Bit sürüşdürmə registerləri

Mühəndis Kristofer yeni bir kompyuter prosessoru üzərində işləyir.

Prosessorun, **register** adlanan m sayda (0'dan $m-1$'ə qədər nömrələnmiş) b-bit yaddaş xanalarına əli çatır ($m = 100,\ b = 2000$). Register'ləri $r[0], r[1], \ldots, r[m-1]$ olaraq işarələyirik. Hər bir register, 0'dan (ən sağdakı bit) $b-1$'ə (ən soldakı bit) nömrələnmiş b ölçülü bit massividir. Hər i $(0 \le i \le m-1)$ və j $(0 \le j \le b-1)$ üçün, i'ci register'in j'ci bitini $r[i][j]$ ilə göstəririk.

İstənilən $d_0, d_1, \ldots, d_{l-1}$ bit ardıcıllığı üçün (l ixtiyari biri ölçüdür), ardıcılığın **tam qiyməti** $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$ ilə hesablanır. **Register'in tam qiyməti**, onun bit ardıcıllığının tam qiymətinə bərabərdir, yəni $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

Register'in bitləri üzərində dəyişiklik etmək üçün prosessora 9 növ **instruksiya** vermək olar. Hər bir instruksiya bir və ya daha çox register üzərində əməliyyat aparır və alınan cavabı register'lərin birində saxlayır. Aşağıda, $x := y$ ifadəsini, x'in dəyərini y'ə bərabər etmək üçün istifadə etmişik. İnstruksiyaların yerinə yetirdiyi əməliyyatlar aşağıdakı şəkildədir:

- $move(t, y)$: y'ci registerin bitlərini t'ci registerə kopyalamaq. Hər j $(0 \le j \le b-1)$ üçün, $r[t][j] := r[y][j]$.
- $store(t, v)$: t registerini v'yə bərabər elə, burada v, b ölçülü bit massividir. Hər j $(0 \le j \le b-1)$ üçün, $r[t][j] := v[j]$.
- $and(t, x, y)$: x və y register'ləri üzərində bit-bit AND (bitwise-AND) əməliyyatı yerinə yetirib alınan cavabı t registerində saxlamaq. Hər j $(0 \le j \le b-1)$ üçün, əgər $r[x][j]$ **və** $r[y][j]$ 1'ə bərabərdirsə $r[t][j] := 1$, əks halda $r[t][j] := 0$.
- $or(t, x, y)$: x və y register'ləri üzərində bit-bit OR (bitwise-OR) əməliyyatı yerinə yetirib alınan cavabı t registerində saxlamaq. Hər j $(0 \le j \le b-1)$ üçün, əgər $r[x][j]$ **və ya** $r[y][j]$ 1'ə bərabərdirsə $r[t][j] := 1$, əks halda $r[t][j] := 0$.
- $xor(t, x, y)$: x və y register'ləri üzərində bit-bit XOR (bitwise-XOR) əməliyyatı yerinə yetirib alınan cavabı t registerində saxlamaq. Hər j $(0 \le j \le b-1)$ üçün, əgər $r[x][j]$ və $r[y][j]$'dən **yalnızca biri** 1'ə bərabərdirsə $r[t][j] := 1$, əks halda $r[t][j] := 0$.
- $not(t, x)$: x register'i üzərində bit-bit NOT (bitwise-NOT) əməliyyatı yerinə yetirib alınan cavabı t registerində saxlamaq. Hər j $(0 \le j \le b-1)$ üçün, $r[t][j] := 1 - r[x][j]$.
- $left(t, x, p)$: x registerindəki bütün bitləri p vahid sola sürüşdürmək və alınan cavabı t registerində saxlamaq. Alınan cavabı b ölçülü v massivi olaraq işarələyək. Hər j $(0 \le j \le b-1)$ üçün, $j \ge p$ olarsa $v[j] = r[x][j-p]$, əks halda $v[j] = 0$. Hər j $(0 \le j \le b-1)$ üçün, $r[t][j] := v[j]$.

- $right(t, x, p)$: x registerindəki bütün bitləri p vahid sağa sürüşdürmək və alınan cavabı t registerində saxlamaq. Alınan cavabı b ölçülü v massivi olaraq işarələyək. Hər j $(0 \leq j \leq b-1)$ üçün, $j \leq b-1-p$ olarsa $v[j] = r[x][j+p]$, əks halda $v[j] = 0$. Hər j $(0 \leq j \leq b-1)$ üçün, $r[t][j] := v[j]$.

