Бит Шилжүүлэлтийн Регистр

Кристофер инженер хүн ба компьютерийн шинэ процессор дээр ажиллаж байгаа.

Уг процессор нь m тооны b битийн санах ойн үүр рүү (энд $m = 100$ ба $b = 2000$) хандаж чадах ба тэдгээрийг **регистрүүд** гэж нэрлэн 0 - ээс $m - 1$ хүртлэх тоонуудаар дугаарлана. Бид регистрүүдийг $r[0], r[1], \ldots, r[m-1]$ гэж тэмдэглэнэ. Регистр бүр нь 0-ээс (хамгийн баруун талын бит) $b - 1$ (хамгийн зүүн талын бит) хүртлэх тоонуудаар дугаарлагдсан b битийн массив юм. i $(0 \leq i \leq m - 1)$ ба j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд бид i-р регистрийн j-р битийг $r[i][j]$ гэж тэмдэглэнэ.

Ямар нэг $d_0, d_1, \ldots, d_{l-1}$ (дурын l урттай) битүүдийн дарааллын хувьд **бүхэл утга** нь $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$ байна. Бид **регистрт хадгалсан бүхэл тоон утга** i гэж түүний битүүдийн дарааллын бүхэл утга буюу өөрөөр хэлбэл $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$ утгыг хэлнэ.

Уг процессор нь регистрүүдийнхээ битийг өөрчлөх 9 төрлийн **командтай**. Команд бүр нэг эсвэл хэд хэдэн регистр дээр ажиллан, үр дүнгээ регистрүүдийн нэг рүү бичдэг. x-ийн утгыг өөрчлөн y -ийнхтэй адилхан болгох үйлдлийг бид $x := y$ гэж тэмдэглэх болно. Команд бүрийн хийх үйлдлийг доор үзүүлэв.

- $move(t, y)$: y регистр дэх битүүдийн массивыг t регистр рүү хуулах. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд $r[t][j] := r[y][j]$ утга олголтыг хийх.
- $store(t, v)$: t регистрт v-ийн утгыг олгох. Энд v нь b битийн массив байна. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд $r[t][j] := v[j]$ утга олголтыг хийнэ.
- $and(t, x, y)$: x ба y регистрүүд дээр битийн-AND үйлдлийг хийж, үр дүнг нь t регистрт хадгалах. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд, хэрэв $r[x][j]$ ба $r[y][j]$ нь **хоёулаа** 1 бол $r[t][j] := 1$ утга олголтыг, эсрэг тохиолдолд $r[t][j] := 0$ утга олголтыг хийнэ.
- $or(t, x, y)$: x ба y регистрүүд дээр битийн-OR үйлдлийг хийж, үр дүнг нь t регистрт хадгалах. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд, хэрэв $r[x][j]$ ба $r[y][j]$ регистрүүдийн **дор хаяж нэг нь** 1 байвал $r[t][j] := 1$ утга олголтыг, эсрэг тохиолдолд $r[t][j] := 0$ утга олголтыг хийнэ.
- $xor(t, x, y)$: x ба y регистрүүд дээр битийн-XOR үйлдлийг хийж, үр дүнг нь t регистрт хадгалах. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд, хэрэв $r[x][j]$ ба $r[y][j]$ регистрүүдийн **яг нэг нь** 1 байвал $r[t][j] := 1$ утга олголтыг, эсрэг тохиолдолд $r[t][j] := 0$ утга олголтыг хийнэ.
- $not(t, x)$: x регистр дээр битийн-NOT үйлдлийг хийж үр дүнг нь t регистрт хадгалах. j $(0 \leq j \leq b - 1)$ утга бүрийн хувьд $r[t][j] := 1 - r[x][j]$ утга олголтыг хийнэ.

- $left(t, x, p)$: x регистрийн бүх битийг зүүн тийш p битээр шилжүүлж, үр дүнг t регистрт хадгалах. x регистрийн битүүдийг зүүн тийш p битээр шилжүүлсний үр дүн нь b битийн v массив байна. j $(0 \leq j \leq b-1)$ утга бүрийн хувьд хэрэв $j \geq p$ бол $v[j] = r[x][j-p]$ эсрэг тохиолдолд $v[j] = 0$ байна. j $(0 \leq j \leq b-1)$ утга бүрийн хувьд $r[t][j] := v[j]$ утга олголтыг хийх.

- $right(t, x, p)$: x регистрийн бүх битийг баруун тийш p битээр шилжүүлж, үр дүнг t регистрт хадгалах. x регистрийн битүүдийг баруун тийш p битээр шилжүүлсний үр дүн нь b битийн v массив байна. j $(0 \leq j \leq b-1)$ утга бүрийн хувьд хэрэв $j \leq b-1-p$ бол $v[j] = r[x][j+p]$ эсрэг тохиолдолд $v[j] = 0$ байна. j $(0 \leq j \leq b-1)$ утга бүрийн хувьд $r[t][j] := v[j]$ утга олголтыг хийх.

