2 – I use the hash function (key mod len(A) to organize the list A in an array. If there’s any collision, I use the open adressing method.

After it, I use the same hash function on the list B, but inserting the elements in the same array of list A, but this time if there’s a collision and key == arr[hash(key)], I use the chaining method, otherwise I will use the open addressing method.

To build this array will take O(n+n) = O(n).

After the array is ready, I check if at every index there’s a linked list with an even size (meaning that both list had the specific key), than I check if the elements are in the linked list are equal (worst case all the elements

3-

1. 7

/\

4 21

/ /\

2 16 29

\

19

b - 16

/\

4 21

/ /\

2 19 29

4 –

In a post-order we have that the root is the last element of the array. With in-order and knowing the root, we can know each elements are on the left side of the root and each ones are on the right side.

5 –

A – Yes, it’s a binary search tree cause we have that every Node keeps the rule that every children from left is smaller and the right children is always bigger. The rule keeps from the root to the leafs, meaning that all the elements from the left side of the root are smaller than it and all the elements at the right side of the root are bigger than it.