

Stabilizatsiya (stbz)

Ixtiyoriy $x[0], x[1], \dots, x[m-1]$ va $y[0], y[1], \dots, y[m-1]$ **binar** massivlarning **go'zalligi** quyidagi tartibda hisoblanadi.

Har daqiqada quyidagi **protssess** bo'ladi deylik:

- $z^{\rightarrow 1}$ deb z massivning 1 marta o'ngga surilgan holatini aytamiz, ya'ni $z^{\rightarrow 1} = z[m-1], z[0], z[1], z[2], \dots, z[m-2]$
- Barcha $0 \leq i < m$ uchun, $x[i] := x[i] \& x^{\rightarrow 1}[i]$ hamda $y[i] := y[i] \& y^{\rightarrow 1}[i]$ amallarini bir vaqtda bajarish, bu yerda $\&$ bitwise AND amali.

Ikkala massivlar ham o'zgarishdan to'xtaganida protsess ham to'xtaydi.

Protssess boshlanishidan oldin siz quyidagi **almashtirtish**ni istalgancha bajara olasiz:

- Ixtiyoriy i tanlang va $x[i]$ va $y[i]$ qiymatlarini almashtirib qo'ying.

x va y massivlarning **go'zalligi** deb u massivlar aro xohlagancha almashtirishlar qilgandan so'ng protsess bajarilishiga ketadigan minimal vaqtga aytiladi.

Sizga uzunliklari n bo'lgan $a[0], a[1], \dots, a[n-1]$ va $b[0], b[1], \dots, b[n-1]$ **binar** massivlar berilgan.

Sizga ikki xil shakldagi so'rovlar beriladi:

- i, c, d – massivlarni o'zgartirish, $a[i] := c$ va $b[i] := d$
- l, r – Agar $m = r - l + 1$, $x = a[l \dots r]$ va $y = b[l \dots r]$ bo'lsa, ularning **go'zalligini** topish.

E'tibor bering: Protssess faqatgina x va y massivlarga ta'sir qiladi. Berilgan a va b massivlar faqatgina `update` so'rovi orqali o'zgarishi mumkin. Umuman aytganda, `query` so'rovlari bir biriga hech qanday ta'sir o'tqazmaydi.

Implementation details

Siz quyidagi uchta protsedurani dasturlashingiz kerak:

```
void init(int n, vector<int> a, vector<int> b)
```

- n : massivlar uzunligi.

- a, b : uzunligi n bo'lgan massivlar.
- Bu protsedura hech narsa qaytarmaydi.
- Bu protsedura aynan bir marta, barcha so'rovlardan oldin chaqiriladi.

```
int query(int l, int r)
```

- l, r : so'rovdagi oraliq.
- Bu protsedura $m = r - l + 1$, $x = a[l \dots r]$ va $y = b[l \dots r]$ bo'lsa x va y ning go'zalligini qaytarishi lozim.

```
void update(int i, int c, int d)
```

- i : 0 va $n - 1$ oraliqidagi indeks.
- $a[i] := c$ va $b[i] := d$ shaklidagi massivni o'zgartirish so'rovi.
- Bu protsedura hech narsa qaytarmaydi.

`query` va `update` protseduralari jami q marta chaqiriladi.

Example

Quyidagi chaqiruvni ko'raylik:

```
init(6, [1, 0, 1, 1, 1, 1], [0, 0, 1, 1, 1, 1])
```

Demak, $n = 5$, $a = [1, 0, 1, 1, 1]$ va $b = [0, 0, 1, 1, 1]$.

