Questions about the practice:

1. Is the tree that is created from the node files (dict * .dat) complete or almost complete? Justify your answer.

The trees created in the part 3 of the practice are almost complete. Depending on the parameter we use to compile them, 'B' or 'N' . The difference will be set with the parameter introduced before execution:

A **balanced** (B) binary tree is the binary tree where the depth of the two subtrees of every node never differ by more than 1.

A **complete** (N) binary tree is a binary tree whose all levels except the last level are completely filled and all the leaves in the last level are all to the left side.

2. a) What is the relationship between the "shape" of a tree and its traversing modes?

Taking in consideration that the functions that go through the branches and counting the depth of the tree are done by recursive functions. Only the order of elements inserted in a data structure can affect it, if the data structure cares about the order, which is the case.

b) Can you tell if a binary search tree is well constructed according to how it is traversed?

No, in certain cases, for example inOrder it can be camouflaged by the way it is traversed because its root, left, right. But when we traverse it in PostOrder/PreOrder you can reconstruct the tree from the output and see if it was correctly constructed or not.

3. Compare and describe the differences between the trees generated by the executables p4_e3 with the last argument B or N (number of nodes, depth, routes, etc.).

With the param 'B' the nodes are ordered in memory and inserted so that the tree's created in the most **balanced** way possible making it theoretically faster to work (O(log(n))), reading and creating the tree files. In the other way, with the param 'N' **(not balanced)**, we will get a tree of nodes ordered in the way that it was read from the file.

In the following picture we can see the depth difference which goes from 3 to 5 (‘B’ to ‘N’). The number of nodes is the same, that doesn't change because the file we are reading is the same. We can appreciate a tiny difference in the search execution time but it’s too minor (in this case which are only 10 nodes) for being considered ($1x10^{-6}$ seconds).