- $add(t, x, y)$: x болон y регистрт байгаа бүхэл утгуудыг нэмж, үр дүнг t регистрт хадгалах. Нэмэх үйлдлийг 2^b модулиар хийнэ. X нь x регистрт хадгалагдсан бүхэл тоон утга, Y нь y регистрт хадгалагдсан бүхэл тоон утга байг. T нь үйлдлийг гүйцэтгэсний дараа t регистрт хадгалагдах утга байг. Хэрэв $X + Y < 2^b$ бол t-гийн битүүдэд $T = X + Y$ байхаар утга олгоно. Эсрэг тохиолдолд t-ийн битүүдийг $T = X + Y - 2^b$ байхаар утга олгоно.

Кристофер таныг шинэ процессорыг ашиглан хоёр төрлийн бодлогыг бодохыг хүсч байгаа. Бодлогын төрлийг s бүхэл тоогоор илэрхийлнэ. Хоёр төрлийн бодлогын аль алинд нь та дээрх командуудаас бүрдсэн дарааллаар илэрхийлэгдэх **програмыг** зохионо.

Програмын **оролт** тус бүр нь k битийнх байх $a[0], a[1], \ldots, a[n-1]$ гэсэн n ширхэг тооноос тогтоно. Өөрөөр хэлбэл $a[i] < 2^k$ ($0 \leq i \leq n-1$) байна. Програмыг биелүүлэхийн өмнө оролтын тоонуудыг дараалсан байдлаар 0-р регистрт хадгалсан байна. Энэ үед i $(0 \leq i \leq n-1)$ утга бүрийн хувьд $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ гэсэн дараалсан k тооны битээр үүсэх бүхэл тоо нь $a[i]$-тэй тэнцүү байна. Мөн $n \cdot k \leq b$ байна. 0-р регистр дэх бусад бүх битүүд (өөрөөр хэлбэл $n \cdot k$ - гаас $b-1$ хүртлэх индекстэй битүүд) болон бусад бүх регистрийн бүх битийн анхны утга нь 0 байна.

Програмыг ажиллуулна гэдэг нь командуудыг дарааллаар нь биелүүлэхийг хэлнэ. Сүүлийн командыг биелүүлсний дараа 0-р регистрийн битүүдийн эцсийн утга дээр үндэслэн програмын **гаралтыг** тооцоолно. Тухайлбал, гаралт нь $c[0], c[1], \ldots, c[n-1]$ гэсэн n тооны бүхэл тоонууд ба энд i ($0 \leq i \leq n-1$) утга бүрийн хувьд $c[i]$ нь 0-р регистрийн $i \cdot k$ - р битээс $(i+1) \cdot k - 1$-р бит хүртлэх дарааллаар үүсэх бүхэл тоон утга байна. Програм ажиллаж дууссаны дараа 0-р регистрийн үлдсэн битүүд (дор хаяж $n \cdot k$ индекстэй) болон бусад бүх регистрийн бүх битүүд дурын утгатай байж болно.

- Эхний $(s = 0)$ бодлого нь оролтын $a[0], a[1], \ldots, a[n-1]$ бүхэл тоонууд дотроос хамгийн багыг нь олох тухай юм. Тодруулбал, $c[0]$ нь $a[0], a[1], \ldots, a[n-1]$ тоонуудын хамгийн бага нь байна. $c[1], c[2], \ldots, c[n-1]$ тоонууд нь дурын утгатай байж болно.

- Хоёр дахь $(s = 1)$ бодлого нь оролтын $a[0], a[1], \ldots, a[n-1]$ бүхэл тоонуудыг үл өсөхөөр эрэмбэлэх тухай юм. Тухайлбал, бүх i ($0 \leq i \leq n-1$) утгуудын хувьд $c[i]$ нь $a[0], a[1], \ldots, a[n-1]$ тоонуудын дотор хамгийн багаараа $1+i$-рт орох тоо байна (өөрөөр хэлбэл $c[0]$ нь оролтын бүхэл тоонуудын хамгийн бага нь байна).

Эдгээр бодлогуудыг бодох, дээд тал нь q командаас тогтох програмуудыг Кристоферт зохиож өгнө үү.

Хэрэгжүүлэлтийн мэдээлэл

Та дараах функцийг хэрэгжүүлнэ:

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

- s: бодлогын төрөл.
- n: оролтын бүхэл тоонуудын тоо.
- k: оролтын бүхэл тоо бүрийн битийн тоо.
- q: командуудын зөвшөөрөгдөх хамгийн их тоо.
- Уг функц нь яг нэг удаа дуудагдах ба шаардлагатай даалгаврыг гүйцэтгэх командуудын дарааллыг зохионо.

