Sudėtingos klaidos radimas

Ilšatas, programų sistemų inžinierius, projektuoja efektyvias duomenų struktūras. Vieną dieną jis sukūrė naują duomenų struktūrą. Ši struktūra gali saugoti n bitų neneigiamų sveikųjų skaičių aibę, beto n yra dvejeto laipsnis, t.y. $n = 2^b$ kuriam nors sveikajam skaičiui b.

Pradžioje duomenų struktūra tuščia. Programa, kuri naudoja šią struktūrą, turi veikti pagal tokias taisykles:

- Į duomenų struktūrą programa gali dėti elementus, kurie yra n bitų sveikieji skaičiai. Skaičiai dedami po vieną iš eilės pasinaudojant funkcija `add_element(x)`. Jei programa bando įdėti elementą, jau esantį duomenų struktūroje, nieko neįvyksta.
- Įdėjus paskutinį elementą, programa privalo lygiai vieną kartą iškviesti funkciją `compile_set()`.
- Norėdama patikrinti ar elementas x yra duomenų struktūroje, programa gali kviesti funkciją `check_element(x)`. Ši funkcija gali būti kviečiama daug kartų.

Pirmą kartą realizuodamas šią duomenų struktūrą, Ilšatas įvėlė klaidą funkcijoje `compile_set()`. To pasėkoje, funkcija perdėlioja kiekvieno struktūros elemento bitus tuo pačiu būdu. Ilšatas nori nustatyti, kaip jo programa perdėlioja bitus.

Nagrinėkime seką $p = [p_0, \ldots, p_{n-1}]$, kurioje kiekvienas elementas nuo 0 iki $n-1$ sutinkamas lygiai vieną kartą. Tokia seka vadinama *perstata* (angl. permutation). Panagrinėkime elementą, kurio dvejetainiai skaitmenys yra $a_0, \ldots, a_{n-1}$ (čia a_0 yra labiausiai reikšminis bitas). Kviečiant funkciją `compile_set()`, šis elementas bus pakeistas elementu $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$.

Ta pati perstata p yra naudojama perdėliojant kiekvieno elemento bitus. Perdėlioti galima bet kaip, taip pat ir taip: $p_i = i$ kiekvienam $0 \le i \le n-1$.

Pavyzdžiui, $n = 4$, $p = [2, 1, 3, 0]$, o į aibę buvo įdėti sveikieji skaičiai, kurių dvejetainė išraiška `0000`, `1100` ir `0111`. Iškviesta funkcija `compile_set` pakeis šiuos elementus atitinkamai į `0000`, `0101` ir `1110`.

Parašykite programą, kuri rastų perstatą p kviesdama šios duomenų struktūros funkcijas. Programa turi atlikti šiuos veiksmus nurodyta tvarka:

1. pasirinkti n bitų sveikųjų skaičių aibę,
2. įdėti šiuos skaičius į duomenų struktūrą,
3. iškviesti funkciją `compile_set`, kuri dėl klaidos modifikuos pradinę skaičių aibę,
4. patikrinti kai kurių elementų buvimą modifikuotoje aibėje,
5. naudojantis gauta informacija rasti perstatą p.

Atkreipkite dėmesį, kad programa gali kviesti funkciją `compile_set` tik vieną kartą.

Ribojimas ir kitų funkcijų kvietimų skaičius:
- `add_element` kviečiama ne daugiau kaip w kartų (w žymi "įrašymą"),
- `check_element` kviečiama ne daugiau kaip r kartų (r žymi "skaitymą").

Realizacija

Parašykite funkciją (metodą):
- `int[] restore_permutation(int n, int w, int r)`
  - `n`: skaičius bitų, reikalingų užrašyti aibės elementus dvejetaine išraiška (taipogi perstatos p ilgis).
  - `w`: maksimalus `add_element` kvietimų skaičius.
  - `r`: maksimalus `check_element` kvietimų skaičius.
  - funkcija turi grąžinti rastą perstatą p.

