A - Para la estructura que contiene los datos de partida utilizo un *array*, puesto que es una serie de números definidos, con tamaño fijo que no requiere borrado o insertado de elementos. Requiere menos espacio que una lista y es fácil de iterar.

El problema pide imprimir el siguiente elemento mayor. Esto se puede realizar directamente en el método donde obtengo el "output" y no sería necesario almacenar los datos como yo he hecho en la resolución del problema. Sin embargo, siguiendo un poco la filosofía de Uncle Bob, me gusta intentar que cada función haga una cosa. Por tanto, hice una para obtener el "output" y otra para imprimir el resultado. El "output"lo guarde en un *LinkedHashMap*, puesto que permite mantener el orden de cada elemento del valor inicial (como *key*) y asociarle su siguiente elemento mayor (como *value*).

B – Al haber dos bucles *for* anidados la complejidad sería O(n^2) en el peor de los casos. El bucle final para imprimir el output no aumenta esta complejidad.

En cuanto a si considero que se podría mejorar el código, puede ser. Si lo que se quiere es reducir el número de recursos y solo imprimir el "output" haría lo que comente en el punto A. Imprimir el resultado según recorro el *array* (eliminando así un *map* y una función con un nuevo bucle).

En cuanto a mejorar la complejidad temporal he buscado información (puesto que si supiese como mejorarlo lo habría hecho desde el principio) y se puede realizar con un *Stack*, reduciendo la misma a O(n) según la fuente de información (aunque tiene un bucle *while* dentro del bucle *for*, con lo que en el peor de los casos en mi opinión estaríamos hablando de O(n^2) también).

Aun así, adapté mi método por curiosidad. Midiendo tiempos con *System.nanoTime()* con ambos datos de ejemplo, me da un tiempo 3 veces superior usando el *Stack*. Además, este método no permite almacenar los elementos por orden según el array inicial, por lo que habría que retocar el método de impresión (por ejemplo, iterando el *array* obtienes la *key* del *map* con la que obtienes el *value*).

A continuación, dejo el método que genera el output adaptado con *Stack*:

```
static Map<Integer, Integer> getNextGreaterElementOutput(int numbers[]) {
    final int arrayLength = numbers.length;
    Stack<Integer> stack = new Stack<>();
    Map<Integer, Integer> output = new LinkedHashMap<Integer, Integer>();

     for (int i = arrayLength - 1; i >= 0; i--) {
         if (!stack.empty()) {
             while (!stack.empty() && stack.peek() <= numbers[i]) {
                 stack.pop();
             }
         }
         output.put(numbers[i], stack.empty() ? -1 : stack.peek());
         stack.push(numbers[i]);
     }
   return output;
}
```