Командуудын дарааллыг зохиохын тулд уг функц нь доорх функцуудээс нэг эсвэл хэд хэдийг нь дуудна:

```
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)
```

- Функц бүр харгалзан $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ эсвэл $add(t, x, y)$ командыг програм руу нэмнэ.
- Бүх зөв командуудын хувьд t, x, y утгууд нь дор хаяж 0, дээд тал нь $m - 1$ байна.
- Бүх зөв командуудын хувьд t, x, y утгууд хос хосоороо ялгаатай байх албагүй.
- $left$ ба $right$ командуудын хувьд p нь дор хаяж 0, дээд тал нь b байна.
- $store$ командын хувьд v-ийн урт нь b байна.

Та өөрийн бодолтоо тестлэхийн тулд дараах функцийг дуудаж болно:

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

- Таны бодолтыг шалгах үед уг функцийн дуудалтыг хэрэгсэхгүй.
- Жишээ шалгагч дээр энэ функц програм руу $print(t)$ үйлдлийг нэмнэ.
- Жишээ шалгагч програмыг биелүүлэх үедээ $print(t)$ үйлдэлтэй дайралдвал t регистрийн эхний $n \cdot k$ битүүдээр үүсгэгдэх n тооны k битийн бүхэл тоонуудыг хэвлэнэ ("Жишээ шалгагч" хэсгээс дэлгэрэнгүйг үз).

- t нь $0 \leq t \leq m-1$ нөхцлийг хангана.
- Уг функцийн аль ч дуудалт нь зохиож буй командуудын тоог нэмэгдүүлэхгүй.

Сүүлийн командыг нэмснийхээ дараа `construct_instructions` функц төгснө. Үүний дараа програмыг тодорхой тооны тестээр шалгах ба тест бүр нь $a[0], a[1], \ldots, a[n-1]$ гэсэн n тооны k битийн бүхэл тооноос тогтох оролттой байна. Таны програмын гаралт $c[0], c[1], \ldots, c[n-1]$ нь дараах нөхцлүүдийг хангаж байвал таны бодолт уг тестийг давна:

- Хэрэв $s = 0$ бол $c[0]$ нь $a[0], a[1], \ldots, a[n-1]$ тоонуудын хамгийн бага нь байна.
- Хэрэв $s = 1$ бол i ($0 \leq i \leq n-1$) бүрийн хувьд $c[i]$ нь $a[0], a[1], \ldots, a[n-1]$ тоонуудын дотор хамгийн багаараа $1+i$-рт орох тоо байна.

Таны бодолтыг шалгах үед доорх алдааны мэдээллүүд гарч болно:

- `Invalid index`: функцуудын аль нэгийг дуудахдаа t, x эсвэл y параметр дээр регистрийн буруу (магадгүй сөрөг) индексийг зааж өгсөн.
- `Value to store is not b bits long`: `append_store` - д өгсөн v-ийн урт нь b-тэй тэнцүү биш байх.
- `Invalid shift value`: `append_left` эсвэл `append_right`-д өгсөн p утга нь 0 - ээс b хүртлэх завсарт байхгүй байх.
- `Too many instructions`: таны функц q - ээс олон команд нэмэхийг оролдсон.

Жишээ

Жишээ 1

$s = 0,\ n = 2,\ k = 1,\ q = 1000$ гэж үзье. Оролтон дээр $a[0]$ ба $a[1]$ гэсэн хоёр бүхэл тоо байгаа ба хоёулаа $k = 1$ битийн тоо байна. Програмыг биелүүлэхийн өмнө $r[0][0] = a[0]$ ба $r[0][1] = a[1]$ байна. Процессор дахь бусад бүх битүүд 0 утгатай байна. Програм дахь бүх командыг гүйцэтгэсний дараа $c[0] = r[0][0] = \min(a[0], a[1])$ буюу $a[0]$ ба $a[1]$-ийн хамгийн бага утгатай болсон байх ёстой.

Програмын оролт нь зөвхөн дөрвөн боломжтой:

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

Бүх дөрвөн тохиолдлын хувьд $\min(a[0], a[1])$ нь $a[0]$ ба $a[1]$ дээр битийн-AND үйлдэл хийсний үр дүнтэй тэнцүү байна гэдгийг харж болно. Иймд програмыг зохион нэг боломжит шийдэл нь доорх дуудалтууд байж болно:

1. `append_move(1, 0)`, $r[0]$-ээс $r[1]$ рүү хуулах командыг нэмнэ.
2. `append_right(1, 1, 1)`, $r[1]$ дэх бүх битийг авч, тэдгээрийг баруун тийш 1 битээр шилжүүлээд буцааж $r[1]$-т хадгалах командыг нэмнэ. Бүхэл тоо бүр 1 битийн урттай тул ингэснээр $r[1][0]$ нь $a[1]$-тэй тэнцүү болно.

