# Bitų postūmio registrai

Inžinierius Kristupas dirba su naujos rūšies kompiuterio procesoriumi.

Procesoriui pasiekiamos $m$ atskirų $b$ bitų dydžio atminties lastelių (čia $m = 100$ ir $b = 2000$), vadinamų **registrais** ir sunumeruotų nuo $0$ iki $m - 1$. Šiuos registrus žymime $r[0], r[1], \ldots, r[m-1]$. Kiekvienas registras yra masyvas, sudarytas iš $b$ bitų, sunumeruotų nuo $0$ (dešiniausias bitas) iki $b - 1$ (kairiausias bitas). Kiekvienam $i$ $(0 \leq i \leq m - 1)$ ir kiekvienam $j$ $(0 \leq j \leq b - 1)$, $i$-ojo registro $j$-ąjį bitą žymime $r[i][j]$.

Bitų sekos $d_0, d_1, \ldots, d_{l-1}$ **skaitinė vertė** yra lygi $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$ (čia $l$ – sekos ilgis). Sakome, kad **skaitinė vertė, įrašyta registre** $i$, yra to registro bitų sekos skaitinė vertė, t.y. $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

Procesorius palaiko $9$-ių tipų **instrukcijas**, kuriomis galima keisti registrų bitų reikšmes. Kiekviena instrukcija dirba su vienu ar daugiau registrų bei įrašo reikšmę į vieną iš šių registrų. Toliau naudosime žymėjimą $x := y$, nurodantį operaciją, pakeičiančią $x$ reikšmę taip, kad ji taptų lygi $y$. Instrukcijų atliekamos operacijos aprašytos žemiau:

- $move(t, y)$: Nukopijuoti bitų masyvą, esantį registre $y$, į registrą $t$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := r[y][j]$.
- $store(t, v)$: Registrui $t$ priskirti reikšmę $v$, kur $v$ yra $b$ bitų masyvas. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := v[j]$.
- $and(t, x, y)$: Atlikti bitinę AND operaciją su registrais $x$ ir $y$, o rezultatą įrašyti į registrą $t$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := 1$ jei **abiejų** $r[x][j]$ ir $r[y][j]$ reikšmės yra $1$, priešingu atveju priskirti $r[t][j] := 0$.
- $or(t, x, y)$: Atlikti bitinę OR operaciją su registrais $x$ ir $y$, o rezultatą įrašyti į registrą $t$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := 1$ jei **bent vieno** iš $r[x][j]$ ir $r[y][j]$ reikšmė yra $1$, priešingu atveju priskirti $r[t][j] := 0$.
- $xor(t, x, y)$: Atlikti bitinęXOR operaciją su registrais $x$ ir $y$, o rezultatą įrašyti į registrą $t$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := 1$ jei **lygiai vieno** iš $r[x][j]$ ir $r[y][j]$ reikšmė yra $1$, priešingu atveju priskirti $r[t][j] := 0$.
- $not(t, x)$: Atlikti bitinę NOT operaciją su registru $x$, o rezultatą įrašyti į registrą $t$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := 1 - r[x][j]$.
- $left(t, x, p)$: Pastumti visus registro $x$ bitus į kairę per $p$ bitų ir rezultatą įrašyti į registrą $t$. Registro $x$ bitų postūmio į kairę per $p$ bitų rezultatas yra masyvas $v$, sudarytas iš $b$ bitų. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, $v[j] = r[x][j - p]$ jei $j \geq p$, priešingu atveju $v[j] = 0$. Kiekvienam $j$ $(0 \leq j \leq b - 1)$, priskirti $r[t][j] := v[j]$.

- $right(t, x, p)$: Pastumti visus registro $x$ bitus į dešinę per $p$ bitų ir rezultatą įrašyti į registrą $t$. Registro $x$ bitų postūmio į dešinę per $p$ bitų rezultatas yra masyvas $v$, sudarytas iš $b$ bitų. Kiekvienam $j$ $(0 \leq j \leq b-1)$, $v[j] = r[x][j+p]$ jei $j \leq b-1-p$, priešingu atveju $v[j] = 0$. Kiekvienam $j$ $(0 \leq j \leq b-1)$, priskirti $r[t][j] := v[j]$.
- $add(t, x, y)$: Sudėti registrų $x$ ir $y$ skaitines vertes ir rezultatą įrašyti į registrą $t$. Sudėtis atliekama moduliu $2^b$. Registro $x$ skaitinę vertę prieš atliekant operaciją pažymėkime $X$, registro $y$ – $Y$. Registro $t$ skaitinę vertę atlikus operaciją pažymėkime $T$. Jei $X + Y < 2^b$, registro $t$ bitams priskiriame tokias reikšmes, kad $T = X + Y$. Priešingu atveju – tokias, kad $T = X + Y - 2^b$.

