M269 Prsntn25J TMA02

Notes

Phil Molyneux

19 January 2026

M269 Prsntn25J TMA02

Introduction

- **Preamble** outlines points on grading, disallowed code, software and the use of Generative AI
- **Grading** for this year is significantly different to previous years and you should note that the grading is banded so you have to get all the points in one band to get at least that band — pay particular attention to *ruff* messages
- **Disallowed Code** will generate *allowed* messages
- **Software** — you should create yourself a sheet of shortcut commands you use — I have created summaries for the previous version of *Jupyter Notebook* — some of the shortcuts have changed

  Menu Bar › Run › Run All Cells

  ⌘ + ⇧ + C (Mac) Ctrl + ⇧ + C (Win)
- **Part 1** is Qs 1, 2, 3
- **Part 2** is Qs 4, 5, 6

M269 Prsntn25J TMA02

Corrections

- **12 January 2026 Message about *allowed* Update for TMA02**
  - *allowed* update to 1.5.5
  - Close any M269 notebooks, browser tabs and terminals
  - Start a new Terminal
  - enter `m269-25j` to activate M269 software
  - Enter `allowed -V` — I find 1.5.4 on Monday, 26 January 2026
  - Enter `pip install --upgrade allowed`
- **26 January 2026 Fabienne Evans TMA02 Q1(b) Correction**
  - Tma02 Q1(b) should provide screenshot from QA4(c) not Q4(b)(ii)
  - Students to receive email

Ruff and *allowed*

Description

- Ruff is a Python linter
- Ruff rules and codes
- **ANN** Type annotations (flake8-annotations)
- **D** docstring (pydocstyle)
- **N** Naming (pep8-naming)
- **A** Shadowing (flake8-builtins)
- **E** Spaces, Tabs (pycodestyle Error)
- **F** Unused things for example, imports(Pyflakes)

Allowed

Usage

- **Description** dsa-ou.github.io/allowed/
- **Installation** dsa-ou.github.io/allowed/docs/installation.html
- **Usage** dsa-ou.github.io/allowed/docs/usage.html
- **Configuration** dsa-ou.github.io/allowed/docs/configuration.html
- **Location** ~/venvs > m268-25j > lib > python3.12 > site-packages > allowed

```
<~><1049> echo $HOME
/Users/molyneux
<~><1050> echo ~
/Users/molyneux

<~><1052> echo $USER
molyneux
```

M269 Prsntn25J TMA02

Q 1

- **Screenshots** use screenshot functionality or other software such as Snagit
- Save the screenshot as `fileName.jpg` or `fileName.png`
- Place the graphic file in the same folder as the Jupyter Notebook
- In the relevant cell insert:

```
![imageDesc](fileName.png (or JPG))
```

Question 2

Q 2(a)

- This asks for implementation of a function `power` which takes two integers as `base` and `exponent` and return $\texttt{base}^{\texttt{exponent}}$
- What *Ruff* error will the following produce ?

```
def power(base: int, exponent: int) -> int:
    """Compute the power of a base raised to a given exponent.

    Preconditions: base >= 0, exponent >= 0
    """
    result = 1
    for i in range(exponent) :
        result = result * base
    return result

# Tests are here
test_table = [
    ['Square', 2, 2, 4],
    ['Cube', 2, 3, 8],
    ["0 to n", 0, 3, 0],
    ["n to 0", 2, 0, 1],
    ["0 to 0", 0, 0, 1]
]
test(power, test_table)
```

Question 2

Q 2(a)

- **Output**

```
Testing power...
Tests finished: 5 passed (100%), 0
failed.
• 8: [B007] Loop control variable i not
used within loop body. Suggested fix:
Rename unused i to _i
```

Question 2

Q 2(b)

- A recursive version of power

```
def power_recursive(b, e) :
  if e == 0 :
    return 1
  else :
    return b * power_recursive(b, e - 1)
```

- Suggest 3 points where the code could be improved and why.

Question 2

Q 2(c)

- **Function**: find first occurrence
- **Inputs**: *items*, a tuple; *tuple list*, a list of tuples
- **Preconditions**: all tuples are the same length
- **Output**: *index*, an integer
- **Postconditions**: *index* is the smallest index of a tuple in *tuple list* which is comparably equal to *items*; if no match is found, index is -1

Question 2

Q 2(c)

```
def find_first_occurrence(items: tuple, tuple_list: list) -> int:
    """Find the first occurrence of items in list_items.

    Note that this is ignoring order of elements.

    Preconditions:
    - all tuples are the same length
    - all tuples' values are mutually comparable
    """
    sorted_items = sorted(items)
    for index in range(len(tuple_list)):
        if sorted(tuple_list[index]) == sorted_items:
            return index
    return -1
```

- State the worst-case complexity of this implementation. Justify your statement.
- You can assume that `sorted()` has worst-case complexity $\Theta(n \log n)$ where n is the length of the list being sorted.

