Күрделі қатені талдау

Ильшат бағдарламалық қамтамасыз ету инженері болып жұмы істейді және қазір деректер құрылымымен жұмыс істейді. Ол жаңа деректер құрылымын ойлап шығарды. Бұл деректер құрылымы n биттен тұратын *теріс емес* сандар жиынын сақтай алады, мұндағы n саны 2 санының дәрежесі. Нақтырақ, кейбір теріс емес бүтін b саны үшін $n = 2^b$.

Деректер құрылымы бастапқыда бос. Осы деректер құрылымын қолданатын программа келесі ережелерге еру қажет:

- Программа деректер құрылымына `add_element(x)` функциясын қолданып n биттен тұратын сандарды біреуден қоса алады. Егер программа деректер құрылымында бар санды қосуға тырысса ештеңе өзгермейді, деректер құрылымы өз қалпында қалады.
- Барлық қосу керек элементтерді қосып болғаннан кейін `compile_set()` функциясын шақыру керек, программа жұмысы барысында бұл функцияны тек бір рет қана қолдана алады.
- Соңында, программа `check_element(x)` функциясы арқылы x санының деректер құрылымындағы сандар жиынының құрамында бар-жоғын тексере алады. Программа бұл функцияны бірнеше рет қолдана алады.

Ильшат осы деректер құрылымын бірінші рет іске асырғанда `compile_set()` функциясында қателік жасап қойды. Осы қате деректер құрылымына енгізілген әр санның биттерін белгілі бір тәсілмен орындарын ауыстырады. Ильшат сізден осы қатенің кесірінен пайда болған ауыстырудың нақты тәсілін тауып беруді сұрайды.

Нақтырақ, 0 -ден $n-1$ -ге дейінгі сандардың әрқайсысы дәл бір реттен ғана кездесетін $p_0, \ldots, p_{n-1}$ сандар қатарын қарастырайық. Біз осы сандар қатарын *ауыстыру* деп атаймыз. Деректер құрылымындағы екілік санау жүйесіндегі цифрлары $a_0, \ldots, a_{n-1}$ болатын санды қарастырайық (a_0 ең бірінші цифры). `compile_set()` функциясы қолданылған уақытта осы сан екілік санау жүйесіндегі цифрлары $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$ болатын санға ауыстырылады.

Дәл осы ауыстыру деректер құрылымындағы әрбір санға қолданылады. Ауыстыру кез келген болуы мүмкін, олардың құрамына әрбір $0 \le i \le n-1$ үшін $p_i = i$ болатын ауыстыру да кіреді.

Мысалы, $n = 4$, $p = [2, 1, 3, 0]$ болсын және сіз деректер құрылымына `0000`, `1100` және `0111` сандарын еңгіздіңіз делік. `compile_set` функциясы осы сандарды сәйкесінше `0000`, `0101` және `1110` сандарына ауыстырады.

Сіздің тапсырмаңыз деректер құрылымымен әрекеттер жасай отырып p ауыстыруын табатын программа жазу. Сіздің программаңыз (берілген ретпен):

1. n биттен тұратын бүтін сандар жиынын ойлап табу,
2. ойлаған сандар жиынын деректер құрылымына енгізу,
3. қате орындалған функция іске асу үшін `compile_set` функциясын шақыру,
4. кей сандардың жаңартылған деректер құрылымының жиынында бар-жоғын `check_element` функциясы арқылы тексеру,
5. пайда болған ақпараттарды пайдаланып p ауыстыруын тауып, қайтару керек.

Сіздің программаңыз `compile_set` функциясын тек бір рет қолдана алатынын байқаңыз.

Оған қоса, сіздің программаңыз берілген әрбір функция үшін қолдану санына шектеу бар. Нақтырақ,

- программа `add_element` функциясын көбінде w рет қолдана алады (w "writes" сөзінен),
- программа `check_element` функциясын көбінде r рет қолдана алады (r "reads" сөзінен).

