International Olympiad in Informatics 2016
12-19th August 2016
Kazan, Russia
day2_2

messy
Country: SYR

تفكيك الخطأ البرمجي المسبب للمشاكل

للبرمجيات ويقوم بتطوير بنى معطيات فعالة. في أحد الأيام ابتكر إلشات بنية معطيات جديدة بنية
هذه قادرة على تخزين إعداد صحيحة غير سالبة مؤلفة من n بت، حيث n هي إحدى قوى العدد 2، أي
من أجل عدد صحيح غير سالب b . $n = 2^b$

ت فارغة في البداية. يجب على أي برنامج يرغب باستخدام بنية المعطيات هذه أن يتبع القواعد التالية:

كن للبرنامج أن يضيف عناصر تحوي أعداد صحيحة \(n\) بت إلى بنية المعطيات، واحد ف كل مرة باستخدام التابع
`add_element(x\)&lrm;`. إذا حاول البرنامج إضاف عناصر موجودة مسبقا في بنية المعطيات لا يحدث أي
شيء.
إضافة آخر عنصر يجب على البرنامج أن يستدعي التابع `compile_set(\)&lrm;` مرة واحدة تماماً.
خيراً يمكن للبرنامج أن يستدعي التابع `check_element(x\)&lrm;` ليتأكد من أن العنصر x موجود في
المعطيات يمكن استخدام هذا التابع أكثر من مرة.

ت بتنفيذ بنية المعيطات أول مرة حصل خطأ برمجي ضمن التابع `compile_set(\)&lrm;`. الخطأ
تيب الخانات الثنائية في كل عنصر من عناصر المموعة بنفس الطريقة. يريد منك إلشات إيجاد الطريقة
ي يتم فيها إعادة ترتيب الخانات بسبب هذا الخطأ البرمجي.

لتكن السلسلة $p = [p_0, \ldots, p_{n1}]$ بحيث أن كل رقم من 0 إلى $n-1$ موجود مرة واحدة فقط، ندعو هذه
سلسلة *تبديل*. لنفرض أحد عناصر السلسلة والذي تكون خاناته في النظام الثنائي هي: $a_0, \ldots, a_{n-1}$ (حيث a_0 هو
ت صاحب أعلى قيمة). عندما يتم استدعاء التابع `compile_set(\)&lrm;`, يتم استبدال هذا العنصر بالعنصر
$a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$.

استخدام نفس التبديل p من أجل إعادة ترتيب خانات كل عنصر من عناصر بنية المعطيات. يمكن لهذا التبديل أن يكون
طياً بما ذلك إمكانية أن يكون $p_i = i$ من أجل $0 \le i \le n-1$.

ل على ذلك لنفعرض أن $n = 4$ $p = [2, 1, 3, 0]$,, وقمت بإدخال الأعداد التي ترميزها الثنائي كالتإلى إلى
مجموعة: `0000`, `1100` و `0111`. إن استدعاء التابع `compile_set` يغير هذه العناصر إلى `0000`, `0101` و
`1110`, بنفس الترتيب.

عليك كتابة برنامج يقوم بإيجاد التبديل p عن طريق التفاعل مع بنية المعطيات لذلك يجب عليك القيام بما يلي وفق
رتيب الموضح:):
موعة من الأعداد الصحيحة والتي يتألف كل منها من n -بتاً,
ذه الأعداد الصحيحة إلى بنية المعطيات.
ستدعاء التابع `compile_set` لتفعيل الخطأ البرمجي,
س وجود بعض العناصر ضمن المجموعة المعدلة,
دة من تلك المعلومات لتحديد التبديل p الذي يتوجب إعادته.

نه يمكن لبرنامجك استدعاء التابع `compile_set` مرة واحدة فقط.

ك حد لعدد المرات التي يمكن لبرنامجك أن يستدعي بها توابع المكتبة بشكل مخصصن يمكن استدعاء
لتابع `add_element` w مرة على الأكثر (w times للقيام بعمليات الكتابة), واستدعاء التابع
`check_element` r مرة على الأكثر (r للقيام بعمليات القراءة).