Kristupas norėtų naujuoju procesoriumi išspręsti dviejų tipų uždavinius. Uždavinio tipas nurodomas skaičiumi $s$. Abiejų tipų uždaviniams turite sugeneruoti **programą**, t.y. aukščiau aprašytų instrukcijų seką.

Programos **įvestį** sudaro $n$ sveikųjų skaičių $a[0], a[1], \ldots, a[n-1]$, kiekvienas jų sudarytas iš $k$ bitų, t.y., $a[i] < 2^k$ ( $0 \leq i \leq n-1$). Prieš programos vykdymą visi šie skaičiai yra įrašomi į registrą $0$ taip, kad kiekvienam $i$ $(0 \leq i \leq n-1)$, $k$ bitų sekos $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ skaitinė vertė būtų lygi $a[i]$. Atkreipkite dėmesį, kad $n \cdot k \leq b$. Visiems kitiems registro $0$ bitams (t.y. tiems, kurių indeksai yra nuo $n \cdot k$ iki $b-1$ imtinai) ir visų kitų registrų bitams priskiriama reikšmė $0$.

Programos vykdymą sudaro visų jos instrukcijų vykdymas iš eilės. Įvykdžius paskutinę instrukciją, programos **išvestis** apskaičiuojama pagal galutines registro $0$ bitų reikšmes. Išvestis yra seka, sudaryta iš $n$ sveikųjų skaičių $c[0], c[1], \ldots, c[n-1]$, kur kiekvienam $i$ ( $0 \leq i \leq n-1$), $c[i]$ yra sekos, sudarytos iš registro $0$ bitų nuo $i \cdot k$ iki $(i+1) \cdot k - 1$, skaitinė reikšmė. Atkreipkite dėmesį, kad po programos įvykdymo likę registro $0$ bitai (turintys indeksus, ne mažesnius nei $n \cdot k$) ir visų kitų registrų visi bitai gali turėti bet kokias reikšmes.

- Pirma užduotis $(s = 0)$: raskite mažiausią skaičių tarp visų įvesties skaičių $a[0], a[1], \ldots, a[n-1]$. T.y., $c[0]$ turi būti mažiausias iš $a[0], a[1], \ldots, a[n-1]$. $c[1], c[2], \ldots, c[n-1]$ reikšmės gali būti bet kokios.
- Antra užduotis $(s = 1)$: išrikiuokite $a[0], a[1], \ldots, a[n-1]$ nemažėjančia tvarka. T.y., kiekvienam $i$ ( $0 \leq i \leq n-1$), $c[i]$ turi būti lygus $(1+i)$-ajam mažiausiam skaičiui iš $a[0], a[1], \ldots, a[n-1]$ (t.y. $c[0]$ yra mažiausias įvestas skaičius).

Pateikite Kristupui šias užduotis sprendžiančiais programas, kurių kiekviena sudaryta iš ne daugiau nei $q$ instrukcijų.

## Realizacija

Parašykite šią procedūrą:

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

- $s$: užduoties tipas.

- $n$: įvestyje esančių skaičių kiekis.
- $k$: kiekvieną įvestyje esantį skaičių sudarančių bitų skaičius.
- $q$: maksimalus leidžiamas operacijų skaičius.
- Ši procedūra iškviečiama lygiai vieną kartą ir turi sukonstruoti seką instrukcijų, kurios sprendžia duotą užduotį.