Birinchi so'rov quyidagicha bo'lsin:

```
query(0, 5)
```

Ya'ni, $x = a$ va $y = b$ massiv juftligining go'zalligi. 0- va 2-indexda alishtiruv qilsak nima bo'lishiga qaraylik: $x = [0, 0, 1, 1, 1]$, $y = [1, 0, 1, 1, 1]$. Endi protsess bajarilsa, massivlar bu ko'rinishda o'zgaradi:

- 1-daqiqa: $x = [0, 0, 1, 1, 1, 0]$, $y = [0, 0, 1, 1, 1, 1]$
- 2-daqiqa: $x = [0, 0, 1, 1, 0, 0]$, $y = [0, 0, 1, 1, 1, 0]$
- 3-daqiqa: $x = [0, 0, 1, 0, 0, 0]$, $y = [0, 0, 1, 1, 0, 0]$
- 4-daqiqa: $x = [0, 0, 0, 0, 0, 0]$, $y = [0, 0, 1, 0, 0, 0]$
- 5-daqiqa: $x = [0, 0, 0, 0, 0, 0]$, $y = [0, 0, 0, 0, 0, 0]$
- 6-daqiqa: $x = [0, 0, 0, 0, 0, 0]$, $y = [0, 0, 0, 0, 0, 0]$

5-daqidada oxirgi o'zgarish bo'lgan, shuning uchun protsessga 5-daqiqa ketdi dteymiz. Protsessga ketadigan vaqtni boshqa alishtiruvlar bilan bundan kamroq vaqt ketadigan qilishning iloji yo'qligini

isbotlash mumkin. Shuning uchun, so'rovga javob - 5.

Ikkinchi so'rov:

```
update(4, 0, 0)
```

a va b massivlarni o'zgartirish so'rovi. Bu so'rovdan so'ng: $a = [1, 0, 1, 1, 0, 1]$, $b = [0, 0, 1, 1, 0, 1]$

Uchinchi so'rov:

```
query(0, 4)
```

So'ralayotgan oraliq $x = [1, 0, 1, 1, 0]$ va $y = [0, 0, 1, 1, 0]$. Bu safar hech qanday alishtirish qilmasak nima bo'lishiga qaraylik:

- 1-daqida: $x = [0, 0, 1, 0, 0, 0]$, $y = [0, 0, 1, 0, 0, 1]$
- 2-daqida: $x = [0, 0, 0, 0, 0, 0]$, $y = [0, 0, 0, 0, 0, 0]$
- 3-daqida: $x = [0, 0, 0, 0, 0, 0]$, $y = [0, 0, 0, 0, 0, 0]$

2-daqidada oxirgi o'zgarish bo'ldi. Har qanday alishtirish qilsak ham protsessni bundan qisqaroq qilishning iloji yo'qligini isbotlash mumkin. Shuning uchun go'zallik 2 bo'ladi va so'rovga javobni 2 qaytarish kerak.

Constraints

- $2 \leq n, q \leq 200\,000$
- $0 \leq a[i], b[i] \leq 1$ (barcha $0 \leq i < n$ uchun)
- $0 \leq c, d \leq 1$
- $0 \leq l < r \leq n - 1$
- $0 \leq i \leq n - 1$

Subtasks

1. (10 ball) $n \leq 15$, $q = 1$
2. (11 ball) $n, q \leq 500$
3. (11 ball) $n, q \leq 2000$
4. (30 ball) $n, q \leq 200\,000$ (hech qanday `update` so'rovlari yo'q)
5. (38 ball) $n, q \leq 200\,000$

Sample Grader

Namunaviy grader ma'lumotlarni quyidagi tartibda o'qiydi:

- qator 1: n

- qator 2: $a[0] \ a[1] \ \dots \ a[n - 1]$
- qator 3: $b[0] \ b[1] \ \dots \ b[n - 1]$
- qator 4: q
- qator $4 + i$ ($0 \leq i < q$): so'rov

so'rov formati quyidagidan biri:

- $0 \ l \ r$ - `query(l, r)`
- $1 \ i \ c \ d$ - `update(i, c, d)`

Har bir bajarilgan so'rov uchun, namunaviy grader javoblarni quyidagi tartibda chiqaradi:

- qator $1 + i$: `query()` funksiyasi qaytargan soni.