This problem set is for students enrolled in **COMP2007**.
It is due at **11:59pm on Sunday 22th September 2013** (Week 8).

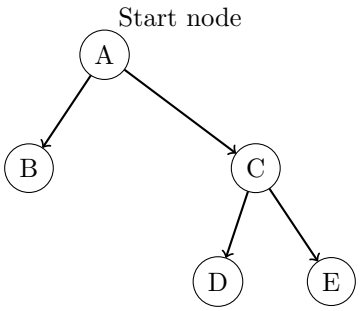
**Problem 1 (40 marks)**

You work for a company that specializes in network security. Your company has developed a sophisticated product that monitors activity on the network and detects malicious software that attempts to infect computers and access information on them. You are the algorithms expert in the company, and your job is to find ways to bypass this security product, so that the company can improve it, if needed.

After looking at the product you realize that it performs well for small networks but it may not scale well for large networks. Therefore if you can design a malicious program (let's call it a virus) that manages to infect many computers on a network as fast as possible, then you can show that the security product is not effective: By the time the virus is detected, it has infected many machines and has copied private files from them.

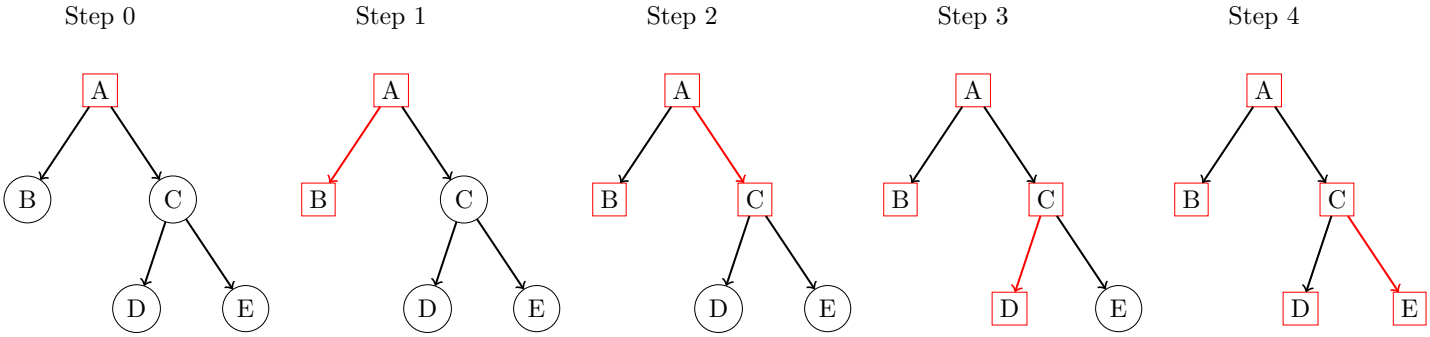
You already have a virus that is capable of infecting machines and jumps from one machine to the next if they are connected on the network by a direct link. However your virus can only work on infecting one computer at the time. Otherwise it creates too much network traffic from a single machine, and it will be detected quickly. So the problem you need to solve is the following: Each infected machine will attempt to infect all of its neighbors, one by one. Which one do you start from? In what order do you infect your neighbor machines? Once a neighbor machine is infected, it will also start infecting its own neighbors, independently. It takes one time-step to infect one neighbor machine. Your objective is to infect all machines on the network in minimum number of steps.

We assume that the network we want to attach is a tree (connected, undirected, no cycles). An example network is shown in the next figure.

