Exercise n.4

We thought about the BaceFook network as a n-ary tree, in which each person can have several friends. So, starting from a root, it has n children, so n friends, and each of these children has several other friends as children and so on. In other words, every node has as friends its root and its children. The root has as friends only its children.

Every tree has to be composed at least by 2 nodes (so the case in which a user has no friends is not contemplated).

We also suppose (even if is not very realistic) that there are no repetitions in the tree.

We have to solve a minimum vertex cover problem: every node or its parent has to be marked (so must have the software) and in particular we want to find the minimum number of nodes to mark.

We implemented a simple TreeNode class in order to solve the problem, with methods add\_child(), height(), degree() and give\_software(). The method give\_software() returns a tuple constituted by a counter and a dictionary. The dictionary contains the names of the people (that are the values of the nodes) with a Boolean value that indicates if that person should have the software or not, while the counter indicates the number of total software that have to be distributed.

We divide the problem in subproblems: we divide the tree in subtrees and for each vertex, we need to find the size of the smallest vertex cover in subtree rooted at that vertex.

We have to consider 2 cases: one in which we include the root and one in which we don’t, and choose the solution with the minimum number of marked nodes. So, we call the function recursively on the children in the case in which we include the root and on the children’s children in the case we exclude the root and then take in account the one with the minimum number of distributed software.

So, with dynamic programming we compute both cases simultaneously, avoiding the double call for each child.

We save the minimum number of software that have to be distributed in the variable count. If it’s already calculated, we don’t compute again the function, but return directly it and the dictionary containing the list of all the people network (memoization phase).

We initially set the value false on the node on which the function is called. Then, we put it as true if the minimum number is the one considering the root, otherwise we set true all the children values and so we update the dictionary.

This dynamic programming solution actually solves the problem in time θ(n).