Los *k* elementos mayores

Dada una serie de elementos ordenables, posiblemente muy larga, queremos encontrar los k elementos *distintos* mayores, donde k puede ser mucho más pequeño que el número de elementos de la serie, y queremos sacar partido de ello.

Para conseguirlo sigue los siguientes pasos:

1. Modifica la clase `SetTree` añadiendo, al menos, métodos para consultar (`min`) y eliminar (`erase_min`) el menor elemento de un conjunto no vacío.

   ```
   template <typename T>
   class SetTree {
   public:

     T min() const;      // <-- nuevos métodos
     void erase_min();

   };
   ```

2. Resuelve cada caso de prueba encontrando los k elementos mayores de cada secuencia de N elementos que se encuentra en la entrada. Se espera que la complejidad en espacio adicional de la resolución de un caso de prueba esté en $O(k)$.

3. Analiza el coste de tu solución tanto en tiempo como en espacio adicional.

Entrada

Cada caso de prueba está formado por tres líneas. La primera contendrá el carácter `N` si los elementos de la serie son números, o el carácter `P` si los elementos son palabras. La segunda línea contendrá el valor $k > 0$, que será menor que el número de elementos (distintos) de la serie. La tercera línea contendrá los elementos de la serie (posiblemente con repeticiones). Si son números estarán en el rango $[0..10^9]$, y el fin de la serie vendrá indicado con un `-1`. Si son palabras, estarán formadas por no más de 30 caracteres de la 'a' a la 'z', y el final de la serie estará indicado con la palabra `FIN`.

Salida

Para cada caso de prueba se escribirá una línea con el conjunto que contiene los k elementos mayores de la serie, sin repeticiones y ordenados de menor a mayor, tal y como hace el método `display()` de la clase `SetTree`.

Entrada de ejemplo

```
N
3
1 8 3 14 5 -1
P
2
maria luis marta juan alberto FIN
N
3
1 2 3 4 5 6 6 6 -1
```

Salida de ejemplo

```
{5, 8, 14}
{maria, marta}
{4, 5, 6}
```

Autor

Alberto Verdejo