Bitinihkeregistrid

Insener Christopher töötab uut liiki arvutiprotsessori kallal.

Protsessoril on ligipääs m erinevale b-bitisele mälupesale (kus $m = 100$ ja $b = 2000$), mida kutsutakse **registriteks** ja mis on nummerdatud 0 kuni $m - 1$. Tähistame registreid kui $r[0], r[1], \ldots, r[m-1]$. Iga register on massiiv b bitist, nummerdatud 0 (parempoolseim bitt) kuni $b - 1$ (vasakpoolseim bitt). Iga i jaoks $(0 \leq i \leq m - 1)$ ja iga j jaoks $(0 \leq j \leq b - 1)$ tähistame $j$ndat bitti registris i kui $r[i][j]$.

Iga (suvalise pikkusega l) bitijada $d_0, d_1, \ldots, d_{l-1}$ jaoks on jada **täisarvuväärtus** suurus $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$. Ütleme, et **registri täisarvuväärtus** i on selle bitijada täisarvuväärtus, s.t. see on $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

Protsessoril on 9 liiki **käske**, millega saab registrite bitte muuta. Iga käsk tegutseb ühe või rohkema registri peal ja paneb väljundi ühte registrisse. Järgnevalt tähistame operatsiooni, mis muudab x väärtuse võrdseks y omaga, tähisega $x := y$. Iga käsuliigi poolt läbi viidud operatsioonid on kirjeldatud allpool.

- $move(t, y)$: Kopeeri bitimassiiv registrist y registrisse t. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := r[y][j]$.
- $store(t, v)$: Muuda register t võrdseks väärtusega v, kus v on b bitist koosnev massiiv. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := v[j]$.
- $and(t, x, y)$: Vii läbi bitikaupa AND-operatsioon registritel x ja y ning pane tulemus registrisse t. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := 1$ kui **mõlema** $r[x][j]$ ja $r[y][j]$ väärtus on 1, vastasel juhul määra $r[t][j] := 0$.
- $or(t, x, y)$: Vii läbi bitikaupa OR-operatsioon registritel x ja y ning pane tulemus registrisse t. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := 1$ kui **vähemalt üks** bittidest $r[x][j]$ ja $r[y][j]$ on 1, vastasel juhul määra $r[t][j] := 0$.
- $xor(t, x, y)$: Vii läbi bitikaupa XOR-operatsioon registritel x ja y ning pane tulemus registrisse t. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := 1$ kui **täpselt üks** bittidest $r[x][j]$ ja $r[y][j]$ on 1, vastasel juhul määra $r[t][j] := 0$.
- $not(t, x)$: Vii läbi bitikaupa NOT-operatsioon registril x ning pane tulemus registrisse t. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := 1 - r[x][j]$.
- $left(t, x, p)$: Võta registri x bitid, nihuta kõiki vasakule p võrra ja pane tulemus registrisse t. Registri x bittide p võrra vasakule nihutamise tulemus on b bitist koosnev massiiv v. Iga j jaoks $(0 \leq j \leq b - 1)$, $v[j] = r[x][j - p]$ kui $j \geq p$, vastasel juhul $v[j] = 0$. Iga j jaoks $(0 \leq j \leq b - 1)$ määra $r[t][j] := v[j]$.

- $right(t, x, p)$: Võta registri x bitid, nihuta kõiki paremale p võrra ja pane tulemus registrisse t. Registri x bittide p võrra paremale nihutamise tulemus on b bitist koosnev massiiv v. Iga j jaoks $(0 \leq j \leq b-1)$, $v[j] = r[x][j+p]$ kui $j \leq b-1-p$, vastasel juhul $v[j] = 0$. Iga j jaoks $(0 \leq j \leq b-1)$ määra $r[t][j] := v[j]$.

- $add(t, x, y)$: Liida täisarvuväärtused registrites x ja y ning pane tulemus registrisse t. Liitmist viiakse läbi mooduli 2^b järgi. Formaalselt olgu X täisarvuväärtus registris x ja Y täisarvuväärtus registris y enne operatsiooni sooritamist. Olgu T täisarvuväärtus registris t pärast operatsiooni sooritamist. Kui $X + Y < 2^b$, siis määra t bitid selliselt, et $T = X + Y$. Vastasel juhul määra t bitid selliselt, et $T = X + Y - 2^b$.

Christopher tahab, et sa lahendaksid uue protsessoriga kaht tüüpi ülesandeid. Ülesande tüüpi näitab täisarv s. Mõlema ülesandetüübi jaoks pead looma **programmi**, s.t. ülaltoodud käskude jada.

Programmi **sisend** koosneb n täisarvust $a[0], a[1], \ldots, a[n-1]$, millest igaühes on k bitti, s.t. $a[i] < 2^k$ ($0 \leq i \leq n-1$). Enne programmi käivitamist hoiustatakse kõik sisendi arvud järjest registris 0 nii, et iga i jaoks $(0 \leq i \leq n-1)$ k-bitise jada täisarvuväärtus $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ on võrdne väärtusega $a[i]$. Pane tähele, et $n \cdot k \leq b$. Kõik teised bitid registris 0 (s.t. need, mille indeksid on vahemikus $n \cdot k$ and $b-1$, kaasaarvatud) ja kõik teised bitid kõigis teistes registrites lähtestatakse väärtusega 0.

