# Eligiendo Carreras

*Contribución de Lautaro Lasorsa*

## Descripción del problema

Ana y Joe son una pareja y ambos están pensando en escoger su carrera universitaria. Ya saben que irán ambos a la misma universidad. Y necesitan tu ayuda.

Inicialmente ninguno de los dos está interesado en ninguna carrera, pero eso va cambiando a medida que leen los planes de estudio de las carreras. De las carreras que les interesan en un momento dado, les resulta indistinto cursar cualquiera de ellas. Y quieren cursar juntos el comienzo de la carrera lo más posible.

Como aún están leyendo los planes de estudio, las carreras que les interesan van cambiando. Quieren saber siempre, después de cada cambio de opinión de alguno de ellos, cuál es la mayor cantidad de materias que pueden cursar juntos en forma consecutiva desde el comienzo de la carrera, y las carreras con las que ocurriría esto. Si hay más de una posibilidad, les es indistinto cuál sea.

La universidad a la que piensan asistir es muy particular, ya que en ella los alumnos de cada carrera cursan una sola materia por cuatrimestre. Por lo tanto, cada carrera puede describirse como una lista de enteros, donde el *i*-ésimo entero identifica la materia cursada en el *i*-ésimo cuatrimestre.

Cada materia se identifica con un número entre 1 y 1.000.000.000. Si en un cuatrimestre las dos carreras elegidas no tienen la misma materia, entonces ya no importa que en siguientes cuatrimestres coincidan las materias, ya que solamente les interesa maximizar la cantidad de materias en común en forma consecutiva desde el comienzo de la carrera.

Esta universidad es muy peculiar, y pueden existir 2 carreras distintas en las que se cursen exactamente las mismas materias en el mismo orden.

## Detalles de implementación

Debes implementar **dos funciones**:

- `inicializar(N)`, una función que será llamada una única vez al inicio del programa. Recibe un entero `N` que indica la cantidad de veces que Ana o Joe actualizarán sus intereses. No debe retornar nada.
- `actualizar(car, num, per, op, elegidas)`. Será llamada *N* veces, luego de la llamada a `inicializar`. Sus parámetros son:
  - `car`: un arreglo de enteros que describe la carrera, indicando sus materias en orden.
  - `num`: un entero que identifica la carrera
  - `per`: un entero que indica quién realiza la actualización. Un 0 indica que Ana cambia de opinión, y un 1 indica que Joe cambia de opinión.
  - `op`: un entero que indica cuál es el cambio. Un 0 indica que esa carrera ya no es de su interés, y un 1 agrega una nueva carrera de interés. Una persona solamente quita de su lista de intereses una carrera que le interesaba en ese momento, y no agregan una carrera que ya estaba entre sus intereses.
  - `elegidas`: Un arreglo de enteros en el cual se deben almacenar exactamente dos valores: el número que identifica la carrera elegida por Ana, y el que identifica la carrera elegida por Joe, en ese orden.

  La función debe retornar en un entero la máxima cantidad de cuatrimestres que pueden cursar juntos, que podría ser 0. Si alguno de los dos no está interesado en ninguna carrera, la función debe retornar -1, y se deben colocar dos -1 en el arreglo `elegidas`.

## Evaluador local

El evaluador local lee de la entrada estándar una primera línea con el valor $N$, y realiza la llamada `Inicializar(N)`.

Luego lee $N$ actualizaciones. En la actualización $i$ lee:

- Una línea con 4 enteros `num`, `per`, `op` y $M_i$, siendo $M_i$ la cantidad de cuatrimestres que dura la carrera correspondiente a esta actualización.
- Otra línea con exactamente $M_i$ enteros, correspondientes al parámetro `car`.

Por cada actualización, realiza la correspondiente llamada a `actualizar` y escribe en la salida estándar dos líneas: la primera con el valor retornado por la función, y la segunda con el contenido almacenado en el arreglo `elegidas`

## Restricciones

- $1 \leq M_i$ para todo $i$
- $M_1 + M_2 + \cdots + M_N \leq 200.000$
- $1 \leq$ `car[i]`, `num` $\leq 1.000.000.000$

## Ejemplos

El siguiente es un posible ejemplo de entrada y salida al evaluador, para un programa correcto:

Entrada

```
10
1 0 1 5
1 2 3 4 5
1 1 1 5
1 2 3 4 5
5 1 1 6
1 2 3 4 5 9
1 1 0 5
1 2 3 4 5
1 0 0 5
1 2 3 4 5
1000 0 1 7
1 2 3 6 7 8 10
50 0 1 3
100 101 102
1000 0 0 7
1 2 3 6 7 8 10
10000 1 1 4
100 103 105 1100
10000 1 0 4
100 103 105 1100
```

Salida

```
-1
-1 -1
5
1 1
5
1 1
5
1 5
-1
-1 -1
3
1000 5
3
1000 5
0
50 5
1
50 10000
0
50 5
```

En cambio, si recibe:

```
6
1 0 1 10
1 2 3 4 5 6 7 8 9 10
2 0 1 5
11 12 13 14 15
3 0 1 7
100 99 98 97 96 95 94
4 0 1 1
100
5 1 1 1
314
3 1 1 7
100 99 98 97 96 95 94
```

Una salida correcta podría ser:

```
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
0
1 5
7
3 3
```

## Puntuación

Se recibe el 60 % del puntaje por el correcto valor de retorno de la función, y el 40 % restante por además dar un valor correcto en el arreglo `elegidas`.

## Subtareas

1. $1 \leq M_1 + M_2 + \cdots + M_N \leq 500$ (10 ptos.)
2. $1 \leq M_1 + \cdots + M_N \leq 3000$ (20 ptos.)
3. Nunca ninguno de los 2 elimina una carrera de sus intereses (25 ptos.)
4. Sin más restricción (45 ptos.)