Wild operations (wild)

Filippo vrea să-i testeze abilitatea lui Francesco de a gestiona operații complexe pe array-uri, de aceea i-a dat un array $A_0, ..., A_{N-1}$ de lungime N.

Acum Filippo îi va cere lui Francesco să efectueze niște operații pe array, unde fiecare operație poate fi una dintre:

- *schimbarea* valorii lui A_p în x, pentru un număr întreg x și un indice valid p.
- *perturbarea* intervalului $[l, r]$, adică setarea $A_p = \max(A_p, A_{p-1})$ **simultan** pentru toți $l < p \leq r$.

În orice moment, Filippo îi poate cere lui Francesco să-i spună valoarea lui A_p pentru un index valid p.

Francesco este foarte ocupat, așa că a decis să-ți ceară ajutorul pentru a răspunde la întrebările lui Filippo.

Implementare

Trebuie să trimiți un singur fișier cu extensia `.cpp`.

⇦ Printre atașamentele pentru această problemă vei găsi un șablon `wild.cpp` cu o implementare exemplu.

Trebuie să implementezi următoarele funcții:

C++
```cpp
void init(int N, vector<int> A);
```

- Această funcție este apelată o singură dată, la începutul execuției programului tău.
- Numărul întreg N este lungimea tabloului.
- Tabloul A, indexat de la 0 la $N - 1$, este tabloul inițial ales de Filippo.

C++
```cpp
void change(int p, int x);
```

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo efectuează o schimbare.
- Numărul întreg p este indexul valorii modificate în tablou.
- Numărul întreg x este noua valoare de atribuit.

C++
```cpp
void perturb(int l, int r);
```

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo perturbă un interval.
- Numărul întreg l este capătul din stânga al intervalului perturbat de Filippo.
- Numărul întreg r este capătul din dreapta al intervalului perturbat de Filippo.

C++
```cpp
int calc(int p);
```

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo cere valoarea unui element din tablou.

- Numărul întreg p este indexul elementului cerut de Filippo.
- Funcția ar trebui să returneze valoarea lui A_p după aplicarea tuturor operațiilor anterioare.

Evaluator Exemplu

O versiune simplificată a evaluatorului folosit în timpul corectării este disponibilă în directorul aferent acestei probleme. O poți folosi pentru a-ți testa soluțiile local. Evaluatorul exemplu citește datele de intrare din `stdin`, apelează funcția pe care trebuie să o implementezi și scrie în `stdout` în următorul format.

Fie Q numărul total de modificări, perturbări și întrebări făcute de Filippo.
Atunci, fișierul de intrare este format din $2 + Q$ linii, conținând:

- Linia 1: numerele întregi N, Q.
- Linia 2: N numere întregi $A_0, ..., A_{N-1}$, valorile inițiale ale tabloului.
- Linia $3 + i$ ($0 \leq i < Q$): 2 sau 3 numere întregi, într-unul dintre următoarele formate:
  - 1 p x: înseamnă că Filippo schimbă A_p la x.
  - 2 l r: înseamnă că Filippo perturbă intervalul $[l, r]$;
  - 3 p: înseamnă că Filippo cere valoarea lui A_p.

Fișierul de ieșire este format din Q_3 linii (unde Q_3 este numărul de apeluri către `calc`) conținând valorile returnate de funcția `calc`.

Constrângeri

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ pentru toți $0 \leq i < N$.
- $0 \leq p < N$ în fiecare apel către `change` și `calc`.
- $0 \leq l < r \leq N - 1$ în fiecare apel către `perturb`.
- $1 \leq x \leq 10^9$ în fiecare apel către `change`.

Punctaj

Programul tău va fi testat pe mai multe cazuri de test grupate în subprobleme. Pentru a obține punctele pentru o subproblemă, trebuie să rezolvi corect toate cazurile de test din ea.

Fie Q_1 numărul de apeluri către funcția `change` într-un caz de test, atunci:

- **Subtask-ul 0 [0 puncte]**: Exemplu.
- **Subtask-ul 1 [15 puncte]**: Funcția `change` nu este niciodată apelată; $l = 0$, $r = N - 1$ în fiecare apel către `perturb`.
- **Subtask-ul 2 [16 puncte]**: $A_i \leq 10$ pentru toți $0 \leq i < N$ și $x \leq 10$ pentru toate apelurile către `change`.
- **Subtask-ul 3 [13 puncte]**: Apelurile către funcția `change` nu scad valorile ($x \geq A_p$), $Q_1 \leq 1000$ și $l = 0$, $r = N - 1$ în fiecare apel către `perturb`.
- **Subtask-ul 4 [22 puncte]**: Funcția `change` nu este niciodată apelată.
- **Subtask-ul 5 [14 puncte]**: Apelurile către funcția `change` nu scad valorile ($x \geq A_p$), $Q_1 \leq 1000$.
- **Subtask-ul 6 [20 puncte]**: Fără constrângeri adiționale.

Exemple de intrare/ieșire

stdin	stdout
10 28 5 1 7 8 3 2 5 6 9 4 1 1 1 1 0 1 2 0 1 2 2 6 1 6 5 2 2 9 2 2 5 2 4 5 1 4 5 2 3 8 1 8 4 3 0 1 6 3 1 4 1 2 5 7 1 0 3 2 4 5 1 6 3 3 0 3 1 3 2 3 3 3 4 3 5 3 6 3 7 3 8 3 9	1 3 1 7 8 1 8 3 6 4 9

Explicație

Începem cu tabloul $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Evenimentul 1: Filippo schimbă A_1 la 1 (era deja 1): noul tablou este $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Evenimentul 2: Filippo schimbă A_0 la 1: noul tablou este $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Evenimentul 3: Filippo perturbă $[0, 1]$: noul tablou este $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Evenimentul 4: Filippo perturbă $[2, 6]$: noul tablou este $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

De la evenimentul 19 încolo, Filippo doar cere valori din tablou fără a efectua modificări sau perturbări. În acest punct tabloul este $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.