Question 3

Q 3(a)

- Describe a real-life (non-programming) problem that may be solvable by **both** a divide-and-conquer **and** an alternative approach.
- Discuss how **each** of these approaches could be utilised.
- Explain why the **alternative** approach is **more** appropriate than the divide-and-conquer approach

Question 3

Q 3(b)

```
from ex import P # allowed
P([1,2,3])
```

```
integers = []
for integer in range(10):
    integers.append(integer)
    print(len(integers), "integers")
    %timeit -r 5 -n 10000 P(integers)
```

- You have to choose a reasonable range of numbers
- Experiment
- Read the documentation timeit

Question 3

Q 3(b)(ii)

- State (by name or in Big-Oh/Theta notation) and justify what you believe the complexity of the function *P* to be.

Binary Trees

Lowest Common Ancestor

- The **Lowest Common Ancestor** of two nodes k1 and k2 in a tree T is the lowest node that is an ancestor of both k1 and k2
- This is a famous problem — see Wikipedia: Lowest common ancestor
- These notes outline a recursive approach to implementing an algorithm
- Note that there are iterative approaches but they tend to be more complex or assume each node already has a pointer to its parent — you could write a program to add pointers to parents as a separate exercise

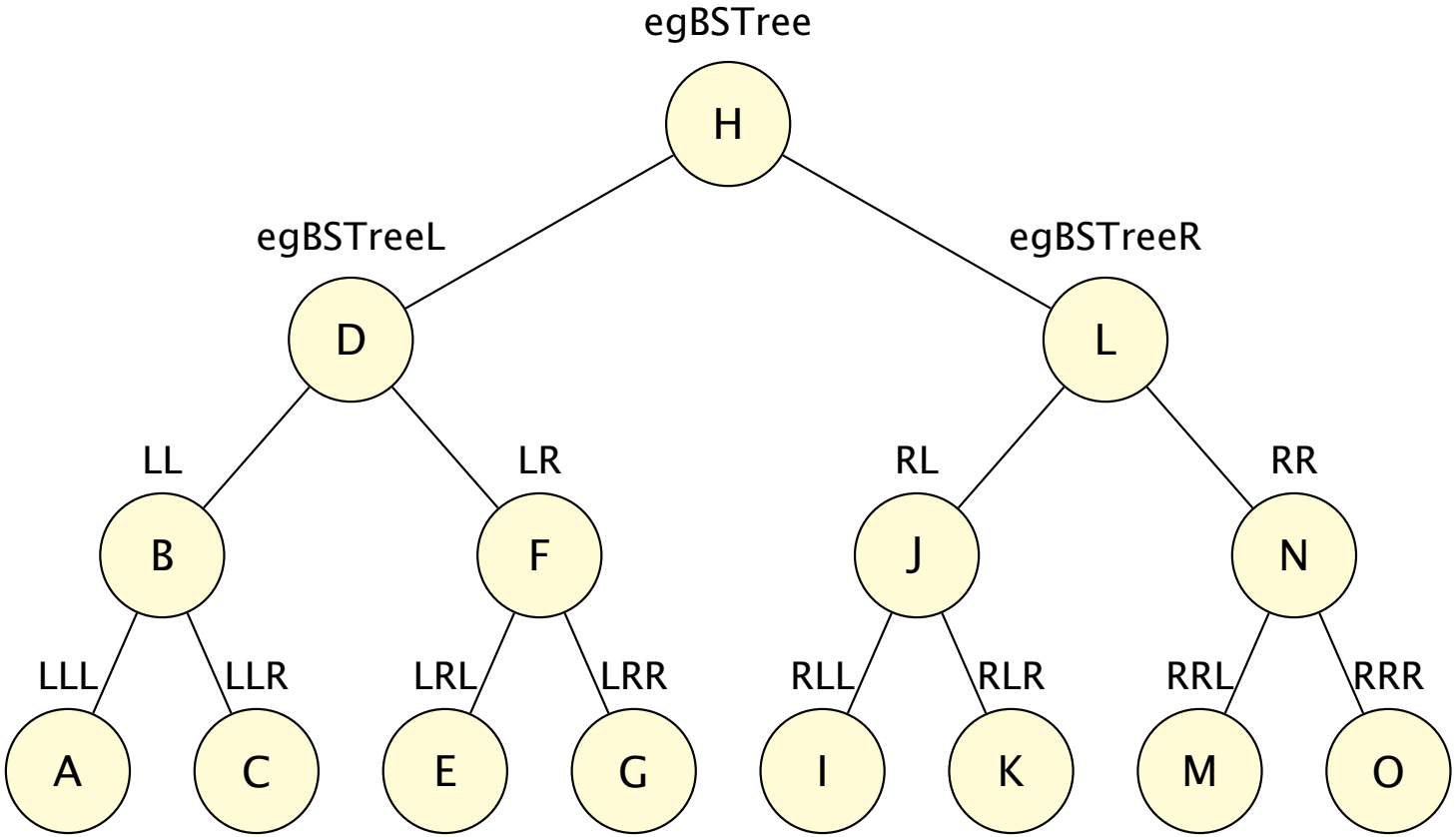
Binary Trees

Lowest Common Ancestor

- There are some subtle points about the *Lowest Common Ancestor* problem that catch students out.
- The two nodes have to be in the tree at the first call to the algorithm — otherwise you can get some incorrect answers
- A node has to be allowed to be regarded as an ancestor (or descendent) to itself otherwise the algorithm goes wrong
- The student has to include code for checking that a given node or key is in the tree (which could of itself be a question part)

