Intersection over two sorted arrays

The intersection algorithm we wrote in Section 3 of Work n°9 is inefficient for large arrays. Try using your function as in Listing 1.

```
1 array1 = [x for x in range(100000)] #The numbers from 0 to 99999
2 array2 = [x for x in range(1000)]   #The numbers from 0 to 999
3 print(common_elements(array1, array2))
```

Listing 1: Trying the intersection on long arrays.

The explanation is the following: the algorithm we wrote compares each element of array1 to each element of array2. Thus, it makes, in this case, $100\ 000 \times 1000 = 100\ 000\ 000$ comparisons. When the two arrays are sorted, there are many ways to make less comparisons to compute their intersection. You can try to implement one of the following better solutions:

- For each element in array1, we search through the whole array array2 that this element is or is not there. In fact, when an element of array2 has already been found in previous iterations (and, is thus in the intersection), we can remove it from array2 before continuing the searches.
- For each element in array1, instead of performing a naive search of this element through the array array2, we can perform a binary search:

  https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
- Instead of performing two nested loops, it is possible to perform only one while loop. We track an index i in array1, an index j in array2. When array1[i] is equal to array2[j], we add this element to the intersection, we increment both i and j. When array1[i] is less than array2[j], we only increment i. When array2[i] is more than array2[j], we only increment j. We then iterate until i and j reach the end of their arrays.