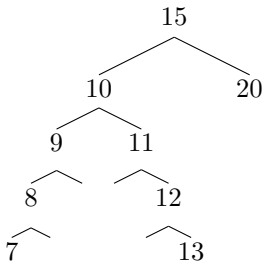
08 - Diámetro y distancias en un BST (EVALUACIÓN)

Descripción

Consideremos el siguiente árbol binario de búsqueda de números enteros usando el orden `less<int>`:



Diremos que la *distancia* $dist(a, b)$ entre dos elementos a y b de un árbol binario de búsqueda es la longitud del camino que los une (únicamente puede existir un camino). Por ejemplo $dist(15, 20) = 1$, $dist(9, 20) = 3$ y $dist(11, 13) = 2$. Además, consideramos que la distancia entre un elemento y él mismo es 0, es decir, $dist(a, a) = 0$ para todo elemento a del árbol binario de búsqueda.

Basándonos en esta idea de distancia, podemos definir la noción de *diámetro* de un árbol binario de búsqueda como la máxima distancia entre 2 de sus elementos. En el ejemplo anterior el diámetro es 6, que coincide con la distancia entre el 7 y el 13.

Extiende la clase `bst` vista en clase (fichero `bst.h`) para incorporar los siguientes métodos públicos:

- `int diametro() const`
- `int distancia(const T& a,const T& b) const`

Estos métodos devuelven el diámetro y la distancia entre dos elementos respectivamente. Los dos métodos deben tener un coste $\in O(n)$, donde n es el número de elementos en el árbol binario de búsqueda. La implementación de `diametro` debe realizarse utilizando un método privado `diametro_rec` recursivo.

Importante: A la hora de realizar el envío al juez, debes elegir el fichero `.cpp` junto con todos los ficheros cabecera `.h` que necesites, concretamente el fichero `bst.h` modificado. Si esto os da algún problema, recomiendo combinar todo el código fuente en un único fichero `main.cpp` que contenga la definición de la clase `bst.h` modificada además de la función `main` y demás funciones auxiliares que necesitéis.

Entrada

La entrada comenzará con una linea conteniendo un número natural N que indica la cantidad casos de prueba que vamos a considerar. Cada caso de prueba comienza con una línea representando un BST: primero contendrá un numero natural M, que es el número de elementos en el BST, y va seguido de M números enteros separados por espacios. Se debe partir de un BST vacío (de enteros y con el orden `less<int>`) e insertar estos M elementos en dicho orden para obtener el BST final. A continuación aparecerá una línea

con un número natural D indicando el número de distancias a calcular. Le seguirán D líneas, cada una conteniendo dos números A y B separados por un espacio sobre los que se deberá calcular la distancia. Los elementos A y B para calcular la distancia siempre aparecerán en el BST.

Salida

Por cada caso de prueba se debe mostrar una primera línea conteniendo el diámetro del árbol considerado. A continuación le seguirán D líneas con la distancia entre los elementos A y B. Después de cada caso de prueba debéis incluir una línea en blanco.

Ejemplo de entrada

```
2
3 10 5 15
2
5 10
15 5
9 15 10 20 9 11 8 12 7 13
3
20 9
7 9
12 13
```

Ejemplo de salida

```
2
1
2

6
3
2
1

```