Preliminary assignment - Paths in a Graph

Author: Anders Nylund

February 11, 2017

Application number: xxxxxx

Abstract

This document is the answer of applicant Anders Nylund to the Preliminary assignment "Paths in a Graph". The assignment is part of the admission process to Computer Science at University of Helsinki.

Task 1 Solution

The method InfPath takes a graph and an integer as argument and checks if an infinite path can be found, starting from the node with the index of the passed integer. InfPath returns the value of the method FindNodesRecursive. As InfPath takes only a graph and an integer as argument, FindNodesRecursive-method needed to be created as it adds one argument more.

The arguments in the methods are passed by value, not reference. Every node has an iterable list of nodes that it has an arc to.

```
1.  procedure InfPath(G, v)
2.     return FindNodesRecursive(G, v, [])
3.
4.  procedure FindNodesRecursive(G, v, checkedNodes)
5.     if checkedNodes contains v
6.        return true
7.     else
8.        checkedNodes.add(v)
9.        foreach endNode in v.endNodes
10.          if FindNodesRecursive(G, endNode, checkedNodes)
11.             return true
12.       return false
```

Instead of having comments in between the pseudo code, line numbers has been added before every line. This allows explanation of each row individually without making the pseudo code hard to read.

1. Declaration of method InfPath that takes a graph and an integer as argument.
2. InfPath returns the return value of FindNodesRecursive(). Here the graph G, starting node v and an initialized empty list is passed as argument.
3. -
4. Declaration of FindNodesRecursive. The arguments are given the same names as when passing them, except for the empty list, which is now called checkedNodes.
5. An if statement that searches for v in the list of checked nodes. The list of checked nodes consist of the previously checked nodes. If the current node has already been checked, it means that an infinite path has been found
6. Return true if the current node was found
7. If the statement on line 5 was false the following block of code will be executed
8. Add the current node the list of checked nodes

9. Iterate through every end node of the current node
10. Make a recursive call to FindNodesRecursive and evaluate the return value
11. If making a recursive call returns true, return true
12. Finally, if the execution has made it to this point and no infinite paths were found, the method returns false

The algorithm creates a 'tree' of recursive calls. For every end node in the current node in iteration, a new call to find the end nodes of the current node in the iteration is made. The recursive method is built to return true as fast as possible when an infinite path is found in the graph.

The call to the recursive method is made for each arc that start from a node.

As the algorithm does not evaluate the graph in any way, finding the node is dependent on the size of the graph. The time complexity if making sure there is no infinite path is

$$\sum_{i=1}^{n} 2^{-}$$

i.e. $O(n^2)$

Task 2

```
Some fancy code
```