Wild operations (wild)

Maus Binna möchte Maus Stofls Fähigkeiten im Umgang mit verrückten Array-Operationen testen. Deshalb hat sie ihm ein Array $A_0, ..., A_{N-1}$ der Länge N gegeben.

Nun wird Binna Stofl bitten, einige Operationen auf dem Array durchzuführen. Jede Operation kann eine der folgenden sein:

- *Ändere* (`change`) den Wert von A_p zu x, für eine ganze Zahl x und einen gültigen Index p.
- *Störe* (`perturb`) den Bereich $[l, r]$, d.h. setze $A_p = \max(A_p, A_{p-1})$ **gleichzeitig** für alle $l < p \leq r$.

Binna kann Stofl jederzeit bitten, ihr den Wert von A_p für einen gültigen Index p zu sagen.

Stofl ist sehr beschäftigt, also hat er beschlossen, dich um Hilfe zu bitten, um Binnas Fragen zu beantworten.

Implementierung

Du musst eine einzelne Datei mit der Erweiterung `.cpp` einreichen.

> ⇦ Unter den Anhängen zu dieser Aufgabe findest du eine Vorlage `wild.cpp` mit einer Beispielimplementierung.

Du musst die folgenden Funktionen implementieren:

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

- Diese Funktion wird einmal aufgerufen, zu Beginn der Ausführung deines Programms.
- Die ganze Zahl N ist die Länge des Arrays.
- Das Array A, indiziert von 0 bis $N - 1$, ist das ursprüngliche Array, das Binna ausgewählt hat.

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

- Diese Funktion wird während der Ausführung deines Programms mehrmals aufgerufen, wenn Binna eine Änderung vornimmt.
- Die ganze Zahl p ist der Index des geänderten Wertes im Array.
- Die ganze Zahl x ist der neue Wert, der zugewiesen werden soll.

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

- Diese Funktion wird während der Ausführung deines Programms mehrmals aufgerufen, wenn Binna einen Bereich stört.
- Die ganze Zahl l ist das linke Ende des von Binna gestörten Bereichs.
- Die ganze Zahl r ist das rechte Ende des von Binna gestörten Bereichs.

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

- Diese Funktion wird während der Ausführung deines Programms mehrmals aufgerufen, wenn Binna den Wert eines Elements des Arrays abfragt.

- Die ganze Zahl p ist der Index des von Binna abgefragten Elements.
- Die Funktion sollte den Wert von A_p zurückgeben, nachdem alle vorherigen Operationen angewendet wurden.

Beispielgrader

Eine vereinfachte Version des Graders ist im Verzeichnis dieser Aufgabe verfügbar. Du kannst sie verwenden, um deine Lösungen lokal zu testen. Der Beispielgrader liest Eingabedaten aus `stdin`, ruft die von dir zu implementierenden Funktionen auf und schreibt in `stdout` im folgenden Format.

Sei Q die Gesamtzahl der von Binna vorgegebenen Änderungen, Störungen und Abfragen.
Dann besteht die Eingabedatei aus $2 + Q$ Zeilen, die Folgendes enthalten:

- Zeile 1: die ganzen Zahlen N, Q.
- Zeile 2: N ganze Zahlen $A_0, ..., A_{N-1}$, die Anfangswerte des Arrays.
- Zeile $3 + i$ ($0 \leq i < Q$): 2 oder 3 ganze Zahlen in einem der folgenden Formate:
  - 1 p x: bedeutet, Binna ändert A_p zu x.
  - 2 l r: bedeutet, Binna stört den Bereich $[l, r]$;
  - 3 p: bedeutet, Binna fragt den Wert von A_p ab.

Die Ausgabedatei besteht aus Q_3 Zeilen (wobei Q_3 die Anzahl der Aufrufe von `calc` ist), die die von der Funktion `calc` zurückgegebenen Werte enthalten.

Einschränkungen

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ für alle $0 \leq i < N$.
- $0 \leq p < N$ in jedem Aufruf von `change` und `calc`.
- $0 \leq l < r \leq N - 1$ in jedem Aufruf von `perturb`.
- $1 \leq x \leq 10^9$ in jedem Aufruf von `change`.

Punktevergabe

Dein Programm wird auf mehreren Testfällen getestet, die in Teilaufgaben gruppiert sind. Um die Punkte für eine Teilaufgabe zu erhalten, musst du alle Testfälle darin korrekt lösen.

Sei Q_1 die Anzahl der Aufrufe der Funktion `change` in einem Testfall, dann gilt:

- **Teilaufgabe 0 [0 Punkte]**: Beispiel.
- **Teilaufgabe 1 [15 Punkte]**: Die Funktion `change` wird nie aufgerufen; $l = 0$, $r = N - 1$ in jedem Aufruf von `perturb`.
- **Teilaufgabe 2 [16 Punkte]**: $A_i \leq 10$ für alle $0 \leq i < N$ und $x \leq 10$ für alle Aufrufe von `change`.
- **Teilaufgabe 3 [13 Punkte]**: Aufrufe der Funktion `change` verringern die Werte nicht ($x \geq A_p$), $Q_1 \leq 1000$ und $l = 0$, $r = N - 1$ in jedem Aufruf von `perturb`.
- **Teilaufgabe 4 [22 Punkte]**: Die Funktion `change` wird nie aufgerufen.
- **Teilaufgabe 5 [14 Punkte]**: Aufrufe der Funktion `change` verringern die Werte nicht ($x \geq A_p$), $Q_1 \leq 1000$.
- **Teilaufgabe 6 [20 Punkte]**: Keine weiteren Einschränkungen.

Beispiele

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

Erklärung

Wir starten mit dem Array $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Ereignis 1: Binna ändert A_1 zu 1 (es war bereits 1): das neue Array ist $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Ereignis 2: Binna ändert A_0 zu 1: das neue Array ist $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Ereignis 3: Binna stört $[0, 1]$: das neue Array ist $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Ereignis 4: Binna stört $[2, 6]$: das neue Array ist $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

Ab Ereignis 19 fragt Binna nur noch Werte im Array ab, ohne Änderungen oder Störungen durchzuführen. An diesem Punkt ist das Array $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.