Rydde opp i en rotete bug

Ilshat en er programmerer som jobber på effektive datastrukturer. En dag så oppfant han en ny datastruktur. Datastrukturen kan inneholde et sett med *ikke-nevative* n-bit heltall, hvor n er en toerpotens. Dvs. $n = 2^b$ for et eller annet ikke-negativt heltall b.

Til å begynne med så er datastrukturen tom. Et program som skal bruke datastrukturen må følge disse reglene:

- Programmet kan legge til n-bits heltall i datastrukturen, en av gangen, ved å kalle funksjonen `add_element(x)`. Dersom programmet prøver å legge til et element som allerede eksisterer i datastrukturen så skjer ingenting.
- Etter å ha lagt til det siste elementet så skal programmet kalle funksjonen `compile_set()` nøyaktig én gang.
- Deretter så kan programmet kalle funksjonen `check_element(x)` for å sjekke om et element x er inneholdt i datastrukturen. Denne funksjonen kan kalles flere ganger.

Når Ilshat først implementerte denne datastrukturen så hadde han en bug i funksjonen `compile_set()`. Buggen gjør at de binære sifrene av alle elementene i settet blir stokket om, på samme måte. Ilshat vil at du skal finne den nøyaktige omstokkingen av sifrene som buggen har forårsaket.

Formelt, betrakt en sekvens $p = [p_0, \ldots, p_{n-1}]$ hvor hvert tall fra 0 til $n-1$ forekommer nøyaktig én gang. Vi kaller en slik sekvens en *permutasjon*. Betrakt så et element i settet hvis binære sifre er $a_0, \ldots, a_{n-1}$ (hvor a_0 er den mest signifikante bitten). Når fuksjonen `compile_set()` blir kalt så blir dette elementet erstattet med elementet $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$.

Den samme permutasjonen p blir brukt til å stokke om på sifrene i alle elementene. Permutasjonen kan være hva som helst, inkludert situasjonen der $p_i = i$ for alle $0 \leq i \leq n-1$.

For eksempel, gitt at $n = 4$, $p = [2, 1, 3, 0]$, og at du har satt heltallene med binære representasjoner `0000`, `1100` og `0111` i datastrukturen. Når funksjonen `compile_set` blir kalt, så blir disse elementene endret til henholdsvis `0000`, `0101` og `1110`.

Din oppgave er å skrive et program som finner permutasjonen p ved å bruke datastrukturen. Programmet ditt skal (i følgende rekkefølge):

1. Velge et sett med n-bits heltall
2. Sette disse heltallene i datastrukturen

3. Kalle funksjonen `compile_set` for å utløse buggen
4. Sjekke noen elementer, for å se om de er inneholdt i det modifiserte settet
5. Bruke denne informasjonen til å finne permutasjonen p og returnere den.

Merk at programmet ditt bare kan kalle funksjonen `compile_set` én gang.

I tillegg så er det begrensinger på hvor mange ganger programmet ditt kan kalle de andre biblioteksfunksjonene.

- Programmet kan kalle `add_element` maksimalt w ganger (w for "writes"),
- Programmet kan kalle `check_element` maksimalt r ganger (r for "reads").

Implementasjonsdetaljer

Du skal implementere én funksjon:

- `int[] restore_permutation(int n, int w, int r)`
  - `n`: antall bits i den binære representasjonen av elementene i settet (og også lengden av p).
  - `w`: det maksimale antall ganger programmet ditt kan kalle `add_element`.
  - `r`: det maksimale antall ganger programmet ditt kan kalle `check_element`.
  - funksjonen skal returnere permutasjonen p.

I C, så er protypen litt annerledes:

- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w` and `r` har samme betydning som over.
  - Funksjonen skal returnere permutasjonen p ved å lagre den i den gitte arrayen `result`: for hver i så skal den lagre verdien p_i i `result[i]`.

Biblioteksfunksjoner

For å benytte datastrukturen så skal programmet bruke disse tre funksjonene:

- `void add_element(string x)`
  Dnne funksjonen legger til elementet beskrevet med `x` til i settet.
  - `x`: en streng av `'0'` and `'1'` tegn, som gir den binære representasjonen av et heltall som skal legges til i settet. Lengden av `x` må være n.

- `void compile_set()`
  Denne funksjonen må kalles nøyaktig én gang. Programmet ditt kan ikke kalle `add_element()` etter dette kallet. Programmet ditt kan ikke kalle `check_element()` før dette kallet.
- `boolean check_element(string x)`
  Denne funksjonen sjekker om elementet `x` er i dette modifiserte settet.
  - `x`: en streng av `'0'` and `'1'` tegn, som gir den binære representasjonen av et heltall som skal legges til i settet. Lengden av `x` må være n.
  - returnerer `true` dersom elementet `x` er i det modifiserte settet, og `false` ellers.

Merk at dersom programmet ditt bryter noen av restriksjonene over, så vil graderen alltid gi svar "Wrong Answer".

For alle strengene så er det første tegnet den mest signifikante bitten i det

tilsvarende heltallet.

Graderen fikser permutasjonen p før funksjonen `restore_permutation` er kalt.

Bruk de gitte templatfilene for detaljer om implementasjonen i ditt språk.

Eksempel

Graderen gjør dette kallet:

- `restore_permutation(4, 16, 16)`. Vi har $n = 4$, og programmet kan gjøre maksimalt 16 "writes" og 16 "reads".

Programmet gjør følgende kall:

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

Bare én permutasjon er stemmer overens med disse resultatene fra `check_element()`: permutasjonen $p = [2, 1, 3, 0]$. Derfor skal `restore_permutation` returnere `[2, 1, 3, 0]`.

Subtasks

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

Sample grader

The sample graderen leser input i følgende format:

- linje 1: heltall n, w, r,
- linje 2: n heltall som gir elementene av p.