Բիթային տեղաշարժերի ռեգիստրներ

Ինժեներ Քրիստոֆերն աշխատում է համակարգչային նոր տիպի պրոցեսորի վրա։

Պրոցեսորին հասանելի են հիշողության m տարբեր b-բիթանոց բջիջներ (որտեղ $m = 100$ և $b = 2000$), որոնք կոչվում են **ռեգիստրներ**, և համարակալված են 0-ից $m - 1$ թվերով։ Ռեգիստրները նշանակենք $r[0], r[1], \ldots, r[m - 1]$-ով։ Յուրաքանչյուր ռեգիստր b բիթերի զանգված է, որոնք համարակալված են սկսած 0-ից (ամենաաջ բիթը) մինչև $b - 1$ (ամենաձախ բիթը)։ Յուրաքանչյուր i $(0 \leq i \leq m - 1)$ և յուրաքանչյուր j $(0 \leq j \leq b - 1)$ համար i-րդ ռեգիստրի j-րդ բիթը նշանակենք $r[i][j]$-ով։

Բիթերի ցանկացած $d_0, d_1, \ldots, d_{l-1}$ (կամայական l երկարության) հաջորդականության համար, հաջորդականության **ամբողջ արժեքը** հավասար է $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$։ Կասենք, որ i **ռեգիստրում պահվող ամբողջ արժեքը** նրա բիթերի հաջորդականության ամբողջ արժեքն է, այսինքն, $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b - 1]$։

Պրոցեսորն ունի 9 տիպի **հրամաններ**, որոնք կարող են օգտագործվել ռեգիստրների բիթերը փոխելու համար։ Յուրաքանչյուր հրաման վերաբերում է մեկ կամ ավել ռեգիստրների և գործողության արդյունքը պահում է մեկ ռեգիստրում։ Ստորև մենք կօգտագործենք $x := y$ տեսքի գրառումներ, որոնք նշանակում են, որ x-ի արժեքը պետք է դառնա y-ին հավասար։ Հրամանների միջոցով կատարվող գործողությունների ցուցակը բերված է ստորև։

- $move(t, y)$: y ռեգիստրի բիթերի զանգվածը պատճենում է t ռեգիստրի բիթերի զանգվածի վրա։ Յուրաքանչյուր j $(0 \leq j \leq b - 1)$ համար, $r[t][j] := r[y][j]$ վերագրում է կատարվում։

- $store(t, v)$: t ռեգիստրի պարունակությունը դառնում է հավասար v-ի, որտեղ v-ն b բիթերի զանգված է։ Յուրաքանչյուր j $(0 \leq j \leq b - 1)$ համար, $r[t][j] := v[j]$ վերագրում է կատարվում։

- $and(t, x, y)$: Կատարում է բիթային AND գործողությունը x և y ռեգիստրների վրա և արդյունքը պահում է t ռեգիստրում։ Յուրաքանչյուր j $(0 \leq j \leq b - 1)$ համար, $r[t][j] := 1$, եթե $r[x][j]$-ն և $r[y][j]$-ն **երկուսն էլ** 1 են, և $r[t][j] := 0$ հակառակ դեպքում։

- $or(t, x, y)$: Կատարում է բիթային OR գործողությունը x և y ռեգիստրների վրա և արդյունքը պահում է t ռեգսիտրում։ Յուրաքանչյուր j $(0 \leq j \leq b - 1)$ համար, $r[t][j] := 1$, եթե $r[x][j]$-ից և $r[y][j]$-ից **առնվազն մեկը** 1 է, և $r[t][j] := 0$ հակառակ դեպքում։

- $xor(t, x, y)$: Կատարում է բիթային XOR գործողությունը x և y ռեգիստրների վրա և արդյունքը պահում է t ռեգսիտրում: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $r[t][j] := 1$, եթե $r[x][j]$-ից և $r[y][j]$-ից **ճիշտ մեկը** 1 է, և $r[t][j] := 0$ հակառակ դեպքում:

- $not(t, x)$: Կատարում է բիթային NOT գործողությունը x ռեգիստրի վրա, և արդյունքը պահում է t ռեգիստրում: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $r[t][j] := 1 - r[x][j]$ վերագրում է կատարվում:

- $left(t, x, p)$: x ռեգիստրի բոլոր բիթերը տեղաշարժվում են դեպի ձախ p դիրքով, և արդյունքը պահվում է t ռեգիստրում: x ռեգիստրի բիթերի p դիրքով դեպի ձախ տեղաշարժը b բիթերից կազմված v զանգված է: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $v[j] = r[x][j-p]$, եթե $j \geq p$, և $v[j] = 0$ հակառակ դեպքում: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $r[t][j] := v[j]$:

- $right(t, x, p)$: x ռեգիստրի բոլոր բիթերը տեղաշարժվում են դեպի աջ p դիրքով, և արդյունքը պահվում է t ռեգիստրում: x ռեգիստրի բիթերի p դիրքով դեպի աջ տեղաշարժը b բիթերից կազմված v զանգված է: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $v[j] = r[x][j+p]$, եթե $j \leq b-1-p$, և $v[j] = 0$ հակառակ դեպքում: Յուրաքանչյուր j $(0 \leq j \leq b-1)$ համար, $r[t][j] := v[j]$:

- $add(t, x, y)$: Գումարում է x և y ռեգիստրներում պահվող ամբողջ արժեքները և արդյունքը պահում է t ռեգիստրում: Գումարն արվում է ըստ մոդուլ 2^b-ի: Ֆորմալ, դիցուկ x ռեգիստրում պահվող արժեքը X է, իսկ y ռեգիստրում՝ Y, նախքան գործողությունը կատարելը: T-ն թող լինի գործողությունը կատարելուց հետո t ռեգիստրում պահվող արժեքը:Եթե $X + Y < 2^b$, t-ի բիթերին այնպիսի արժեքներ են տրվում, որ $T = X + Y$: Հակառակ դեպքում t-ի բիթերին այնպիսի արժեքնե են տրվում, որ $T = X + Y - 2^b$:

Քրիստոֆերը ցանկանում է, որ դուք նոր պրոցեսորի միջոցով երկու տիպի խնդիր լուծեք: Խնդրի տիպը նշանակենք s ամբողջ թվով: Երկու տիպի խնդիրներում էլ դուք պետք է ստեղծեք **ծրագիր**, որը բաղկացած է վերը նկարագրված հրամաններից:

Ծրագրի **մուտքային տվյալները** բաղկացած են n ամբողջ $a[0], a[1], \ldots, a[n-1]$ թվերից, որոնցից յուրաքանչյուրն ունի k բիթ, այսինքն, $a[i] < 2^k$ ($0 \leq i \leq n-1$): Նախքան ծրագրի կատարումը բոլոր մուտքային թվերը հաջորդաբար պահվում են 0 ռեգիստրում, այնպես որ յուրաքանչյուր i $(0 \leq i \leq n-1)$ համար $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ k բիթերի հաջորդականության ամբողջ արժեքը հավասար է $a[i]$: Նկատենք, որ $n \cdot k \leq b$: 0 ռեգիստրի մնացած բոլոր բիթերը (այսինքն նրանք, որոնց ինդեքսներն ընկած են $n \cdot k$-ի և $b-1$-իի միջև, ներառյալ), ինչպես նաև մյուս ռեգիստրների բոլոր բիթերը սկզբնարժեքավորված են 0-ներով:

Ծրագիրն աշխատացնելիս հրամանները հերթով կատարվում են: Վերջին հրամանի կատարումից հետո ծրագրի **ելքային տվյալները** հաշվվում են 0 ռեգիստրի բիթերի վերջնական արժեքների հիման վրա: Մասնավորապես, ելքը n ամբողջ թվերի $c[0], c[1], \ldots, c[n-1]$ հաջորդականություն է, որտեղ յուրաքանչյուր i ($0 \leq i \leq n-1$) համար $c[i]$-ն 0 ռեգիստրի $i \cdot k$-ից մինչև $(i+1) \cdot k - 1$ բիթերից կազմված հաջորդականության ամբողջ արժեքն է: Նկատենք, որ ծրագիրը կատարելուց հետո 0