3. `append_and(0, 0, 1)`, $r[0]$ ба $r[1]$ регистрүүд дээр битийн-AND үйлдлийг хийж, үр дүнг нь $r[0]$ рүү хадгалах командыг нэмнэ. Энэ командыг биелүүлсний дараа $r[0][0]$ - ийн утга нь $r[0][0]$ ба $r[1][0]$ утгууд дээр битийн-AND үйлдэл хийсний үр дүнтэй тэнцүү болох бөгөөд энэ нь бидний хүссэн $a[0]$ ба $a[1]$ утгууд дээр хийсэн битийн-AND үйлдлийн үр дүнтэй адил болно.

Жишээ 2

$s = 1,\ n = 2,\ k = 1,\ q = 1000$ гэж үзье. Өмнөх жишээний адилаар програмд зөвхөн дөрвөн боломжит оролт байна. Бүх дөрвөн тохиолдлын хувьд $\min(a[0], a[1])$ нь $a[0]$ ба $a[1]$ дээр битийн-AND хийсэнтэй, $\max(a[0], a[1])$ нь $a[0]$ ба $a[1]$ дээр битийн-OR хийсэнтэй тэнцүү байна. Нэг боломжит шийдэл нь дараах дуудалтуудыг хийх явдал байна:

1. `append_move(1,0)`
2. `append_right(1,1,1)`
3. `append_and(2,0,1)`
4. `append_or(3,0,1)`
5. `append_left(3,3,1)`
6. `append_or(0,2,3)`

Эдгээр командуудыг биелүүлсний дараа $c[0] = r[0][0]$ нь $\min(a[0], a[1])$ утгыг, $c[1] = r[0][1]$ нь $\max(a[0], a[1])$ утгыг агуулж байх ба ийм байдлаар оролтыг эрэмбэлнэ.

Хязгаарлалт

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

Дэд бодлого

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

Жишээ шалгагч

Жишээ шалгагч нь оролтыг дараах хэлбэрээр уншина:

- мөр 1 : s n k q

Үүний дараа тус бүр нь нэг тестийг илэрхийлэх тодорхой тооны мөр байрлана. Тест бүр доорх хэлбэртэй байна:

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

Энэ нь $a[0], a[1], ..., a[n-1]$ гэсэн n бүхэл тооноос тогтох оролттой тестийг илэрхийлнэ. Тестүүдийн төгсгөлд зөвхөн -1 тоог агуулах мөр байна.

Жишээ шалгагч нь эхлээд `construct_instructions(s, n, k, q)`-г дуудна. Хэрэв уг дуудалт нь бодлогын өгүүлбэрт дурдсан зарим хязгаарлалтуудыг зөрчвөл жишээ шалгагч нь "Хэрэгжүүлэлтийн мэдээлэл" хэсгийн төгсгөлд байгаа алдааны мэдээллүүдийн жагсаалтаас аль нэг алдааг нь хэвлээд гарна. Эсрэг тохиолдолд жишээ шалгагч нь эхлээд `construct_instructions(s, n, k, q)`-ийн нэмсэн команд бүрийг дарааллаар нь хэвлэнэ. $store$ командын хувьд, v-г 0-р индексээс $b-1$-р индекс хүртэл хэвлэнэ.

Дараа нь шалгагч тестүүдийг дарааллаар нь боловсруулна. Тест бүрийн хувьд шалгагч нь зохиосон програмыг уг тестийн оролт дээр ажиллуулна.

$print(t)$ үйлдэл бүрийн хувьд $d[0], d[1], \ldots, d[n-1]$ нь i ($0 \leq i \leq n-1$) утга бүрийн хувьд $d[i]$ нь t регистрийн (үйлдлийг гүйцэтгэх үед) $i \cdot k$ - аас $(i+1) \cdot k - 1$ хүртлэх дугаартай битүүдийн дарааллаар үүсэх бүхэл тоо байх бүхэл тоон дараалал байг. Шалгагч нь уг дарааллыг дараах хэлбэрээр хэвлэнэ: `register` t: $d[0]$ $d[1]$ $\ldots$ $d[n-1]$.

Бүх командыг биелүүлсний дараа жишээ шалгагч нь програмын гаралтыг хэвлэнэ.

Хэрэв $s = 0$ бол тест бүрийн хувьд жишээ шалгагчийн гаралт нь доорх хэлбэртэй байна:

- $c[0]$.

Хэрэв $s = 1$ бол тест бүрийн хувьд жишээ шалгагчийн гаралт нь доорх хэлбэртэй байна:

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

Бүх тестийг биелүүлсний дараа шалгагч `командын тоог` хэвлэнэ: X. Энд X нь таны програмд байгаа командын тоо юм.