Programmi käivitamine koosneb selle käskudejada õiges järjekorras täideviimisest. Pärast viimase käsu täideviimist on leitakse programmi **väljund** registri 0 bittide viimase väärtuse põhjal. Täpsemalt on väljund n täisarvust koosnev jada $c[0], c[1], \ldots, c[n-1]$, kus iga i jaoks ($0 \leq i \leq n-1$) on $c[i]$ jada täisarvuväärtus, mis koosneb registri 0 bittidest $i \cdot k$ to $(i+1) \cdot k - 1$. Pane tähele, et pärast programmi jooksutamist võivad kõik ülejäänud bitid registris 0 (indeksitega vähemalt $n \cdot k$) ja kõik teised bitid kõigis teistes registrites olla suvalise väärtusega.

- Esimene ülesanne $(s = 0)$ on leida vähim täisarv sisendi täisarvude $a[0], a[1], \ldots, a[n-1]$ hulgas. Täpsemalt peab $c[0]$ olema miinimum hulgast $a[0], a[1], \ldots, a[n-1]$. $c[1], c[2], \ldots, c[n-1]$ väärtused võivad olla suvalised.

- Teine ülesanne $(s = 1)$ on sorteerida sisendarvud $a[0], a[1], \ldots, a[n-1]$ mittekahanevas järjestuses. Täpsemalt peab iga i ($0 \leq i \leq n-1$) puhul $c[i]$ olema võrdne järjekorras $1+i$-nda täisarvuga, kui lugeda arve $a[0], a[1], \ldots, a[n-1]$ vähimast suurimani (s.t. $c[0]$ on sisendis toodud arvudest vähim).

Anna Christopherile ülimalt q käsust koosnevad programmid, mis need ülesanded ära lahendaksid.

Realisatsioon

Lahendusena tuleb realiseerida funktsioon

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

- s: ülesande tüüp.
- n: täisarvude hulk sisendis.

- k: bittide arv igas sisendarvus.
- q: maksimaalne lubatud juhiste arv.
- Seda funktsiooni kutsutakse välja täpselt ühe korra ja see peab koostama antud ülesannet lahendava käskudejada.

See funktsioon peaks käskudejada loomiseks välja kutsuma üht või enamat järgmist funktsiooni:

```
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)
```

- Iga funktsioon lisab programmi vastavalt käsu $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ või $add(t, x, y)$.
- Kõigi asjakohaste käskude jaoks peavad t, x, y olema vähemalt 0 ja ülimalt $m - 1$.
- Kõigi asjakohaste käskude jaoks ei ole t, x, y tingimata paarikaupa erinevad.
- $left$ ja $right$ käskude jaoks peab p olema vähemalt 0 ja ülimalt b.
- $store$ käskude jaoks peab v pikkus olema b.

Võid oma lahenduse testimise hõlbustamiseks kutsuda välja ka järgmist funktsiooni:

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

- Kõiki selle funktsiooni väljakutseid ignoreeritakse lahenduse hindamise ajal.
- Näidishindajas lisab see väljakutse programmi operatsiooni $print(t)$.
- Kui näidishindaja kohtab programmi käivitamise ajal operatsiooni $print(t)$, siis väljastab see n k-bitist täisarvu, mis on moodustatud registri t esimesest $n \cdot k$ bitist (detailide jaoks vt "Näidishindaja" lõiku).
- t peab rahuldama $0 \leq t \leq m - 1$.
- Ükski selle funktsiooni väljakutse ei suurenda konstrueeritud käskude arvu.

Pärast viimase käsu lisamist peaks `construct_instructions` tagastama tühja värtuse. Programmi testitakse siis mingi hulga testjuhtude peal, millest igaüks on n k-bitisest täisarvust koosnev sisend $a[0], a[1], \ldots, a[n-1]$. Sinu lahendus läbib antud testjuhu, kui programmi väljund $c[0], c[1], \ldots, c[n-1]$ antud sisendi jaoks rahuldab järgmiseid tingimusi:

- Kui $s = 0$, siis peab $c[0]$ olema vähim väärtus hulgast $a[0], a[1], \ldots, a[n-1]$.
- Kui $s = 1$, siis peab iga i ($0 \leq i \leq n - 1$) korral $c[i]$ olema järjekorras $1 + i$-s täisarv, kui arvud $a[0], a[1], \ldots, a[n-1]$ järjestada vähimast suurimani.

Sinu lahenduse hindamine võib anda ühe järgmistest veateadetest:

- `Invalid index`: parameetriks t, x või y oli mõnel funktsiooni väljakutsel vale (võimalik, et negatiivne) registriindeks.
- `Value to store is not b bits long`: käsu `append_store` sisendi v pikkus ei ole b.
- `Invalid shift value`: käsu `append_left` või `append_right` sisendväärtus p ei kuulu vahemikku 0 ja b (kaasaarvatud).
- `Too many instructions`: sinu funktsioon proovis lisada rohkem kui q käsku.