Ši procedūra turi sukonstruoti instrukcijų seką, iškviesdama vieną arba daugiau iš žemiau aprašytų procedūrų:

```
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)
```

- Kiekviena procedūra prideda atitinkamai $move(t, y)$, $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ arba $add(t, x, y)$ instrukciją prie programos.
- Visoms instrukcijoms $t$, $x$, $y$ reikšmės turi būti ne mažesnės nei $0$ ir ne didesnės nei $m - 1$.
- Visoms instrukcijoms $t$, $x$, $y$ reikšmės neprivalo būti skirtingos.
- $left$ ir $right$ instrukcijoms $p$ reikšmė turi būti ne mažesnė nei $0$ ir ne didesnė nei $b$.
- $store$ instrukcijai, $v$ ilgis turi būti lygus $b$.

Savo sprendimui testuoti galite naudoti šią procedūrą:

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

- Vertinant jūsų sprendimą, visi šios procedūros iškvietimai bus ignoruojami.
- Pavyzdinėje vertinimo programoje ši procedūra prie programos pridės $print(t)$ operaciją.
- Kai pavyzdinė vertinimo programa vykdydama jūsų programą pasieks $print(t)$ operaciją, ji išspausdins $n$ sveikųjų skaičių ( $k$ bitų ilgio), kuriuos sudaro pirmieji $n \cdot k$ registro $t$ bitų (žr. „Pavyzdinė vertinimo programa“).
- $t$ turi tenkinti $0 \leq t \leq m - 1$.
- Šios procedūros iškvietimas nedidina panaudotų instrukcijų skaičiaus.

Pridėjusi paskutinę instrukciją, `construct_instructions` turi baigti darbą. Tada programa vertinama su tam tikru testų skaičiumi, kurių kiekvienas yra apibūdinamas įvestimi, sudaryta iš $n$ ( $k$ bitų ilgio) sveikųjų skaičių $a[0], a[1], \ldots, a[n-1]$. Jūsų sprendimas įveikia duotą testą, jei programos išvestis $c[0], c[1], \ldots, c[n-1]$ tenkina šias sąlygas:

- Jei $s = 0$, $c[0]$ turi būti mažiausia vertė iš $a[0], a[1], \ldots, a[n-1]$.

- Jei $s = 1$, kiekvienam $i$ ($0 \leq i \leq n - 1$), $c[i]$ turi būti $(1 + i)$-asis mažiausias skaičius iš $a[0], a[1], \ldots, a[n - 1]$.

Vertinimo programa gali pateikti šiuos klaidų pranešimus:

- `Invalid index`: neteisingas (galimai neigiamas) registro indeksas buvo pateiktas kaip parametras $t$, $x$ arba $y$ kuriame nors procedūros iškvietime.
- `Value to store is not b bits long`: $v$, pateikto `append_store`, ilgis nėra lygus $b$.
- `Invalid shift value`: $p$ reikšmė, pateikta `append_left` arba `append_right` nėra nuo $0$ ir $b$ imtinai.
- `Too many instructions`: jūsų procedūra bandė iškviesti daugiau nei $q$ instrukcijų.

# Pavyzdžiai

## Pavyzdys nr. 1

Tarkime, kad $s = 0$, $n = 2$, $k = 1$, $q = 1000$. Įvedami du skaičiai $a[0]$ ir $a[1]$, kiekvienas jų sudarytas iš $k = 1$ bito. Prieš vykdant programą, $r[0][0] = a[0]$ ir $r[0][1] = a[1]$. Visiems kitiems procesoriaus bitams yra priskirta reikšmė $0$. Kai visos programos instrukcijos įvykdomos, turime gauti $c[0] = r[0][0] = \min(a[0], a[1])$, t.y. mažesnis iš $a[0]$ ir $a[1]$.

Galimos 4 programos įvestys:

- $1$-as atvejis: $a[0] = 0, a[1] = 0$
- $2$-as atvejis: $a[0] = 0, a[1] = 1$
- $3$-ias atvejis: $a[0] = 1, a[1] = 0$
- $4$-as atvejis: $a[0] = 1, a[1] = 1$

Pastebėkime, kad visiems 4 atvejams $\min(a[0], a[1])$ lygus bitiniam AND, atliekamam su $a[0]$ ir $a[1]$. Galime sukonstruoti programą, atliekančią tokius iškvietimus:

1. `append_move(1, 0)`, kuris prideda instrukciją nukopijuoti $r[0]$ į $r[1]$.
2. `append_right(1, 1, 1)`, kuris prideda instrukciją, kuri paima visus $r[1]$ bitus, paslenka juos į dešinę per $1$ bitą, ir išsaugo rezultatą atgal į $r[1]$. Kadangi kiekvienas skaičius yra $1$ bito ilgio, gauta $r[1][0]$ reikšmė yra lygi $a[1]$.
3. `append_and(0, 0, 1)`, kuris prideda instrukciją atlikti bitinį AND su $r[0]$ ir $r[1]$, ir tada išsaugoti rezultatą $r[0]$. Po šios instrukcijos įvykdymo, $r[0][0]$ yra priskiriama bitinio AND, atlikto su $r[0][0]$ ir $r[1][0]$ reikšmė, kuri yra lygi bitinio AND, atlikto su $a[0]$ ir $a[1]$ reikšmei, ko ir norėjome.

## Pavyzdys nr. 2

Tarkime, kad $s = 1$, $n = 2$, $k = 1$, $q = 1000$. Kaip ir ankstesniame pavyzdyje, yra tik 4 galimos programos įvesties reikšmės. Visiems 4 atvejams, $\min(a[0], a[1])$ yra bitinis AND, atliktas su $a[0]$ ir $a[1]$, ir $\max(a[0], a[1])$ yra bitinis OR, atliktas su $a[0]$ ir $a[1]$. Galimas sprendimas yra atlikti tokius iškvietimus:

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)`

Kai įvykdomos šios instrukcijos, $c[0] = r[0][0]$ yra priskirta $\min(a[0], a[1])$, ir $c[1] = r[0][1]$ yra priskirta $\max(a[0], a[1])$, kas ir išrikiuoja įvestį.

## Ribojimai

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

## Dalinės užduotys

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

## Pavyzdinė vertinimo programa

Pavyzdinė vertinimo programa įvestį skaito šiuo formatu:

- 1-oji eilutė: $s\ n\ k\ q$

Toliau skaitomos eilutės, kurios aprašo po vieną testą. Kiekvienas testas pateikiamas tokiu formatu:

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

Taip apibūdinamas testas, kurio įvestį sudaro $n$ sveikųjų skaičių $a[0], a[1], \ldots, a[n-1]$. Testų aprašymas užbaigiamas eilute, kurioje įvedamas tik skaičius $-1$.

Pavyzdinė vertinimo programa pirma iškviečia `construct_instructions(s, n, k, q)`. Jeigu šis kvietimas pažeidžia kurį nors sąlygoje aprašytą apribojimą, pavyzdinė vertinimo programa atspausdina vieną iš klaidos pranešimų, išvardintų skyrelio „Realizacija“ pabaigoje, ir baigia darbą. Kitu atveju pavyzdinė vertinimo programa pirmiausia iš eilės atspausdina kiekvieną instrukciją, pridėtą

`construct_instructions(s, n, k, q)` kvietimo. $store$ instrukcijoms $v$ atspausdinamas nuo indekso $0$ iki indekso $b-1$.

Toliau pavyzdinė vertinimo programa iš eilės vykdo testus. Kiekvienam testui ji paleidžia sukonstruotą programą su testo įvestimi.

Kiekvienai $print(t)$ instrukcijai, tarkime, kad $d[0], d[1], \ldots, d[n-1]$ yra sveikųjų skaičių seka, kurioje kiekvienam $i$ ($0 \le i \le n-1$), $d[i]$ yra skaitinė reikšmė bitų sekos nuo $i \cdot k$ iki $(i+1) \cdot k - 1$ registre $t$ (instrukcijos įvykdymo momentu). Tuomet vertinimo programa atspausdiną seką tokiu formatu:

`register` $t$: $d[0]\ d[1]\ \ldots\ d[n-1]$.

Įvykdžiusi visas instrukcijas pavyzdinė vertinimo programa išveda programos vykdymo rezultatą testui.

Jeigu $s = 0$, pavyzdinės vertinimo programos išvesties formatas kiekvienam testui bus toks:

- $c[0]$.

Jeigu $s = 1$, pavyzdinės vertinimo programos išvesties formatas kiekvienam testui bus toks:

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

Įvykdžiusi visus testus vertinimo programa išveda `number of instructions:` $X$, kur $X$ nurodo instrukcijų skaičių jūsų programoje.