ا صيل الـتـنجيز

عليك تنجيز التابع التالي (الطريقة)::

- `;int[] restore_permutation(int n, int w, int r\)&lrm`
  - `n`: عدد الخانات الثنائية في التمثيل الثنائي لكل عنصر من المجموعة (وهو بنفس الوقت طول p).
  - `w`: أكبر عدد مسموح لاستدعاء التابع `add_element` التي يمكن لبرنامجك القيام بها.
  - `r`: أكبر عدد مسموح لاستدعاء التابع `check_element` التي يمكن لبرنامجك القيام بها.

  يجب على التابع أن يعيد التبديل المستعاد p.

ن أجل لغة C يمكنك مراجعة النسخة الانكليزية من المسألة.

بع الـمكتبة

جك من التفاعل مع بنية المعطيات يجب أن يقوم البرنامج باستخدام التوابع الثلاثة التالية:

- `;void add_element(string x\)&lrm`

التابع يضيف العنصر الموصوف بالسلسلة النصية `x` إلى المجموعة.

- `x`: سلسلة من `'0'` و `'1'` تمثل التمثيل الثنائي للعدد الصحيح الذي يجب أن تتم إضافته إلى المجموعة ، طول `x` يجب أن يكون n.

- `;void compile_set(\)&lrm`

ع يجب استدعاوه مرة واحدة فقط لا يمكن لبرنامجك استدعاء `;add_element(\)&lrm` بعد استدعاء هذا التابع . لا يمكن لبرنامجك استدعاء `;check_element(\)&lrm` قبل استدعاء هذا التابع.

- `;boolean check_element(string x\)&lrm`

قوم هذا التابع بفحص فيما إذا كان العنصر `x` موجوداً في المجموعة المعدلة.

- `x`: سلسلة نصية مؤلفة من المحارف `'0'` و `'1'` والذي يعطي التمثيل الثنائي للعنصر الذي يجب أن يتم فحصه, يجب أن يكون طول `x` مساوياً n.
- يعيد `true` إذا كان العنصر `x` موجوداً في السلسلة المعدلة, و `false` في حال عدم وجوده.

برنامجك بانتهاك أي من المحددات أعلاه سيكون نتيجة التقييم "جواب خاطئ".

ن يكون المحرف الأول ممثلاً البت صاحب أعلى قيمة للعدد الصحيح الموافق.

م المصحح بتثبيت التبديل p قبل استدعاء التابع `restore_permutation`.

خدام ملفات القالب المعطى من أجل تفاصيل التمجيز الخاصة بلغة البرمجة التي تختارها.

مثال

مصحح باستدعاء التابع بالشكل التالي:

- `;restore_permutation(4, 16, 16\)&lrm`. لدينا $n = 4$ و يمكن للبرنامج أن يقوم بـ 16 عملية كتابة و 16 عملية قراءة.

نامج باستدعاءات التوابع بالشكل التالي:

```
* `add_element("0001"\)&lrm;`
* `add_element("0011"\)&lrm;`
* `add_element("0100"\)&lrm;`
* `compile_set(\)&lrm;`
* `check_element("0001"\)&lrm;` returns `false`
* `check_element("0010"\)&lrm;` returns `true`
* `check_element("0100"\)&lrm;` returns `true`
* `check_element("1000"\)&lrm;` returns `false`
```

```
* `check_element("0011"\)&lrm;` returns `false`
* `check_element("0101"\)&lrm;` returns `false`
* `check_element("1001"\)&lrm;` returns `false`
* `check_element("0110"\)&lrm;` returns `false`
* `check_element("1010"\)&lrm;` returns `true`
* `check_element("1100"\)&lrm;` returns `false`
```

فقط يتوافق مع هذه القيم التي تم إعادتها من التابع `check_element(\)&lrm;`: وهو التبديل $p = [2, 1, 3, 0]$. وهكذا يجب على التابع `restore_permutation` أن يعيد `[2, 1, 3, 0]`.

Subtasks

(20 علامة) $p_i \neq i$, $r = 256$, $w = 256$, $n = 8$ ومن أجل دليلين على الأكثر (i , $0 \leq i \leq n - 1$)
(18 points) $n = 32$, $w = 320$, $r = 1024$,
(11 points) $n = 32$, $w = 1024$, $r = 320$,
(21 points) $n = 128$, $w = 1792$, $r = 1792$,
(30 points) $n = 128$, $w = 896$, $r = 896$.

مثال المصحح

المصحح بقراءة الدخل بالصيغة التالية

السطر 1: الأعداد الصحيحة n r , w ,,
السطر 2: n أعداد صحيحة تمثل عناصر p.