Näited

Näide 1

Olgu $s = 0,\ n = 2,\ k = 1,\ q = 1000$. Sisendis on kaks täisarvu $a[0]$ ja $a[1]$, kummaski $k = 1$ bitt. Enne programmi jooksutamist $r[0][0] = a[0]$ ja $r[0][1] = a[1]$. Kõik teised bitid protsessoris lähtestatakse nulliks. Pärast kõiki käskude käivitamist peab meil olema $c[0] = r[0][0] = \min(a[0], a[1])$, mis on hulga $a[0]$ ja $a[1]$ vähim väärtus.

Programmile on ainult 4 võimalikku sisendit:

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

Paneme tähele, et kõigi 4 juhu puhul on $\min(a[0], a[1])$ võrdne bitikaupa AND-operatsiooniga $a[0]$ ja $a[1]$ peal. Seega on üks võimalik lahendus konstrueerida programm järgmiste väljakutsetega:

1. `append_move(1, 0)`, mis lisab käsu kopeerida $r[0]$ registrisse $r[1]$.
2. `append_right(1, 1, 1)`, mis lisab käsu võtta kõik bitid registris $r[1]$, nihutab neid 1 biti võrra paremale ja paneb tulemuse tagasi registrisse $r[1]$. Kuna iga täisarv on 1 biti pikkune, siis on tulemusena $r[1][0]$ väärtus võrdne $a[1]$-ga.
3. `append_and(0, 0, 1)`, mis lisab käsu leida bitikaupa AND registritest $r[0]$ ja $r[1]$ ja panna tulemus registrisse $r[0]$. Pärast selle käsu täitmist on $r[0][0]$ väärtus bitikaupa AND-operatsiooni tulemus registritel $r[0][0]$ ja $r[1][0]$, mis on võrdne bitikaupa AND-operatsiooniga $a[0]$ ja $a[1]$ peal, nagu vaja.

Näide 2

Olgu $s = 1,\ n = 2,\ k = 1,\ q = 1000$. Nagu eelmiseski näites on programmile ainult 4 võimalikku sisendit. Kõigil 4 juhul on $\min(a[0], a[1])$ bitikaupa AND-operatsioon $a[0]$ ja $a[1]$ peal ning $\max(a[0], a[1])$ bitikaupa OR-operatsioon $a[0]$ ja $a[1]$ peal. Võimalik lahendus on teha järgmised väljakutsed:

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)`

Pärast nende käskude täitmist on $c[0] = r[0][0]$ väärtus $\min(a[0], a[1])$ ja $c[1] = r[0][1]$ väärtus $\max(a[0], a[1])$, mis sorteerib sisendi ära.

Piirangud

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

Alamülesanded

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

Näidishindaja

Näidishindaja loeb sisendit järgmises vormingus:

- rida 1 : s n k q

Sellele järgneb mingi hulk ridu, millest igaüks kirjeldab üht testjuhtu. Iga testjuht on antud järgmises vormingus:

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

ja kirjeldab testjuhtu, kus sisendiks on n täisarvu $a[0], a[1], \ldots, a[n-1]$. Kõigile testjuhtudele järgneb rida, millel on vaid arv -1.

Näidishindaja kutsub kõigepealt välja `construct_instructions(s, n, k, q)`. Kui see väljakutse rikub mõnd ülesande tingimustes välja toodud piirangut, siis väljastab näidishindaja ühe lõigu "Realisatsioon" lõpus välja toodud veateadetest ja lõpetab töö. Vastasel juhul väljastab näidishindaja kõigepealt iga käsu, mille `construct_instructions(s, n, k, q)` paika pani, vastavas järjekorras. $store$ käskude jaoks prinditakse v välja indeksist 0 indeksini $b - 1$.

Siis läbib näidishindaja järgemööda testjuhud. Iga testjuhu jaoks käivitatakse konstrueeritud programmi testjuhu sisendiga.

Olgu iga $print(t)$ operatsiooni jaoks $d[0], d[1], \ldots, d[n-1]$ selline täisarvude jada, et iga i ($0 \leq i \leq n-1$) puhul on $d[i]$ registri t bitijada $i \cdot k$ to $(i+1) \cdot k - 1$ täisarvuväärtus (hetkel, mil operatsiooni tehakse). Näidishindaja väljastab selle jada järgmises vormingus: `register` t: $d[0]\ d[1]\ \ldots\ d[n-1]$.

Kui kõik käsud on käivitatud, siis väljastab näidishindaja programmi väljundi.

Kui $s = 0$, siis on näidishindaja väljund iga testjuhu jaoks järgmises vormingus:

- $c[0]$.

Kui $s = 1$, siis on näidishindaja väljund iga testjuhu jaoks järgmises vormingus:

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

Pärast kõigi testjuhtude täitmist väljastab näidishindaja `number of instructions:` X, kus X on sinu programmi käskude arv.