Horses

Mansur adora criar cavalos, tal como os seus antigos ancestrais faziam. Ele tem agora a maior manada de cavalos do Cazaquistão. No entanto, este não foi sempre o caso. Há N anos atrás, Mansur era apenas um dzhigit (cazaque para *um homem jovem*) e tinha um único cavalo. Ele sonhava ganhar muito dinheiro e finalmente tornar-se um bai (cazaque para *um homem muito rico*).

Numeramos os anos de 0 a $N-1$ por ordem cronológica (i.e., o ano $N-1$ é o mais recente). A meteorologia de cada ano influenciou o crescimento da manada. Para cada ano i, Mansur recorda-se de um coeficiente de crescimento $X[i]$, que é um inteiro positivo. Se começasse o ano i com h cavalos, então terminaria o ano com $h \cdot X[i]$ cavalos na sua manada.

Os cavalos só podiam ser vendidos no final de cada ano. Para cada ano i, Mansur recorda-se também de um inteiro positivo $Y[i]$: o preço pelo qual ele podia vender um cavalo no final do ano i. No final de cada ano, era possível vender um número arbitrário de cavalos, cada um ao mesmo preço $Y[i]$.

Mansur questiona-se qual é a maior quantia de dinheiro que ele poderia ter agora se tivesse escolhido os melhores momentos para vender os seus cavalos durante os N anos. Você tem a honra de ser um convidado do Mansur no seu toi (cazaque para *dia festivo*), e ele pediu-lhe para responder a esta pergunta.

A memória do Mansur melhora ao longo da noite, e por isso ele faz uma sequência de M atualizações. Cada atualização irá mudar apenas um dos valores $X[i]$ ou um dos valores $Y[i]$. Depois de cada atualização ele pergunta-lhe novamente a quantia de dinheiro que podia ter ganho ao vender os seus cavalos. As atualizações do Mansur são cumulativas: cada uma das suas respostas deve ter em conta todas as atualizações anteriores. Note que um único $X[i]$ ou $Y[i]$ pode ser atualizado múltiplas vezes.

As respostas para as questões do Mansur pode ser enormes. Para evitar trabalhar com números grandes, só tem de reportar as respostas módulo $10^9 + 7$.

Exemplo

Suponha que existem $N = 3$ anos, com a seguinte informação:

	0	**1**	**2**
X	2	1	3
Y	3	4	1

Para estes valores iniciais, Mansur pode ganhar o máximo se vender ambos os seus cavalos no final do ano 1. Todo o processo irá parecer-se com o seguinte:

- Inicialmente, Mansur tem 1 cavalo.

- Depois do ano 0 ele terá $1 \cdot X[0] = 2$ cavalos.
- Depois do ano 1 ele terá $2 \cdot X[1] = 2$ cavalos.
- Ele pode agora vender esses dois cavalos. O lucro total será $2 \cdot Y[1] = 8$.

De seguida, suponha que existe uma $M = 1$ atualização: mudar o valor de $Y[1]$ para 2.

Depois desta atualização, teremos:

	0	**1**	**2**
X	2	1	3
Y	3	2	1

Neste caso, uma das soluções ótima é vender um cavalo depois do ano 0 e então três cavalos depois do ano 2. Todo o processo irá parecer-se com o seguinte:

- Inicialmente, Mansur tem 1 cavalo.
- Depois do ano 0 ele terá $1 \cdot X[0] = 2$ cavalos.
- Ele pode agora vender um dos seus cavalos por $Y[0] = 3$, e ficar ainda com um cavalo.
- Depois do ano 1 ele terá $1 \cdot X[1] = 1$ cavalo.
- Depois do ano 2 ele terá $1 \cdot X[2] = 3$ cavalos.
- Ele pode agora vender esses dois cavalos por $3 \cdot Y[2] = 3$. O total de dinheiro será $3 + 3 = 6$.

Tarefa

São dados N, X, Y e uma lista de atualizações. Antes da primeira atualização, e depois de cada atualização, calcule a máxima quantia de dinheiro que Mansur pode receber pelos seus cavalos, módulo $10^9 + 7$.

- `init(N, X, Y)` — O avaliador irá chamar esta função no início e exatamente uma vez.
  - `N`: o número de anos.
  - `X`: um vetor (array) de tamanho N. Para $0 \leq i \leq N - 1$, $X[i]$ contém o valor do coeficiente de crescimento para o ano i.
  - `Y`: um vetor de tamanho N. Para $0 \leq i \leq N - 1$, $Y[i]$ contém o preço de um cavalo no final do ano i.
  - Note que ambos os vetores `X` e `Y` especificam os valores iniciais dados por Mansur (antes de qualquer atualização).
  - Depois de `init` terminar, os vetores `X` e `Y` permanecem válidos, e pode modificar os seus conteúdos se assim o desejar.
  - A função deve devolver a máxima quantia de dinheiro que Mansur pode conseguir para estes valores iniciais de X e Y, módulo $10^9 + 7$.

- `updateX(pos, val)`
  - `pos`: um inteiro no intervalo $0, \ldots, N-1$.
  - `val`: o novo valor de $X[\texttt{pos}]$.
  - A função deve devolver a máxima quantia de dinheiro que o Mansur pode conseguir depois desta atualização, módulo $10^9 + 7$.
- `updateY(pos, val)`
  - `pos`: um inteiro no intervalo $0, \ldots, N-1$.
  - `val`: o novo valor de $Y[\texttt{pos}]$.
  - A função deve devolver a máxima quantia de dinheiro que o Mansur pode conseguir depois desta atualização, módulo $10^9 + 7$.

Pode assumir que todos os valores iniciais, bem como os valores atualizados de $X[i]$ e $Y[i]$ estão entre 1 e 10^9 inclusive.
Depois de chamar `init`, o avaliador irá chamar `updateX` e `updateY` diversas vezes. O número total de chamadas a `updateX` e `updateY` será M.

Subtarefas

subtarefa	pontos	N	M	restrições adicionais
1	17	$1 \le N \le 10$	$M = 0$	$X[i], Y[i] \le 10$, $X[0] \cdot X[1] \cdot \ldots \cdot X[N-1] \le 1,000$
2	17	$1 \le N \le 1,000$	$0 \le M \le 1,000$	nenhuma
3	20	$1 \le N \le 500,000$	$0 \le M \le 100,000$	$X[i] \ge 2$ e $val \ge 2$ para `init` e `updateX` respetivamente
4	23	$1 \le N \le 500,000$	$0 \le M \le 10,000$	nenhuma
5	23	$1 \le N \le 500,000$	$0 \le M \le 100,000$	nenhuma

Avaliador exemplo

O avaliador exemplo lê o input a partir de um ficheiro `horses.in` no seguinte formato

- linha 1: `N`
- linha 2: `X[0]` … `X[N - 1]`
- linha 3: `Y[0]` … `Y[N - 1]`
- linha 4: `M`
- linhas 5, …, `M` + 4: três números `type pos val` (`type=1` para `updateX` e `type=2` para `updateY`).

O avaliador exemplo escreve o valor de retorno de `init`, seguido pelos valores de retorno de todas as chamadas a `updateX` e `updateY`.