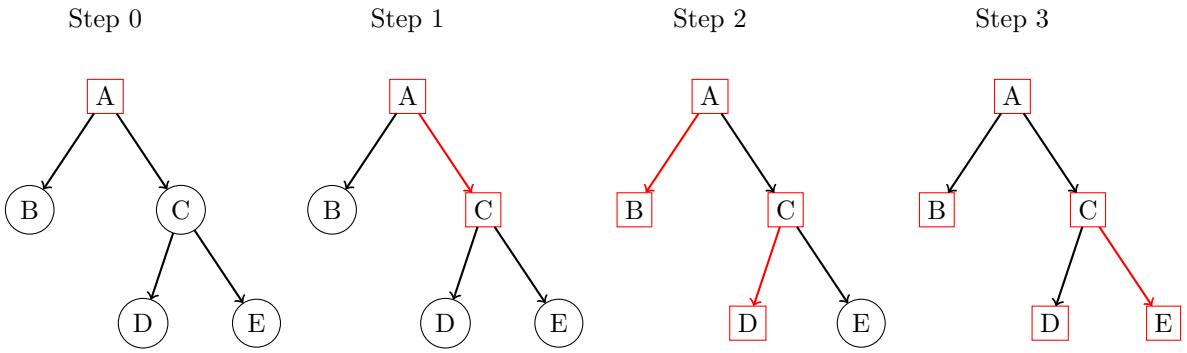



The infection starts with a given node. This is the machine that you have access to. The start node will be fixed (given to you as part of the input). We assume that at time 0, the start node $A$ is infected. At time 1 one of its neighbors will be infected, and our algorithm needs to choose which one is infected first, second and so on. Let's assume that our algorithm chooses to infect the neighbors "from left to right" as shown in the figure. In our example we can choose to infect B first and then C. In that case, at time step 1, A and B are infected. At time step 2, A, C and B are infected. At time step 3, we choose to infect D. And at time step 4, we infect E. Overall this takes 4 steps.

In the following figure, infected nodes are shown as red boxes. A red edge from node X to Y means that X infected Y in that particular step. Only red/infected nodes can have red edges out of them. Each infected node can have only one red edge in each step.



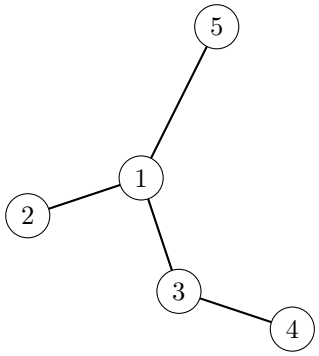
The best strategy however would be for A to infect C first, in step 1. Then in step 2, A infects B and C infects D (independently, in parallel). In step 3, C infects D, and we are done in 3 steps.




The network infection spreads in steps, as described above. In step 0, only the root node is infected. In every step, every infected node in the network will choose one neighbor (that is not infected yet) and will infect it with the virus. The process stops when all nodes are infected.

Your goal is to design an algorithm that determines the minimum number of steps required in order to infect the entire network. Your input is an undirected graph $T$ that is always a tree, and a distinguished node $r$ in the tree, which will be the root of the tree, where the infection will start. Note that the tree is not necessarily binary, it can be any arbitrary tree. Make your algorithm as efficient as possible. Your implementation needs to parse input provided in the format described in the example below:

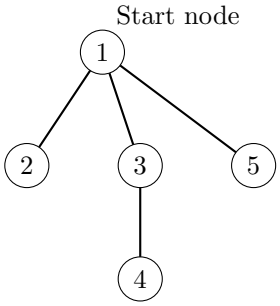
network topology




Input:

```
5
1
3 4
1 3
2 1
1 5
```

Tree with start node at the root




Output: 3

The first line in the input description contains the number of nodes $n \geq 1$ in the tree. The nodes will always be numbered as $1, 2, \ldots, n$. The second line contains the node that will be used as the root (the node where the infection starts). After that, we have a list of undirected edges, one edge per line, in the form `x y` (node id followed by a space, followed by another node id). Edges may be given to you in any order and any orientation (do not assume that they are provided from "top to bottom" or from "left to right"). Edges are undirected, which means if edge $xy$ is in the input then $x$ can infect $y$ and $y$ can infect $x$.

Your task:

- Design an algorithm for this problem (Hint: use dynamic programming)
- Implement your algorithm (and make sure it works by testing it on your own test cases)
- Explain how your algorithm works (in plain English)
- Analyse the running time of your algorithm

You will need to submit / complete

1. your implementation (source code)
2. complete a "test-quiz" on elearning. The quiz will give you some random input instances, and you will need to enter the output of your implementation.
    - Note 1: You should not count on these input cases to test your implementation. You need to do your own testing, and only attempt these input cases when your algorithm is ready. You can only do this test once.

- make sure you can parse large inputs from a given file.
- The test quiz will be available before the deadline, and must be completed by the submission deadline.

3. a pdf file (write-up) with
   - your description and analysis of the algorithm (including running time)

**Submission**
Submission links will appear on elearning on Thursday.
You can implement your program in any of the following programming languages: C, C++, Java, Python.