ռեգիստրի մնացած բիթերը (որոնց ինդեքսներն առնվազն $n \cdot k$ են) և մնացած ռեգիստրների բիթերը կարող են կամայական արժեք ունենալ։

- Առաջին խնդիրն է $(s = 0)$՝ գտնել մուտքային $a[0], a[1], \ldots, a[n-1]$ ամբողջ թվերից փոքրագույնի արժեքը։ Մասնավորապես, $c[0]$-ն պետք է լինի $a[0], a[1], \ldots, a[n-1]$-ից մինիմումը։ $c[1], c[2], \ldots, c[n-1]$-ը կարող են ունենալ կամայական արժեքներ։
- Երկրորդ խնդիրն է $(s = 1)$՝ տեսակավորել մուտքային $a[0], a[1], \ldots, a[n-1]$ ամբողջ թվերը չնվազման կարգով։ Մասնավորապես, յուրաքանչյուր i ($0 \leq i \leq n-1$) համար, $c[i]$-ն պետք է հավասար լինի $a[0], a[1], \ldots, a[n-1]$ թվերից մեծությամբ $i+1$-րդին (այսինքն, $c[0]$-ն պետք է հավասար լինի մուտքային թվերից փոքրագույնին)։

Տրամադրեք Քրիստոֆերին այս խնդիրները լուծող ծրագրեր, որոնք բաղկացած լինեն առավելագույնը q հրամաններից։

Իրականացման մանրամասներ

Դուք պետք է իրականացնեք հետևյալ ֆունկցիան.

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

- s: խնդրի տիպը։
- n: մուտքում ամբողջ թվերի քանակը։
- k: յուրաքանչյուր մուտքային ամբողջում բիթերի քանակը։
- q: հրամանների թույլատրվող մաքսիմում քանակը։
- Այս ենթածրագիրը կանչվում է ճիշտ մեկ անգամ և պետք կառուցի առաջադրվող խնդիրը լուծող հրամանների հաջորդականություն։

Հրամանների հաջորդականությունը կառուցելու համար ֆունկցիան կարող է կատարել հետևյալ ֆունկցիաների մեկ կամ ավել կանչեր։

```
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)
```

- Յուրաքանչյուր ֆունկցիա ծրագրի վերջից ավելացնում է համապատասխանաբար $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ և $add(t, x, y)$ հրամանը։

- Բոլոր համապատասխան հրահանգների համար, t-ն, x-ը, y-ը պետք է լինեն առնվազն 0 և առավելագույնը $m-1$:
- Բոլոր համապատասխան հրահանգների համար, t-ն, x-ը, y-ը պատադիր չէ, որ զույգ առ զույգ տարբեր լինեն:
- Բոլոր $left$ և $right$ համանների համար, p-ն պետք է լինի առնվազն 0 և առավելագույնը b:
- Բոլոր $store$ հրամանների համար, v-ի երկարությունը պետք է առավելագույնը b լինի:

Դուք նաև կարող եք կանչել հետևյալ ֆունկցիան ձեր լուծումը թեստավորելու նպատակով.

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

- Այս ֆունկցիայի կանչերը կանտեսվեն ձեր լուծումը ստուգելու ժամանակ:
- Գրեյդերի նմուշում այս ֆունկցիան ծրագրի վերջում ավելացնում է $print(t)$ հրամանը:
- Գրեյդերի նմուշը $print(t)$ գործողությունը կատարելու ժամանակ n հատ k-բիթանոց ամբողջ թվեր է տպում, որոնք կազմվում են t ռեգիստրի առաջին $n \cdot k$ բիթերից (մանրամասների համար տե՛ս «Գրեյդերի նմուշ» բաժինը):
- t-ն պետք է բավարարի $0 \leq t \leq m-1$ պայմանին:
- Այս ֆունկցիայի կանչերը կառուցված հրամանների քանակը չեն ավելացնում:

Վերջին հրամանը կատարելուց հետո `construct_instructions`-ը պետք է ավարտվի: Ապա ծրագիրը ստուգվում է որոշակի քանակությամբ թեստերի վրա, որոնցից յուրաքանչյուրը նկարագրում է մուտքային տվյալներ՝ կազմված n հատ k-բիթանոց $a[0], a[1], \ldots, a[n-1]$ ամբողջ թվերից: Ձեր ծրագիրն անցնում է տրված թեստը, եթե ծրագրի $c[0], c[1], \ldots, c[n-1]$ ելքային տվյալները բավարարում են հետևյալ պայմաններին.

- եթե $s = 0$, $c[0]$-ն պետք է հավասար լինի $a[0], a[1], \ldots, a[n-1]$-ից փոքրագույնին:
- Եթե $s = 1$, յուրաքանչյուր i ($0 \leq i \leq n-1$) համար $c[i]$-ն պետք է հավասար լինի $a[0], a[1], \ldots, a[n-1]$-ից մեծությամբ $i+1$-րդին:

Ձեր լուկծման ստուգման արդյունքում կարող է ստացվել սխալների վերաբերյալ հետևյալ հաղորդագրություններից մեկը.

- `Invalid index`: ֆունկցիաներից որևէ մեկի կանչի ժամանակ t, x կամ y պարամետրերից որևէ մեկին տրվել է ոչ ճիշտ (հնարավոր է բացասական) ինդեքս:
- `Value to store is not b bits long`: `append_store`-ին տրված v-ի երկարությունը հավասար չէ b:
- `Invalid shift value`: p-ի արժեքը, որ տրվել է `append_left`-ին կամ `append_right`-ին ընկած չէ 0-ի և b-ի միջև, ներառյալ:
- `Too many instructions`: ձեր ֆունկցիան փորձում է q-ից ավել քանակով հրաման կատարել:

Օրինակներ

Օրինակ 1

Ենթադրենք $s = 0,\ n = 2,\ k = 1,\ q = 1000$: Կան երկու մուտքային ամբողջ թվեր՝ $a[0]$ և $a[1]$, յուրաքանչյուրն ունի $k = 1$ բիթ: Նախքան ծրագրի կատարվելը $r[0][0] = a[0]$ և $r[0][1] = a[1]$: Պրոցեսորում մյուս բոլորի բիթերին 0 է վերագրված: Ծրագրի հրամանները կատարելու արդյունքում պետք է լինի $c[0] = r[0][0] = \min(a[0], a[1])$, որը $a[0]$-ի և $a[1]$-ի մինիմումն է:

Կա մուտքային տվյալների 4 հնարավոր տարբերակ.

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

Կարող ենք նկատել, որ բոլոր 4 դեպքերում, $\min(a[0], a[1])$-ը հավասար է $a[0]$-ի և $a[1]$-ի բիթային AND-ին: Հետևաբար, որպես լուծում կարելի է ծրագիրը կառուցել կատարելով հետևյալ կանչերը.

1. `append_move(1, 0)`, որը վերջից ավելացնում է $r[0]$-ն $r[1]$-ին պատճենելու հրամանը:
2. `append_right(1, 1, 1)`, որն վերջից ավելացնում է հրաման, որը վերցնում է $r[1]$-ի բոլոր բիթերը, տեղաշարժում է դրանք 1 բիթով դեպի աջ, և արդյունքը պահում է կրկին $r[1]$-ում: Քանի որ բոլոր մուտքային թվերն ունեն 1 բիթ երկարություն, արդյունքում $r[1][0]$-ն կլինի հավասար $a[1]$-ին:
3. `append_and(0, 0, 1)`, որը վերջից ավելացնում է հրաման, որի կատարման արդյունքում $r[0]$-ի և $r[1]$-ի բիթային AND-ը պահվում է $r[0]$-ում: Այս հրամանի կատարման արդյունքում $r[0][0]$-ին վերագրվում է $r[0][0]$-ի և $r[1][0]$-ի բիթային AND-ը, որը հավասար է $a[0]$-ի և $a[1]$-ի բիթային AND-ին:

