International Olympiad in Informatics 2016



12-19th August 2016 Kazan, Russia day2 2

messy Country: DOM

Unscrambling a Messy Bug

Ilshat es un ingeniero de software trabajando con estructuras de datos eficientes. Un día él inventó una nueva estructura. Esta estructura de datos puede almacenar un conjunto de enteros no negativos de n-bits , donde n es una potencia de dos. Esto es, $n=2^b$ para algún entero b.

La estructura está inicialmente vacía. Un programa que use la estructura debe seguir las siguientes reglas:

- El programa puede añadir elementos que sean enteros de n-bits, uno a la vez, mediante la función add_element(x). Si el programa intenta añadir un elemento que ya está presente en la estructura, nada pasa.
- Después de añadir el último elemento el programa debe llamar a la función compile set() exactamente una vez.
- Finalmente, el programa puede llamar a la función check_element(x) para verificar si un elemento x está presente en la estructura de datos o no. Esta función puede ser usada múltiples veces.

Cuando Ilshat implementó por vez primera esta estructura, él creó un error en la función <code>compile_set()</code>. El error reordena los dígitos binarios de cada elemento en el conjunto de la misma manera. Ilshat quiere que tú encuentres el reordenamiento exacto de los dígitos causado por el error.

Formalmente, considera una secuencia $p=[p_0,\ldots,p_{n-1}]$ en la cual cada número de 0 a n-1 aparece exactamente una vez. Nosotros llamamos a esa secuencia una permutaci'on. Considera un elemento del conjunto, cuyos dígitos en binario son a_0,\ldots,a_{n-1} (con a_0 siendo el bit más significativo). Cuando la función $compile_set()$ es llamada, este elemento es reemplazado por el elemento $a_{p_0},a_{p_1},\ldots,a_{p_{n-1}}$.

La misma permutación p es usada para reordenar los dígitos de cada elemento. La permutación puede ser arbitraria, incluyendo la posibilidad que $p_i=i$ para cada $0\leq i\leq n-1$.

Por ejemplo, supón que n=4, p=[2,1,3,0], y has insertado en el conjunto enteros cuyas representaciones binarias son 0000, 1100 y 0111. Al llamar a la función compile_set, se cambiarían estos elementos por 0000, 0101 y 1110, respectivamente.

Tu tarea es escribir un programa que encuentre la permutación p mediante la interacción con la estructura de datos. Tu programa debe (en el siguiente orden):

1. escoger el conjunto de enteros de n-bits,

- 2. insertar esos enteros en la estructura de datos,
- 3. llamar a la función compile set para provocar el error,
- 4. verificar si algunos elementos están presentes en el conjunto modificado,
- 5. usar esa información para determinar y retornar la permutación p.

Fíjate que tu programa solo puede llamar una vez a la función compile set.

Adicionalmente, hay un límite en el número de veces que tu programa llama a funciones de libería. A saber, puede

- llamar add element a lo sumo w veces (w es para "escrituras"),
- llamar check_element a lo sumo r veces (r es para "lecturas").

Detalles de Implementación

Tú debes implementar una función (método):

- int[] restore permutation(int n, int w, int r)
 - n: el número de bits en la representación binaria de cada elemento en el conjunto (y también la longitud de p).
 - w: el máximo número de operaciones add_element que tu programa puede realizar.
 - r: el máximo número de operaciones check_element que tu programa puede realizar.
 - la función debe retornar la permutación restaurada p.

En el lenguaje C, la definición de la función difiere un poco:

- void restore permutation(int n, int w, int r, int* result)
 - \circ n, w and r tienen el mismo significado de arriba.
 - la función debe retornar la permutación restaurada p mediante el almacenamiento de la misma en el arreglo provisto result: para cada i, debe almacenar el valor de p_i en result[i].

Funciones de librería

Para interactuar con la estructura de datos, tu programa debe usar las siguientes tres funciones (métodos):

void add element(string x)

Esta función añade el elemento descrito por x al conjunto.

- \circ x: es una cadena de caracteres '0' y '1' dando la representación binaria del entero que debe ser añadido al conjunto. La longitud de x debe ser n.
- o void compile_set()
 - Esta función debe ser llamada exactamente una vez. Tu programa no puede llamar add_element() después de esta llamada. Tu programa no puede llamar check element() antes de esta llamada.
- boolean check element(string x)

Esta función verifica si un elemento x está presente en el conjunto modificado.

- x: una cadena de caracteres '0' y '1' dando la representación binaria del entero que debe ser verificado. La longitud de x debe ser n.
- retorna true si el elemento x está en el conjunto modificado, y false en otro caso.

Nota que si tu programa viola cualquier restricción de arriba, la calificación como consecuencia será "Wrong Answer".

Para todas las cadenas, el primer caracter es el bit más significativo en el entero correspondiente.

El grader corrige la permutación p antes que la función $restore_permutation$ es llamada.

Por favor usa las plantillas proporcionadas para los detalles de implementación en tu lenguaje de programación.

Ejemplo

El grader realiza la siguiente llamada a función:

• restore_permutation(4, 16, 16). Tenemos n=4 y el programa puede hacer a lo máximo 16 "escrituras" y 16 "lecturas".

Tu programa realiza la siguiente secuencia de llamadas a funciones:

```
add_element("0001")
add_element("0100")
add_element("0100")
compile_set()
check_element("0001") returns false
check_element("0010") returns true
check_element("1000") returns true
check_element("1000") returns false
check_element("0011") returns false
check_element("0101") returns false
check_element("1001") returns false
check_element("1010") returns false
check_element("1010") returns false
check_element("1010") returns false
```

Solo una permutación es consistente con estos valores retornados por check_element(): la permutación p=[2,1,3,0]. Entonces, restore permutation debe retornar [2, 1, 3, 0].

Subtareas

```
1. (20 puntos) n=8, w=256, r=256, p_i\neq i para al menos 2 índices i ( 0\leq i\leq n-1),
2. (18 puntos) n=32, w=320, r=1024,
3. (11 puntos) n=32, w=1024, r=320,
4. (21 puntos) n=128, w=1792, r=1792,
5. (30 puntos) n=128, w=896, r=896.
```

Grader de ejemplo

El grader de ejemplo lee la entrada en el siguiente formato:

```
\circ línea 1: enteros n, w, r,
```

 $\circ~$ línea 2: n~enteros dando los elementos de $\,p\,.$