Algoritmo de Visibilidad

Cuando se pinta una parrilla de productos en los frontales web de las tiendas de comercio electrónico es necesario filtrar aquellos productos que se han quedado sin stock para facilitar al usuario el encontrar productos que pueda comprar.

Para ello se deben comprobar todas las tallas de cada producto para ver si tienen stock, y en caso de que ninguna talla tenga stock, ese producto no deberá mostrarse en la parrilla.

Existen dos casuísticas especiales:

- La primera es cuando una talla está marcada como back soon, en este caso, aunque la talla no tenga stock, el producto debe mostrarse igual, puesto que es un producto que va a volver a estar a la venta cuando vuelva a entrar stock.
- La segunda es cuando un producto tiene tallas "especiales", en este caso el producto solo estará visible si al menos una talla especial y una no especial tiene stock (o está marcada como back soon). Si solo tienen stock (o están marcadas como back soon) tallas de uno de los dos grupos el producto no debe mostrarse. Esta casuística se utiliza en productos compuestos, por ejemplo, un cojín que consta de un relleno y una funda, solo se muestra si hay stock tanto del relleno como de la funda, si no hay stock de ninguno o solo del relleno o solo de la funda, entonces el cojín no se muestra.

Se pide desarrollar una aplicación en Spring boot, con un endpoint que ejecute un algoritmo que lea tres ficheros en formato csv que simulan las tablas en base de datos:

- **product.csv**: fichero con los siguientes campos:
  - **id**: identificador de producto.
  - **sequence**: posición del producto en la parrilla.
- **size.csv**: fichero con los siguientes campos:
  - **id**: identificador de talla.
  - **productId**: identificador de producto.
  - **backSoon**: true si la talla es back soon o false en caso contrario.
  - **special**: true si la talla es especial o false en caso contrario.
- **stock.csv**: fichero con los siguientes campos:
  - **sizeId**: identificador de talla.
  - **quantity**: unidades disponibles en almacén de dicha talla.

Y que ofrezca como salida la lista de identificadores de producto, ordenados por el campo sequence, que cumplan las condiciones de visibilidad explicadas anteriormente y separados por comas.

Se han de añadir también un conjunto de test para validar que el algoritmo está funcionando correctamente, según lo esperado.

Ejemplo:

product.csv:

```
1, 10
2, 7
3, 15
4, 13
5, 6
```

size.csv:

```
11, 1, true, false
12, 1, false, false
13, 1 , true, false
21, 2, false, false
22, 2, false, false
23, 2, true, true
31, 3, true, false
32, 3, true, false
33, 3, false, false
41, 4, false, false
42, 4, false, false
43, 4, false, false
44, 4, true, true
51, 5, true, false
52, 5, false, false
53, 5, false, false
54, 5, true, true
```

stock.csv:

```
11, 0
12, 0
13, 0
22, 0
31, 10
32, 10
33, 10
41, 0
42, 0
43, 0
44, 10
51, 10
52, 10
53, 10
54, 10
```

Salida:

```
5,1,3
```

Estructuras de datos utilizadas en el algoritmo

Una vez realizado el problema del algoritmo de visibilidad, comenta qué estructuras de datos (Listas, Sets, etc) has seleccionado para resolverlo y porque las has considerado como las más adecuadas en cada caso.

Complejidad temporal del algoritmo

Una vez resuelto el algoritmo de visibilidad. ¿Qué complejidad temporal expresada en notación “O” crees que tiene? ¿Consideras que se podría mejorar de alguna manera?