Afluentes navegables

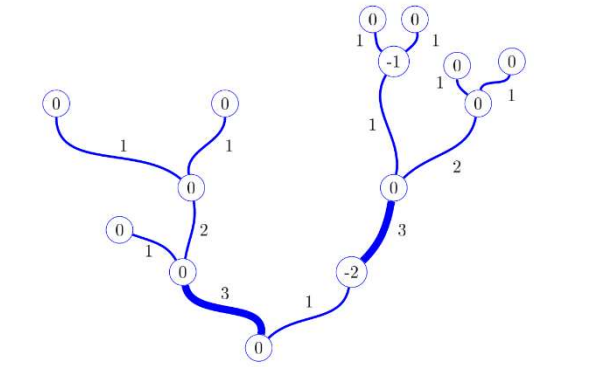
El objetivo de este control es familiarizarse con el procesamiento de árboles binarios.

1) El problema

El Ministerio de Ordenación Fluvial nos ha pedido implementar un programa que permita calcular los tramos de río navegables de una cuenca fluvial. Para ello, nos proporciona el siguiente conocimimiento sobre la estructura de una cuenca:

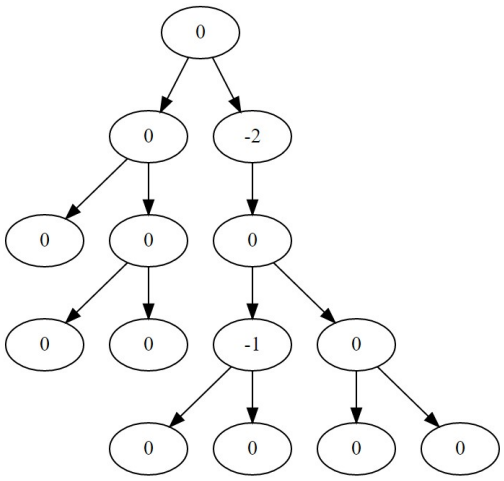
- Una cuenca fluvial está formada por *tramos de río*.
- Un tramo de río es *navegable* si contiene un caudal mayor o igual a 3 m^3/s.
- Los afluentes que nacen de los manantiales llevan un caudal de un 1 m^3/s.
- Los tramos de río pueden confluir para dar lugar a un nuevo tramo de mayor caudal. Los tramos que confluyen se denominan *afluentes*. Se considera que siempre confluyen exactamente dos afluentes. Cada vez que dos afluentes confluyen se calcula el caudal del tramo resultante como la suma de los caudales de dichos afluentes.
- La cuenca puede contener *embalses*, que hacen decrecer en una determinada cantidad el caudal del tramo saliente. En un embalse pueden confluir dos afluentes, o solamente un tramo de río. Así mismo, nótese que el caudal de un tramo no puede ser negativo, pero sí 0 (en este caso el tramo de río estará *seco*).

La figura que se muestra a continuación representa esquemáticamente una cuenca fluvial: la cuenca del río "Aguas Limpias", que contiene 2 tramos navegables.



Los círculos etiquetados con 0 representan la confluencia de dos afluentes. Los etiquetados con un número negativo representan un embalse. En este caso, la magnitud del número representa la cantidad en la que el embalse hace decrecer el caudal.

En base a esta información, hemos decidido representar las cuencas como árboles binarios de enteros. Por ejemplo, el siguiente árbol representa la cuenca "Aguas Limpias":



2) **Trabajo a realizar**

Se debe construir un programa que lea una serie de filas, cada una representando una cuenca, e imprima por la salida estándar el número de tramos navegables de cada cuenca.

Los árboles se codifican en la entrada de acuerdo con el siguiente criterio:

- El árbol vacío se representa como #
- Un árbol simple con raíz *A* se representa como [*A*]
- Un árbol compuesto con raíz A se representa como (τ_i A τ_d), donde τ_i es la representación del hijo izquierdo, y τ_d la del derecho.

De esta forma, la cuenca "Aguas Limpias" se codificará como:

```
(([0]0([0]0[0]))0(#-2(([0]-1[0])0([0]0[0]))))
```

Ejemplo de entrada / salida:

Entrada	Salida
(([0]0([0]0[0]))0(#-2(([0]-1[0])0([0]0[0]))))	2
(([0]0([0]0[0]))0(#-2((([0]-4[0])0[0])0(([0]-2[0])0[0]))))	1
[0]	0

Se proporciona el archivo `main.cpp` en el que se implementa la lógica de entrada / salida necesaria. El código proporcionado no debe modificarse.

Hay que añadir a dicho archivo la implementación de la siguiente función:

```
// Devuelve el número de tramos navegables en una cuenca.
// Parámetros:
//     cuenca: La cuenca representada como un árbol binario
// Resultado:
//     Número de afluentes navegables en la cuenca
// Precondición: En 'cuenca' hay una representación válida de
//               una cuenca fluvial (esta precondición no es
//               necesario comprobarla)
int numeroAfluentesNavegables(const Arbin<int>& cuenca);
```

Aparte de esta función, podrán añadirse todas aquellas funciones auxiliares que se consideren necesarias.