WORKSHEET

A4.5 Trees

Trees and graphs · A level · OCR H446 1.4.2, AQA 7517 4.2.5.1, Eduqas A500QS 1.1 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Trees as connected graphs with no cycles, rooted trees and their vocabulary, binary trees and typical uses.

- A tree is a graph
- Rooted trees
- Where rooted trees are used
- A rooted tree in Python
- Is it a tree?
- Binary trees

Questions

6 marks in all

1. Which is the definition of a tree? [1 mark]

- ☐ **A** A connected, undirected graph with no cycles
- ☐ **B** A directed graph in which every vertex has two children
- ☐ **C** Any graph with a root
- ☐ **D** An undirected graph in which every vertex has the same degree

2. A tree has 20 vertices. How many edges does it have? [1 mark]

3. In a rooted tree, which node has no parent? [1 mark]

- ☐ **A** The root
- ☐ **B** Every leaf
- ☐ **C** Every internal node
- ☐ **D** The node with the most children

4. Which of these are typical uses of rooted trees? [1 mark]

Tick every answer that is true.

- ☐ **A** A file system of folders and files
- ☐ **B** An expression tree for an arithmetic expression
- ☐ **C** A binary search tree for fast searching
- ☐ **D** A circular queue of print jobs

5. What is a binary tree? [1 mark]

- ☐ **A** A rooted tree in which each node has at most two children
- ☐ **B** A tree with exactly two leaves
- ☐ **C** A tree that stores only 0s and 1s
- ☐ **D** A graph with two roots

6. What does this program print? [1 mark]

```
CHILDREN = {"r": ["a", "b"], "a": ["c", "d", "e"], "b": ["f"]}
def leaves(node):
    kids = CHILDREN.get(node, [])
    if not kids:
        return 1
    return sum(leaves(k) for k in kids)
print(leaves("r"), leaves("a"))
```

The task: facts about a tree

`TREE` is a rooted tree stored as a dictionary: each key is a node that has children, and its value is the list of those children. Leaves appear only inside children lists, never as keys, and the keys are in no particular order, so the root is not necessarily first. Write a recursive function `height(node)` that returns the height of the subtree rooted at `node`, which is 0 for a leaf. Then print five lines: `root:` and the root; `nodes:` and the number of nodes; `edges:` and the number of edges; `leaves:` and every leaf in alphabetical order, separated by single spaces; and `height:` and the height of the whole tree. Work everything out from `TREE`.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

TREE = {
    "fetch": ["approach", "grip"],
    "mission": ["find", "fetch", "return"],
    "approach": ["align", "creep"],
    "find": ["scan", "turn"],
    "return": ["drive", "drop"],
}

def height(node):
    return 0
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a4-5-trees/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Print every node of `TREE` with its depth, the root first.
2. Add one node to `TREE` so that its height becomes 4, and check your program agrees.
3. Write `is_binary(tree)` that returns True when no node in a tree stored like `TREE` has more than two children.