- $add(t, x, y)$: x və y registerlərində saxlanılan dəyərlərin cəmini tapmaq və alınan cavabı t registerində saxlamaq. Alınan cəmin 2^byə qalığı nəzərə alnacaq. Daha dəqiq desək, x registerində saxlanılan dəyərin X, y registerində saxlanılan dəyərin Y olduğunu fərz edək. Əməliyyatdan sonra t registerində alınan dəyərin T olsun. Onda, əgər $X + Y < 2^b$ olarsa, $T = X + Y$. Əks halda, $T = X + Y - 2^b$.

Kristofer səndən yeni prosessoru istifadə edərək 2 növ tapşırıq yerinə yetirməyini istəyir. Tapşırığın növü s ilə işarələnib. Hər iki tapşırıq üçün, yuxarıda verilmiş instruksiyalardan ibarət olan bir **proqram** tələb olunur.

Proqrama verilən **giriş verilənləri**, hər birində k bit olan n ədəddən ($a[0], a[1], \ldots, a[n-1]$) ibarətdir. Yəni, $a[i] < 2^k$ ($0 \leq i \leq n-1$). Proqram başlamazdan əvvəl, bütün giriş verilənləri 0'cı registerdə ardıcıl şəkildə saxlanılır. Belə ki, hər i $(0 \leq i \leq n-1)$ üçün, $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ ardıcıllığının dəyəri $a[i]$'ə bərabərdir. Qeyd etmək lazımdır ki, $n \cdot k \leq b$. 0'cı registerdəki digər bütün bitlər (yəni $n \cdot k$ və $b-1$ arasında, özləri də daxil olmaqla, bütün bitlər), həmçinin digər registerlərdəki bütün bitlər başda 0'a bərabərdir.

Proqramı işlətmək, onun instruksiyalarını ardıcıl şəkildə yerinə yetirməkdir. Son instruksiya yerinə yetirildikdən sonra, proqramın **nəticəsi** 0'cı registerdəki bitlərin son qiymətinə nəzərən hesablanır. Daha dəqiq olsaq, alınan nəticə n uzunluqlu $c[0], c[1], \ldots, c[n-1]$ tam ədədlərə ardıcıllığıdır. Burada hər i ($0 \leq i \leq n-1$) üçün, $c[i]$nin dəyəri 0'cı registerin $i \cdot k$'dən $(i+1) \cdot k - 1$'ə qədər olan bit ardıcıllığının tam qiymətinə bərabərdir. Proqram bitdikdən sonra, 0'cı registerin yerdə qalan bitləri və yerdə qalan registerlərin bitləri hər hansı dəyərləri ala bilərlər.

- Birinci tapşırıq $(s = 0)$ verilən $a[0], a[1], \ldots, a[n-1]$ ədədləri arasından ən kiçik ədədi tapmaqdır. Yəni, $c[0]$ dəyəri $a[0], a[1], \ldots, a[n-1]$ ədədləri arasından minimum ədədin qiymətinə bərabər olacaq. Digər $c[1], c[2], \ldots, c[n-1]$ ədədləri ixtiyari qiymət ala bilər.

- İkinci tapşırıq $(s = 1)$ verilən $a[0], a[1], \ldots, a[n-1]$ ədədlərini azalmayan şəkildə sıralamaqdır. Yəni, hər i ($0 \leq i \leq n-1$) üçün, $c[i]$ dəyəri $a[0], a[1], \ldots, a[n-1]$ ədədləri arasından $1+i$-ci ən kicik ədədə bərabər olacaq.(məsələn, $c[0]$ verilən ədədlər arasındakı ən kiçik ədədə bərabər olacaq).

Kristoferə bu problemləri həll edə biləcək, hər biri ən çox q instruksiyadan ibarət olan proqramlar yazın.

İmplementasiya detalları

Siz aşağıdakı proseduru (funksiyanı) yerinə yetirməlisiz:

```
void construct_instructions(int s, int n, int k, int q)
```

- s: tapşırığın növü.
- n: giriş verilənlərinin sayı.
- k: hər giriş verilənin içindəki bitlərin sayı.
- q: icazə verilən maksimum instruksiya sayı.
- Bu prosedur bir dəfə çağırılır və verilmiş tapşırığı yerinə yetirəcək instruksiya ardıcıllığı yaratmalıdır.

Bu prosedur instruksiya ardıcıllığı yaratmaq üçün aşağıdakı prosedurları bir və ya daha çox dəfə çağırmalıdır:

```
void append_move(int t, int y)
void append_store(int t, bool[] v)
void append_and(int t, int x, int y)
void append_or(int t, int x, int y)
void append_xor(int t, int x, int y)
void append_not(int t, int x)
void append_left(int t, int x, int p)
void append_right(int t, int x, int p)
void append_add(int t, int x, int y)
```

- Hər prosedur uyğun olaraq $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ or $add(t, x, y)$ instruksiyalarını proqramın ardıcıllığına əlavə edir.
- Bütün uyğun prosedurlar üçün, t, x, y ən az 0 və ən çox $m - 1$ ola bilər.
- Bütün instruksiyalar üçün, t, x, y müxtəlif olmaya bilər.
- $left$ və $right$ instruksiyaları üçün, p ən az 0 və ən çox b ola bilər.
- $store$ instruksiyası üçün, v'nin uzunluğu b olmalıdır.