Օրինակ 2

Ենթադրենք $s = 1,\ n = 2,\ k = 1,\ q = 1000$: Առաջին օրինակի նման կա մուտքային տվյալների 4 տարբերակ, $a[0]$-ի և $a[1]$-ի բիթային AND-ի միջոցով կստանանք $\min(a[0], a[1])$, իսկ $a[0]$-ի և $a[1]$-ի բիթային OR-ի միջոցով կստանանք $\max(a[0], a[1])$: Հնարավոր լուծումներից մեկը ստացվում է հետևյալ կանչերի միջոցով.

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)`

Այս հրամանները կատարելուց հետո $c[0] = r[0][0]$ պարունակում է $\min(a[0], a[1])$, և $c[1] = r[0][1]$ պարունակում է $\max(a[0], a[1])$, որը մուտքային տվյալների տեսակավորումն է։

Սահմանափակումներ

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

Ենթախնդիրներ

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

Գրեյդերի նմուշ

Գրեյդերի նմուշը մուտքային տվյալները կարդում է հետևյալ ձևաչափով.

- line 1 : s n k q

Սրան հաջորդում են ինչ-որ քանակությամբ տողեր, որոնցից յուրաքանչյուրը նկարագրում է մեկ թեստ։ Յուրաքանչյուր թեստ ունի այսպիսի ձևաչափ.

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

սա նշանակում է, որ մուտքային տվյալները բաղկացած են n ամբողջ $a[0], a[1], \ldots, a[n-1]$ թվերից։ Բոլոր թեստերի նկարագրություններն ավարտվում են մեկ տողով, որը պարունակում է միայն -1 թիվը։

Գրեյդերի նմուշը սկզբում կանչում է `construct_instructions(s, n, k, q)`-ը։ Եթե կանչը հակասում է խնդրում նկարագրված սահմանափակումներից որևէ մեկին, գրեյդերի նմուշը տպում է «Իրականացման մանրամասներ» բաժնում նշված սխալների մասին հաղորդագրություններից որևէ մեկը։ Հակառակ դեպքում գրեյդերի նմուշը նախ հերթով տպում է `construct_instructions(s, n, k, q)`-ի տված հրամանները։ $store$ հրամանի համար, v-ն տպվում է 0 ինդեքսից մինչև $b - 1$ ինդեքսը։

Ապա գրեյդերի նմուշը հերթով մշակում է թեստերը։ Յուրաքանչյուր թեստի համար այն կատարում է կառուցված ծրագիրը թեստի մուտքային տվյալների վրա։

Յուրաքանչյուր $print(t)$ գործողության համար, թող $d[0], d[1], \ldots, d[n-1]$ ամբողջ թվերի հաջորդականություն է, այնպիսին որ յուրաքանչյուր i ($0 \leq i \leq n-1$) համար, $d[i]$-ն t ռեգիստրի $i \cdot k$ to $(i+1) \cdot k - 1$ բիթերի հաջորդականության ամբողջ արժեքն է (երբ գործողությունը կատարվում է)։ Գրեյդերը տպում է հաջորդականությունը հետևյալ ձևաչափով. `register` t: $d[0]\ d[1]\ \ldots\ d[n-1]$:

Բոլոր հրամանները կատարելուց հետո գրեյդերի նմուշը տպում է ծրագրի ելքային տվյալները։

Եթե $s = 0$, գրեյդերի նմուշի արտածումը յուրաքանչյուր թեստի համար այսպիսի ձևաչափ ունի.

- $c[0]$.

Եթե $s = 1$, գրեյդերի նմուշի արտածումը յուրաքանչյուր թեստի համար այսպիսի ձևաչափ ունի.

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

Բոլոր թեստերը կատարելուց հետո գրեյդերի նմուշը տպում է `number of instructions:` X որտեղ X-ը ձեր ծրագրի հրամանների քանակն է։