Hibajavító

Ilshat egy új tároló adatszerkezetet fejlesztett ki, amelyben n bites *nem negatív* számokat tárol (n kettő hatvány, azaz $n = 2^b$ valamely b nem negatív egsz számra).

A tároló kezdetben üres. A következő műveletek végezhetők:

- Az `add_element(x)` művelet az x elemet beteszi a tárolóba. Ha a tároló tartalmazta az x-et, akkor a művelet hatástalan.
- Az utolsó elem hozzáadása után a `compile_set()` műveletet kell meghívni, pontosan egyszer.
- A `compile_set()` művelet után a `check_element(x)` műveletekkel lekérdezhetjük, hogy az x benne van-e a tárolóban. Ez többször is hívható.

A `compile_set()` művelet hibát tartalmaz, a bitek sorrendjét megváltoztatja. A hiba pontosan úgy írható le, hogy a 0 .. $n-1$ számoknak van olyan $p_0, \ldots, p_{n-1}$ permutációja, amellyel megadható a bitek cseréje. Azaz az $a_0, \ldots, a_{n-1}$ bitsorozatot a $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$ sorozatra cseréli.

A művelet minden, a tárolóban levő elemre ezt a cserét hajtja végre. A permutáció tetszőleges lehet, akár minden $0 \leq i \leq n-1$-re $p_i = i$ is lehet.

Például, ha $n = 4$, $p = [2, 1, 3, 0]$, és a tárolóba a `0000`, `1100` és `0111` bináris alakú számokat tettük, akkor a `compile_set` után ezek lesznek a tárolóban: `0000`, `0101` és `1110`.

Írj programot a p permutáció megkeresésére a következő módszerrel:

1. határozd meg n bites számok egy halmazát,
2. tedd be ezeket a tárolóba,
3. hívd meg a `compile_set` eljárást,
4. ellenőrizd, hogy bizonyos számokat tartalmazza-e a módosított tároló,
5. ezen információk alapján határozd meg a p permutációt!

A `compile_set` csak egyszer hívható.

Az alábbi korlátokat kell betartanod:

- az `add_element` eljárás legfeljebb w-szer hívható (w "írás" lehet),
- a `check_element` eljárás legfeljebb r-szer hívható (r "olvasás" lehet).

Megvalósítás

Egyetlen metódust kell készítened:

- `int[] restore_permutation(int n, int w, int r)`
  - `n`: a bitek száma az elemek bináris reprezentációjában (és a p elemeinek

száma)
    - `w`: az `add_element` hívásainak korlátja.
    - `r`: a `check_element` hívásainak korlátja.
    - a metódus a p permutációt adja eredményül.

C nyelv esetén:

- `void restore_permutation(int n, int w, int r, int* result)`
    - `n`, `w` és `r` ugyanaz, mint fent.
    - a p permutációt a `result` tömb tartalmazza: minden i-re `result[i]`= p_i legyen!

Könyvtári függvények

Az alábbi három függvényt hívhatod:

- `void add_element(string x)`
  Az `x`-et beteszi a tárolóba.
    - `x`: pontosan n darab,`'0'` és `'1'` karaktereket tartalmazó szöveg, az egész szám bináris alakja.

- `void compile_set()`
  Egyszer kell meghívni Utána az `add_element()` nem hívható. Előtte a `check_element()` nem hívható.
- `boolean check_element(string x)`
  Ellenőrzi, hogy az `x` benne van-e a módosított tárolóban.
    - `x`: pontosan n darab,`'0'` és `'1'` karaktereket tartalmazó szöveg, a kérdezett egész szám bináris alakja.
    - értéke `true`, ha az `x` benne van a módosított tárolóban, `false` egyébként.

Ha nem tartod be a követelményeket, az értékelő "Wrong Answer" értékelést ad.

A bináris reprezentációban az első bit a legnagyobb helyiértékű.

Használd a mintában megadott függvényeket!

Példa

Az értékelő így hívja a függvényedet:

- `restore_permutation(4, 16, 16)`. Az $n = 4$ és a programod legfeljebb 16 "írás"-t és 16 "olvasás"-t hívhat.

A programod a következő függvényhívásokat végzi:

- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` eredménye `false`
- `check_element("0010")` eredménye `true`
- `check_element("0100")` eredménye `true`
- `check_element("1000")` eredménye `false`
- `check_element("0011")` eredménye `false`
- `check_element("0101")` eredménye `false`

- `check_element("1001")` eredménye `false`
- `check_element("0110")` eredménye `false`
- `check_element("1010")` eredménye `true`
- `check_element("1100")` eredménye `false`

A kapott értékeknek csak a $p = [2, 1, 3, 0]$ permutáció felel meg, tehát a `restore_permutation` a `[2, 1, 3, 0]` tömböt adja eredményül.

Részfeladatok

1. (20 pont) $n = 8$, $w = 256$, $r = 256$, $p_i \neq i$ legfeljebb 2 i-re ($0 \leq i \leq n - 1$),
2. (18 pont) $n = 32$, $w = 320$, $r = 1024$,
3. (11 pont) $n = 32$, $w = 1024$, $r = 320$,
4. (21 pont) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 pont) $n = 128$, $w = 896$, $r = 896$.

Minta értékelő

A minta értékelő a következőket olvassa:

- 1 . sor: az n, w, r egész számok,
- 2 . sor: n egész szám, a p permutáció elemei.