Həllinizdə kömək olması üçün növbəti proseduru da çağıra bilərsiniz:

```
void append_print(int t)
```

- Həllinizin yoxlanılması zamanı bu prosedura olan çağırılmalar nəzərə alınmayacaq.
- Nümunə grader'də bu prosedur proqramln instruksiya ardıcıllığına $print(t)$ əlavə edir.
- Nümunə grader $print(t)$ instruksiyası ilə qarşılaşdıqda, t'ci registerin ilk $n \cdot k$ bitindən yaranan, k-bit n ədəd çıxışa verir. (Daha ətraflı, Nümunə Grader bölməsində)
- $0 \leq t \leq m - 1$ şərti ödənməlidir.
- Bu prosedurun çağırılması, istifadə olunmuş instruksiya sayında nəzərə alınmır.

Son instruksiyanı yerinə yetirdikdən sonra, `construct_instructions` nəticələnməlidir. Daha sonra proqram bir neçə test ilə yoxlanılır, hər bir testdə n dənə k-bit ədəd $a[0], a[1], \ldots, a[n - 1]$ olur. Əgər proqramın nəticəsindəki $c[0], c[1], \ldots, c[n - 1]$ ədədləri aşağıdakı şərtləri ödəyərsə, sizin həlliniz düzgün sayılır:

- $s = 0$, $c[0]$ dəyəri, $a[0], a[1], \ldots, a[n - 1]$ arasındakı ən kiçik ədədə bərabər olmalıdır.

- $s = 1$, hər i ($0 \leq i \leq n - 1$) üçün, $c[i]$ dəyəri, $a[0], a[1], \ldots, a[n - 1]$ ədədləri arasında $i + 1$'ci ən kiçik ədədə bərabər olmalıdır.

Həllinizin yoxlanılması zamanı növbəti mesajlar gələ bilər:

- `Invalid index`: hansısa prosedurun çağırılması zamanı, register indeksi olaraq t, x or y dəyişənlərinin biri üçün yanlış ədəd verilib (mənfi ədəd də ola bilər məsələn).
- `Value to store is not b bits long`: `append_store` proseduruna göndərilən $v'nin ölçüsü b'yə bərabər deyil.
- `Invalid shift value`: `append_left` və ya `append_right` prosedurlarına göndərilən p dəyəri 0 və b arasında deyil.
- `Too many instructions`: q'dən daha çox instruksiya istifadə etmisiniz.

Nümunələr

Nümunə 1

Fərz edin ki $s = 0$, $n = 2$, $k = 1$, $q = 1000$. İki giriş veriləni var: $a[0]$ and $a[1]$, hər biri $k = 1$ bitdir. Proqram başlamazdan əvvəl, $r[0][0] = a[0]$ and $r[0][1] = a[1]$ olur. Prosessordakı digər bütün bitlər 0'a bərabərdir. Proqramdakı bütün instruksiyalar bitdikdən sonra $c[0] = r[0][0] = \min(a[0], a[1])$ olmalıdır.

4 mümkün hal var:

- Hal 1: $a[0] = 0, a[1] = 0$
- Hal 2: $a[0] = 0, a[1] = 1$
- Hal 3: $a[0] = 1, a[1] = 0$
- Hal 4: $a[0] = 1, a[1] = 1$

Hər 4 hal üçün, $\min(a[0], a[1])$ dəyərinin $a[0]$ və $a[1]$ ədədlərinin bit-bit AND əməliyyatı nəticəsində yarnan cavabına bərabər olduğunu görə bilərik. Buna görə də, növbəti çağırışları edərək proqramı yarada bilərik:

1. `append_move(1, 0)`, $r[0]$ dəyərini $r[1]$'ə kopyalamaq instruksiyasını proqrama əlavə edir.
2. `append_right(1, 1, 1)`, $r[1]$ içindəki bütün bitləri götürüb, onları 1 vahid sağa sürüşdürüb, alınan cavabı yenidən $r[1]$'ə geri yazan instruksiyanı proqrama əlavə edir. Hər bir ədəd 1 bit uzunluğunda olduğu üçün, $r[1][0]$ = $a[1]$ olacaq.
3. `append_and(0, 0, 1)`, $r[0]$ və $r[1]$ ədədləri üzərində AND əməliyyatı aparıb, alınan cavabı $r[0]$'ə yazan instruksiyanı proqrama əlavə edir. Bundan sonra, $r[0][0]$ dəyəri $r[0][0]$ və $r[1][0]$ ədədlərinin bit-bit AND qiymətinə bərabər olur, eyni zamanda bu $a[0]$ və $a[1]$ ədədlərinin bit-bit AND qiymətinə bərabərdir, istədiyimiz nəticəni əldə etdik.