C kalba šios funkcijos prototipas atrodo taip:
- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w` ir `r` reiškią tą patį, ką ir aukščiau.
  - funkcija turi grąžinti rastą perstatą p užrašydama ją į masyvą `result`: kiekvienam i ji turi įrašyti p_i reikšmę į `result[i]`.

Bibliotekos funkcijos

Kad galėtų sąveikauti su duomenų struktūra, jūsų programa turi naudoti šias funkcijas (metodus):
- `void add_element(string x)`
  Funkcija įdeda elementą `x` į aibę.
  - `x`: simbolinė eilutė, sudaryta iš `'0'`'ių ir `'1'`'ų. Ji vaizduoja sveikąjį skaičių, kuris turi būti įdėtas į aibę. `x` ilgis privalo būti n.

- `void compile_set()`
  Ši funkcija privalo būti iškviesta lygiai vieną kartą. Po šios funkcijos iškvietimo programa nebegali kviesti `add_element()`. Prieš šios funkcijos kvietimą programa negali kviesti `check_element()`.
- `boolean check_element(string x)`
  Ši funkcija patikrina ar elementas `x` yra modifikuotoje aibėje.
  - `x`: simbolinė eilutė, sudaryta iš `'0'`'ių ir `'1'`'ų. Ji vaizduoja sveikąjį skaičių, kurio buvimas aibėje bus tikrinamas. `x` ilgis privalo būti n.
  - grąžina `true`, jei elementas `x` yra modifikuotoje sekoje, kitu atveju — `false`.

Atkreipkite dėmesį, kad programai pažeidus vieną iš anksčiau aprašytų ribojimų, jos vertinimo rezultatas bus "Wrong Answer".

Pirmasis visų simbolinių eilučių simbolis atitinka labiausiai reikšminį atitinkamo skaičiaus bitą.

Prieš kviesdama `restore_permutation`, vertinimo programa fiksuoja konkrečią perstatą p.

Pateiktuose failų šablonuose rasite realizacijai reikalingą informaciją. Pasirinkite šabloną, atitinkantį jūsų programavimo kalbą.

Pavyzdys

Vertinimo programai atlikus kvietimą:

- `restore_permutation(4, 16, 16)`. $n = 4$, o programa gali atlikti ne daugiau kaip 16 "įrašymų" ir 16 "skaitymų".

Programa atlieka žemiau pateiktus kvietimus:

- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` grąžina `false`
- `check_element("0010")` grąžina `true`
- `check_element("0100")` grąžina `true`
- `check_element("1000")` grąžina `false`
- `check_element("0011")` grąžina `false`
- `check_element("0101")` grąžina `false`
- `check_element("1001")` grąžina `false`
- `check_element("0110")` grąžina `false`
- `check_element("1010")` grąžina `true`
- `check_element("1100")` grąžina `false`

Tik viena perstata atitinka `check_element()` grąžintas reikšmes — $p = [2, 1, 3, 0]$. Todėl `restore_permutation` turi grąžinti `[2, 1, 3, 0]`.

Dalinės užduotys

1. (20 taškų) $n = 8$, $w = 256$, $r = 256$, $p_i \neq i$ ne daugiau kaip dviem skirtingiems i ($0 \leq i \leq n - 1$),
2. (18 taškų) $n = 32$, $w = 320$, $r = 1024$,
3. (11 taškų) $n = 32$, $w = 1024$, $r = 320$,
4. (21 taškas) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 taškų) $n = 128$, $w = 896$, $r = 896$.

Pavyzdinė vertinimo programa.

Pavyzdinė vertinimo programa skaito duomenis tokiu formatu:

- 1-oji eilutė: sveikieji skaičiai n, w, r,
- 2-oji eilutė: n sveikųjų skaičių, aprašančių perstatos p elementus.