lower-bound

Queremos añadir a la clase `set` un nuevo método

```
std::pair<bool,T> lower_bound(T const& e) const;
```

que dado un elemento `e` devuelva si existen en el conjunto elementos mayores o iguales que `e` y en caso afirmativo cuál es el menor de ellos.

Por ejemplo, si `s` es un `set` que denota el conjunto {2,8,20,32,47}, la llamada `s.lower_bound(12)` devolvería <true,20>. En cambio, la llamada `s.lower_bound(60)` devolvería <false,?> (en este caso la segunda componente del par está indefinida).

En el problema se modificará la clase `set` del fichero `set_eda.h` que se proporciona en el campus. Se añadirá a la clase un método público `lower_bound` que reciba el elemento, y un método protegido que reciba el elemento y un puntero a la raíz del árbol. El método protegido recorrerá el árbol y calculará los valores pedidos.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso consta de cuatro líneas. En la primera aparece el número $N > 0$ de elementos que se insertarán en el conjunto. En la segunda aparecen esos N números (en ningún orden concreto). En la tercera línea aparece el número M de preguntas que se harán. Y en la cuarta línea, esas preguntas, cada una de las cuales consiste en un número.

La entrada termina con un `0`.

Salida

Para cada caso se escribirá una línea por cada pregunta, con el *lower bound* correspondiente, si existe. Si no existe se escribirá `NO HAY`.

Detrás de cada caso se escribirán tres guiones, `---`.

Entrada de ejemplo

```
5
10 20 30 40 50
4
12 20 31 60
5
50 40 30 20 10
4
12 20 31 60
0
```

Salida de ejemplo

```
20
20
40
NO HAY
---
20
20
40
NO HAY
---
```

Autor: Isabel Pita