Example 2

$s = 1$, $n = 2$, $k = 1$, $q = 1000$ olsun. Öncəki nümunədə olduğu kimi, 4 müxtəlif giriş veriləni mümkündür var. Hər 4 hal üçün, $\min(a[0], a[1])$ dəyəri $a[0]$ və $a[1]$ ədədlərinin bit-bit AND

qiymətinə, $\max(a[0], a[1])$ dəyəri isə $a[0]$ və $a[1]$ bit-bit OR qiymətinə bərabərdir. Mümkün bir həll yolu nöbəti çağırışları etməkdir:

1. `_move(1,0)`
2. `_right(1,1,1)`
3. `_and(2,0,1)`
4. `_or(3,0,1)`
5. `_left(3,3,1)`
6. `_or(0,2,3)`

Bu instruksiyaları yerinə yetirdikdən sonra, $c[0] = r[0][0]$ dəyəri $\min(a[0], a[1])$ cavabına, $c[1] = r[0][1]$ dəyəri isə $\max(a[0], a[1])$ cavabına bərabər olur, bu da girişə verilənləri azalmayan şəkildə sıralayır.

Məhdudiyyətlər

- $m = 100$
- $b = 2000$
- $0 \le s \le 1$
- $2 \le n \le 100$
- $1 \le k \le 10$
- $q \le 4000$
- $0 \le a[i] \le 2^k - 1$ ($0 \le i \le n - 1$)

Alt tapşırıqlar

1. (10 xal) $s = 0, n = 2, k \le 2, q = 1000$
2. (11 xal) $s = 0, n = 2, k \le 2, q = 20$
3. (12 xal) $s = 0, q = 4000$
4. (25 xal) $s = 0, q = 150$
5. (13 xal) $s = 1, n \le 10, q = 4000$
6. (29 xal) $s = 1, q = 4000$

Nümunə grader (qiymətləndirici)

Nümunə grader inputu aşağıdakı formatda oxuyur:

- sətir 1 : $s\ n\ k\ q$

Daha sonra t sətir gəlir, hər biri müxtəlif testi izah edir. Hər bir test verilmiş formadadır:

- $a[0]\ a[1]\ \ldots\ a[n-1]$

Bu testdə n ədəd verilmişdir: $a[0], a[1], \ldots, a[n-1]$. Bütün testlər bitdikdən sonra, içində yalnızca -1 olan bir sətir gəlir.

Grader əvvəlcə `construct_instructions(s, n, k, q)` prosedurunu çağırır. Əgər hansısa şərtə əməl olunmazsa, onda yuxarıda qeyd olunmuş mesajlardan biri gəlir və dayanır. Əks halda, grader `construct_instructions(s, n, k, q)` tərəfindən yaradılmış instruksiya ardıcıllığını çıxışa verir. $store$ prosedurları üçün, v çıxışa verilir (0'dan $b-1$'ə qədər).

Sonra grader digər testləri də yoxlayır. Hər test üçün, yaradılmış proqramı giriş verilənləri ilə test edir.

Hər $print(t)$ proseduru üçün, elə $d[0], d[1], \ldots, d[n-1]$ ardıcıllığını nəzərə alın ki, hər i ($0 \leq i \leq n-1$) üçün, $d[i]$ dəyəri t registerinin $i \cdot k$ to $(i+1) \cdot k - 1$ bit ardıcıllığının tam qiymətinə bərabər olsun. Grader, verilmiş formatda bu ardıcıllığı çıxışa verir: `register` t: $d[0]\ d[1]\ \ldots\ d[n-1]$.

Bütün instruksiyalar bitdikdən sonra, grader proqramın cavabını çıxışa verir.

Əgər $s = 0$ olarsa, onda çıxışa verilmə formatı bu formada olur:

- $c[0]$.

Əgər $s = 1$ olarsa, onda çıxışa vermə formatı bu formada olur:

- $c[0]\ c[1]\ \ldots\ c[n-1]$.

Bütün testləri bitirdikdən sonra, grader `number of instructions:` X çıxışa verir, burada X sənin proqramında istifadə olunan instruksiyaların sayıdır.