README

Problem 4

Introduction

The problem is an example of a vertex cover problem on a tree. So ,for simplicity, in this and in all the documents and files related to the problem number 4, we will refer to the vertex cover on a tree problem and not to the bacefook problem.

Description of the algorithm

The algorithm that we used is dynamic and takes time O(n)
In the following paragraphs we will describe how the algorithm works and what are the steps followed in order to build a dynamic algorithm.

How the problem is divided in subproblems:

The optimal solution for a tree, is obtained considering the optimal solutions of the subtrees rooted in the children and grandchildren of the root.

How the optimal solutions of the subproblems are combined in order to solve a problem of bigger size

Define VC(r) as the minimum vertex cover for the tree rooted in r.
For each node i of the tree we consider two cases:

1. The node i is inserted in the vertex cover, in this case the optimal solution of his children must be considered.
   $VCin(i) = 1 + \sum_{k=1}^{children(i)} VC(k)$
2. The node i is not inserted in the vertex cover.
   In this case, all the children of i have to be inserted in the vertex cover, and the optimal solutions of his grandchildren must considered.
   $VCout(i) = children(i) + \sum_{k=1}^{grandchildren(i)} VC(k)$

At the end the minimum of this two quantites is considered:
$VC(i) = min(VCin(i), VCout(i))$

Borderline cases:

1. If the tree is empty, the solutions is 0
2. If the node i has no children, the solution for VC(i) is 0

In which order the subproblems are solved:

Recursion is used in order to visit the tree, so the problems are solved in the same order of an inorder visit of a tree.

Notes

VC(i) could be evaluated multiple times, because there is overlapping between subproblems.
So when the algorithm computes VC(i) for a node i it memorizes it in the node in order to avoid this problem.

How to test the script

The function that evaluates the vertex cover for the tree is placed in the **vertex_cover_tree.py** file.
This function need as parameter a tree, that have to be an istance of the Tree class, that we implemented in the **tree.py** file.
In **test.py** the function can be tested, useing a function that allows to build a random tree.