may be `NULL` only when `length` is 0 and must be nonnegative.

تصحيح الحل الخاص بك قد ينتج أحد رسائل الخطأ

- `Invalid index`: موضع bit خاطئ (قد يكون سالباً) تم استعماله ك A أو B لأحد استدعاءات الإجراءات أو تم تضمينه في الإجابة المولدة من طرف `construct_instructions(N, M, K)`.
- `Too many instructions`: لا توجد bits ذاكرة كافية لتخزين نتائج التعليمات. بصفة أخرى, لقد استعلمت أكثر من $K - N \cdot M$ من التعليمات.
- `construct_instructions`: `Too many bits in the answer` قامت بتوليد أكثر من 20 bits للإجابة.

إذا لم يقم البرنامج الخاص بك بإنتاج أي رسالة خطأ وقام بحساب العدد الصحيح من الجزر, سيتم الحكم على البرنامج الخاص بك بحكم Accepted ونتيجتك سيتم حسابها على حسب عدد التعليمات المستعملة (أنظر إلى Subtasks). غير ذلك, سيتم الحكم عليه بWrong Answer.

Examples

Example 1

لنفترض $N = 1$, $M = 2$ و $K = 10^7$.

توجد فقط أربع خرائط إدخال محتملة للبرنامج:

- الحالة 1: 0 0
- الحالة 2: 0 1
- الحالة 3: 1 0
- الحالة 4: 1 1

نلاحظ أنه من أجل كل الحالات, عدد الجزر يساوي 1 إذا كان أي من الخلايا $(0,0)$ أو $(0,1)$ خلية برية. وبالتالي, حل ممكن من أجل $N = 1$ و $M = 2$هو تشكيل برنامج عن طريق استعمال تعليمة or وحيدة:

- `append_or(0, 1)`, والذي سيقوم بحساب bitwise-OR للخلايا $(0,0)$ و $(0,1)$ من الخريطة والتي هي مخزنة في الbits 0 و 1 من الذاكرة على التوالي, ويخزن النتيجة في الbit 2 من الذاكرة.

بعد ذلك, سلسلة الإجابة التي تولدها `construct_instructions` هي $[2]$, بما أن الbit 2 من الذاكرة يحتوي على 1 إذا كان هناك بالضبط جزيرة واحدة في الخريطة و 0 في الحالات الأخرى.

Example 2

من أجل $N = 5$, $M = 5$, $K = 10^7$ والخريطة التالية:

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{matrix}$$

عدد الجزر هو 4. وبالتالي, عندما يتم تنفيذ برنامجك الثابت على هاته القيم, الbits التي يتم توليد مواضعها من طرف `construct_instructions` عليها أن تحتوي على 0, 0, و 1 بهذا الترتيب بما أن تمثيل 4 في النظام الثنائي هو 100_2.

Constraints

- $1 \le N, M \le 32$
- $K = 10^7$
- كل خلية من الخريطة هي إما 0 أو 1
- كل الجزر جميلة. بصفة أخرى, من أجل أي خلايا برية تنتمي لنفس الجزيرة وتقع على نفس السطر أو العمود, كل الخلايا التي بينها هي أيضاً خلايا برية.

Subtasks

1. (40 نقطة) كل جزيرة عبارة عن خلية برية واحدة بالضبط.

2. (20 نقطة) من أجل كل جزيرة, توجد أسطر r_1, r_2 وأعمدة c_1, c_2 بحيث الجزيرة مكونة بالضبط من جميع الخلايا (i, j) بحيث $r_1 \le i \le r_2$ و $c_1 \le j \le c_2$. بصفة أخرى, كل جزيرة تشكل مستطيل مملوء من الخلايا البرية.
3. (40 نقطة) لا توجد قيود إضافية.

بافتراض أن برنامجك يحصل على حكم Accepted, ويستعمل I تعليمة. نتيجتك P من أجل حالة الإختبار, اعتماداً على رقم المهمة الفرعية الخاصة به يتم حسابه كالآتي:

- Subtask 1:
  - إذا كان $I \le 5\,030$, $P = 40$.
  - إذا كان $5\,030 < I \le 10\,760$, $P = 22 + 18 \cdot \frac{10\,760 - I}{5\,730}$.
  - إذا كان $10\,760 < I \le 22\,530$, $P = 12 + 10 \cdot \frac{22\,530 - I}{11\,770}$.
  - غير ذلك, $P = 8$.
- Subtask 2:
  - إذا كان $I \le 62\,000$, $P = 20$.
  - غير ذلك, $P = 8$.
- Subtask 3:
  - إذا كان $I \le 9\,100$, $P = 40$.
  - إذا كان $9\,100 < I \le 15\,880$, $P = 22 + 18 \cdot \frac{15\,880 - I}{6\,780}$.
  - إذا كان $15\,880 < I \le 27\,650$, $P = 13 + 9 \cdot \frac{27\,650 - I}{11\,770}$.
  - غير ذلك, $P = 10$.

Sample Grader

يقوم مصحح العينات بقراءة المدخلات بالشكل التالي

- السطر 1: $K\ M\ N$

يتبع هذا بعض حالات الاختبار. كل حالة اختبار تتشكل من N سطر:

- السطر i $(0 \le i < N)$: $g[i][0]\ g[i][1] \ldots g[i][M-1]$ (بينها مسافة)

أين $g[i][j]$ $(0 \le i < N, 0 \le j < M)$ تمثل محتويات الخلية (i, j) من الخريطة.

يتبع وصف جميع حالات الاختبار سطر واحد يحتوي فقط على -1.

يقوم مصحح العينات أولاً باستدعاء `construct_instructions` باستعمال التوقيع الخاص باللغة المستعملة. إذا كان هذا الإستدعاء ينتهك أحد القيود التي تم وصفها في نص المسألة, مصحح العينات يقوم بكتابة أحد رسائل الخطأ الموجودة في قسم "Implementation Details" ويقوم بالخروج.

غير ذلك, مصحح العينات يقوم بمعالجة حالات الإختبار بالترتيب. من أجل كل حالة اختبار, يقوم بتنفيذ البرنامج الذي تم إنشاؤه من طرف `construct_instructions` على حالة الإختبار المدخلة.

من أجل كل تعليمة $print(label, A)$. ليكن l طول A, مصحح العينات يقوم بكتابة محتويات مواضع الbits في A بالشكل التالي:

- $memory[A[l-1]] \ldots memory[A[1]]\ memory[A[0]]$:$label$

بعد تنفيذ جميع التعليمات, يقوم مصحح العينات بكتابة العدد الثنائي الذي تم تشكيله من طرف سلسلة الإجابة V التي ولدها برنامجك من أجل كل حالة اختبار. بطريقة أخرى, من أجل كل حالة اختبار, ليكن l طول V, يقوم مصحح العينات بكتابة $2^0 \cdot memory[V[0]] + 2^1 \cdot memory[V[1]] + \ldots + 2^{l-1} \cdot memory[V[l-1]]$.

بعد تنفيذ جميع حالات الإختبار, يقوم مصحح العينات بكتابة

X :`number of instructions`

أين X هو عدد التعليمات في برنامجك.