Қосымша ақпарат

Сіз келесі функцияны іске асыру қажетсіз:

- `int[] restore_permutation(int n, int w, int r)`
  - `n`: сандар жиынының әрбір санының екілік санау жүйесіндегі цифрлар саны (және p ауыстыруының ұзындығы).
  - `w`: программаның `add_element` функциясының қолдану санына шектеу.
  - `r`: программаның `check_element` функциясының қолдану санына шектеу.
  - функция p ауыстыруын қайтару керек.

C программалау тілінде функция прототипі кішкене өзгеше:

- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w` және `r` мағыналары өзгертілмеген.
  - функция p ауыстыруын ұсынылған `result` массивіне жазып қайтару керек: әр i үшін p_i мәні `result[i]` айнымалысына жазылуы қажет.

Берілген функциялар

Программа деректер құрылымымен әрекет жасауы үшін келесі үш функцияны қолдануы қажет:

- `void add_element(string x)`
  Бұл функция деректер құрылымының сандар жиынына `x` санын қосады.
  - `x`: `'0'` және `'1'` символдарынан тұратын жол, жиыныға қосылатын санның екілік санау жүйесінде жазылуы. Жолдың ұзындығы n болуы тиіс.

- `void compile_set()`
  Бұл функция дәл бір рет шақырылуы тиіс. Сіздің программаңыз бұл функцияны қолданғаннан кейін `add_element()` функциясын қолдана

алмайды және бұл функцияға дейін `check_element()` функциясын қолдана алмайды.
- `boolean check_element(string x)`
  Бұл функция `x` элементінің жаңартылған деректер құрылымының сандар жиынында бар-жоғын тексереді.
  - `x`: `'0'` және `'1'` символдарынан тұратын жол, тексерілетін санның екілік санау жүйесінде жазылуы. Жолдың ұзындығы n болуы тиіс.
  - егер `x` элементі жаңартылған жиын құрамында бар болса `true` қайтарады, басқа жағдайларда `false` қайтарады.

Егер сіздің программаңыз жоғарыдағы шектеулердің кез келгенін бұзатын болса сіз "Wrong Answer" жауабын аласыз.

Барлық жолдар үшін ең бірінші символы санның екілік санау жүйесіндегі ең бірінші цифрын көрсетеді.

Анығырақ ақпарат үшін сіздің қолданатын программалау тілі үшін үлгі файлын қолдануыңызды ұсынамыз.

Мысал

Бағалаушы келесі функцияны шақырады:
- `restore_permutation(4, 16, 16)`. Бізде $n = 4$ және программа көп дегенде 16 сан енгізіп, 16 сан тексере алады.

Программа келесі функцияларды шақырады
- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` `false` қайтарады
- `check_element("0010")` `true` қайтарады
- `check_element("0100")` `true` қайтарады
- `check_element("1000")` `false` қайтарады
- `check_element("0011")` `false` қайтарады
- `check_element("0101")` `false` қайтарады
- `check_element("1001")` `false` қайтарады
- `check_element("0110")` `false` қайтарады
- `check_element("1010")` `true` қайтарады
- `check_element("1100")` `false` қайтарады

`check_element()` функциясымен қайтарылған мәндер үшін тек қана бір ауыстыру сәйкес келеді: $p = [2, 1, 3, 0]$. Сондықтан, `restore_permutation` функциясы `[2, 1, 3, 0]` массивін қайтару қажет.

Есеп бөлімдері

1. (20 ұпай) $n = 8$, $w = 256$, $r = 256$, ($0 \leq i \leq n - 1$) болатын көп дегенде 2 i индексі үшін $p_i \neq i$,
2. (18 ұпай) $n = 32$, $w = 320$, $r = 1024$,

3. (11 ұпай) $n = 32$, $w = 1024$, $r = 320$,
4. (21 ұпай) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 ұпай) $n = 128$, $w = 896$, $r = 896$.

Мысалдар бағалаушы

Мысал бағалаушы енгізу элементтері ретінде мәндерді келесі форматта оқиды:

- 1 жол: n, w, r бүтін сандары,
- 2 жол: p ауыстыруын құрайтын n бүтін сандар.