Mozaika

Salma planuje pokolorować glinianą mozaikę na ścianie. Mozaika jest siatką $N \times N$, wykonaną z N^2 początkowo niepokolorowanych kwadratowych płytek o wymiarach 1×1. Rzędy mozaiki są ponumerowane od 0 do $N-1$ od góry do dołu, a kolumny są ponumerowane od 0 do $N-1$ od lewej do prawej. Kafelek w i-tym rzędzie i j-tej kolumnie ($0 \leq i < N$, $0 \leq j < N$) jest oznaczony jako (i, j). Każdy kafelek musi zostać pokolorowany na biało (oznaczany jako 0) lub czarno (oznaczany jako 1).

Aby pokolorować mozaikę, Salma najpierw wybiera dwie tablice X i Y o długości N, każdą składającą się z wartości 0 i 1, takich, że $X[0] = Y[0]$. Koloruje kafelki najwyższego rzędu (rząd 0) zgodnie z tablicą X tak, że kolor kafelka $(0, j)$ jest równy $X[j]$ ($0 \leq j < N$). Koloruje również kafelki w najbardziej lewej kolumnie (kolumna 0) zgodnie z tablicą Y tak, że kolor kafelka $(i, 0)$ jest równy $Y[i]$ ($0 \leq i < N$).

Następnie powtarza poniższe kroki, aż wszystkie kafelki zostaną pokolorowane:

- Znajduje dowolny *niepokolorowany* kafelek (i, j) taki, że jego sąsiad z góry (kafelek $(i-1, j)$) i sąsiad z lewej strony (kafelek $(i, j-1)$) są już *pokolorowani*.
- Następnie koloruje kafelek (i, j) na czarno, jeśli obaj sąsiedzi są biali, a w przeciwnym wypadku koloruje kafelek (i, j) na biało.

Można wykazać, że ostateczne kolory płytek nie zależą od kolejności, w jakiej Salma je koloruje.

Yasmin jest bardzo ciekawa kolorów płytek w mozaice. Zadaje Salmie Q zapytań, ponumerowanych od 0 do $Q-1$. W k-tym zapytaniu ($0 \leq k < Q$) Yasmin określa podprostokąt mozaiki za pomocą:

- Najwyższego wiersza $T[k]$ i najniższego wiersza $B[k]$ ($0 \leq T[k] \leq B[k] < N$),
- Najbardziej lewej kolumny $L[k]$ i najbardziej prawej kolumny $R[k]$ ($0 \leq L[k] \leq R[k] < N$).

Odpowiedzią na zapytanie jest liczba czarnych kafelków w tym podprostokącie. Konkretnie, Salma powinna dowiedzieć się, ile jest takich kafelków (i, j), że $T[k] \leq i \leq B[k]$, $L[k] \leq j \leq R[k]$, a kolor kafelka (i, j) jest czarny.

Napisz program, który odpowiada na zapytania Yasmin.

Szczegóły implementacyjne

Powinieneś zaimplementować poniższą funkcję.

```
std::vector<long long> mosaic(
    std::vector<int> X, std::vector<int> Y,
    std::vector<int> T, std::vector<int> B,
    std::vector<int> L, std::vector<int> R)
```

- X, Y: tablice o długości N opisujące kolory kafelków, odpowiednio w najwyższym rzędzie i najbardziej lewej kolumnie.
- T, B, L, R: tablice o długości Q opisujące zapytania zadane przez Yasmin.
- Ta funkcja powinna zwrócić tablicę C o długości Q taką, że $C[k]$ jest odpowiedzią na k-te zapytanie ($0 \leq k < Q$).
- Ta procedura jest wywoływana dokładnie raz dla każdego testu.

Ograniczenia

- $1 \leq N \leq 200\,000$
- $1 \leq Q \leq 200\,000$
- $X[i] \in \{0, 1\}$ i $Y[i] \in \{0, 1\}$ dla każdego i takiego, że $0 \leq i < N$
- $X[0] = Y[0]$
- $0 \leq T[k] \leq B[k] < N$ i $0 \leq L[k] \leq R[k] < N$ dla każdego k takiego, że $0 \leq k < Q$

Podzadania

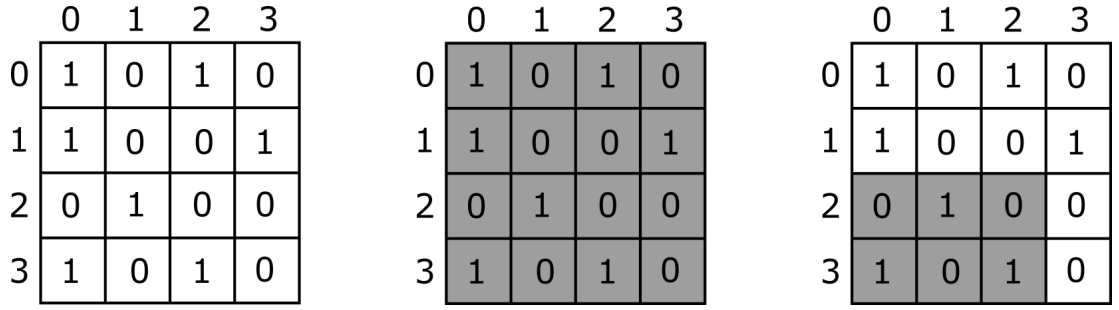
Podzadanie	Punktacja	Dodatkowe ograniczenia
1	5	$N \leq 2; Q \leq 10$
2	7	$N \leq 200; Q \leq 200$
3	7	$T[k] = B[k] = 0$ (dla każdego k takiego, że $0 \leq k < Q$)
4	10	$N \leq 5000$
5	8	$X[i] = Y[i] = 0$ (dla każdego i takiego, że $0 \leq i < N$)
6	22	$T[k] = B[k]$ i $L[k] = R[k]$ (dla każdego k takiego, że $0 \leq k < Q$)
7	19	$T[k] = B[k]$ (dla każdego k takiego, że $0 \leq k < Q$)
8	22	Brak dodatkowych ograniczeń.

Przykład

Rozważmy poniższe wywołanie.

```
mosaic([1, 0, 1, 0], [1, 1, 0, 1], [0, 2], [3, 3], [0, 0], [3, 2])
```

Przykład ten ilustrują poniższe obrazki. Na lewym obrazku widoczne są kolory płytek w mozaice. Środkowe i prawe zdjęcie przedstawiają podprostokąty, o które Yasmin zapytała odpowiednio w pierwszym i drugim zapytaniu.



Odpowiedzi na zapytania (czyli liczba jedynek w zacieniowanych prostokątach) wynoszą odpowiednio 7 i 3. Dlatego procedura powinna zwrócić $[7, 3]$.

Przykładowy program oceniający

Format wejścia:

```
N
X[0] X[1] ... X[N-1]
Y[0] Y[1] ... Y[N-1]
Q
T[0] B[0] L[0] R[0]
T[1] B[1] L[1] R[1]
...
T[Q-1] B[Q-1] L[Q-1] R[Q-1]
```

Format wyjścia:

```
C[0]
C[1]
...
C[S-1]
```

S jest długością tablicy C zwróconej przez `mosaic`.