L06: Montañas binarescentes

Estructuras de Datos
Facultad de Informática - UCM

Para realizar este ejercicio, descarga la plantilla que se proporciona en este enunciado (icono del clip al lado del título) o a través del Campus Virtual.

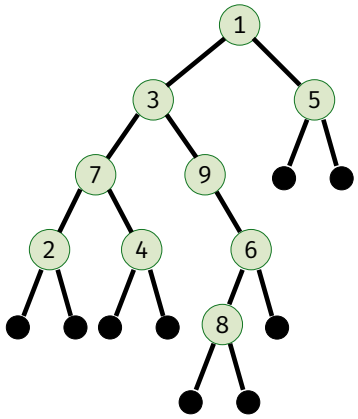
La entrega de este ejercicio consiste en un único fichero `.cpp` que se subirá a *DOMjudge*. Podéis subir tantos intentos como queráis. Se tendrá en cuenta el último intento con el veredicto `CORRECT` que se haya realizado antes de la hora de entrega por parte de alguno de los miembros de la pareja.

No olvidéis poner el nombre de los componentes de la pareja en el fichero `.cpp` que entreguéis. Solo es necesario que uno de los componentes de la pareja realice la entrega.

Evaluación: La práctica se puntuará de 0 a 10 si ha obtenido el veredicto `CORRECT` por parte del juez. En caso contrario, la calificación será de 0. No se realizará evaluación por pares en esta práctica.

Las montañas ***binarescentes*** tienen una curiosa —aunque ordenada– red de caminos de senderismo. En su cima hay un hito del cual salen dos caminos que descienden por la ladera. Cada uno de esos caminos desemboca en un parador, o en otro hito. Si desemboca en un hito, el camino se bifurca en otros dos caminos descendentes que pueden desembocar en un parador o en otro hito, y así sucesivamente. Todas las ramificaciones desembocan eventualmente en un parador.

Por ejemplo, la siguiente figura representa la red de caminos de una montaña binarescente. Los círculos numerados representan hitos, y los círculos de color negro representan paradores:



Me gusta presumir ante mis amigos de lo largas que son mis rutas de senderismo. Para poder demostrarlo hago una foto de los hitos que voy encontrando por el camino y luego las publico en las redes sociales. Por eso, más que llegar muy alto, me interesa que mi ruta atraviese el máximo número posible de hitos. Eso sí: la ruta ha de empezar en un parador, terminar en otro parador distinto, y no debe atravesar dos veces el mismo hito.

Por ejemplo, en el árbol de la figura superior, uno de mis caminos preferidos sería el que empieza en uno de los paradores situados por debajo del hito 2. Desde ahí subiría hasta el hito 3, y bajaría hasta uno de los paradores situados por debajo del hito 8. En total atravesaría 6 hitos.

Dada una montaña binarescente, ¿cuál sería el número máximo de hitos que podría atravesar en una ruta que empiece y acabe en un parador?

1. En la plantilla que se proporciona en el Campus Virtual, implementa la siguiente función:

   ```
   int max_hitos_visitados(const BinTree<int> &montanya);
   ```

   que devuelva el número máximo de hitos que se podrían atravesar en una ruta dentro de la montaña binarescente recibida como parámetro.

2. Determina el coste en tiempo de la función `max_hitos_visitados`. Para ello plantea una recurrencia que aproxime el coste de esta función, suponiendo que el árbol binario correspondiente a la montaña es equilibrado.

3. Determina el coste en tiempo de la misma función, pero planteando una recurrencia que suponga que el árbol binario es degenerado, es decir, que cada nodo tiene todos sus descendientes en el mismo lado.

4. Sube el ejercicio al problema *DOMjudge* con identificador `L06` para comprobar si tu solución es correcta.

Entrada

La entrada contiene varias líneas, cada una de ellas describiendo una montaña binarescente. El carácter punto (`.`) representa un parador. Una cadena de la forma `(izq h dch)` determina un hito cuyo número es `h`, del cual salen dos caminos descendentes, que llegan a los hitos o paradores descritos por `izq` y `dch`.

La entrada finaliza con una montaña con un único parador (es decir, la cadena " . "), que no se procesa.

Salida

Para cada caso se escribirá un número entero que indica el máximo número de hitos que pueden recorrerse en una ruta que empiece y termine en un parador.

Entrada de ejemplo

```
((((. 2 .) 7 (. 4 .)) 3 (. 9 ((. 8 .) 6 .))) 1 (. 5 .))
((((. 9 .) 3 .) 2 (. 4 .)) 1 ((. 6 .) 5 ((. 8 .) 7 .)))
(((. 3 .) 2 .) 1 .)
(. 1 .)
.
```

Salida de ejemplo

```
6
7
3
1
```