Binary Trees

Example egBSTree



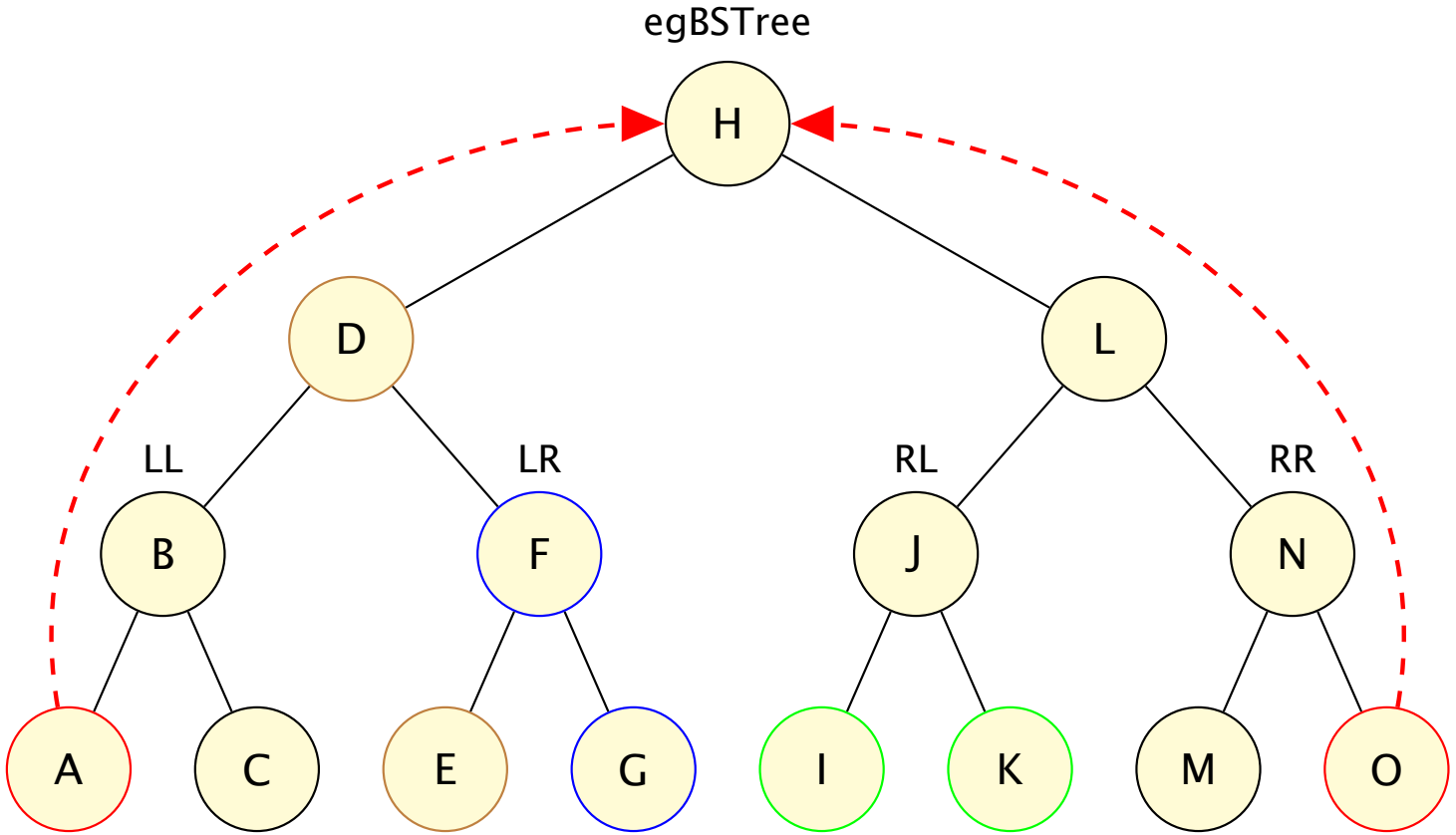
Binary Trees

LCA Questions

- **Questions**
- What should the *lowest_common_multiple* applied to egBSTree and the following arguments, return ?
- (egBSTree, 'A', 'O'), (egBSTree, 'I', 'K')
- (egBSTree, 'F', 'G'), (egBSTree, 'E', 'D')
- (egBSTree, 'A', 'X'), (egBSTree, 'X', 'Y')

Binary Trees

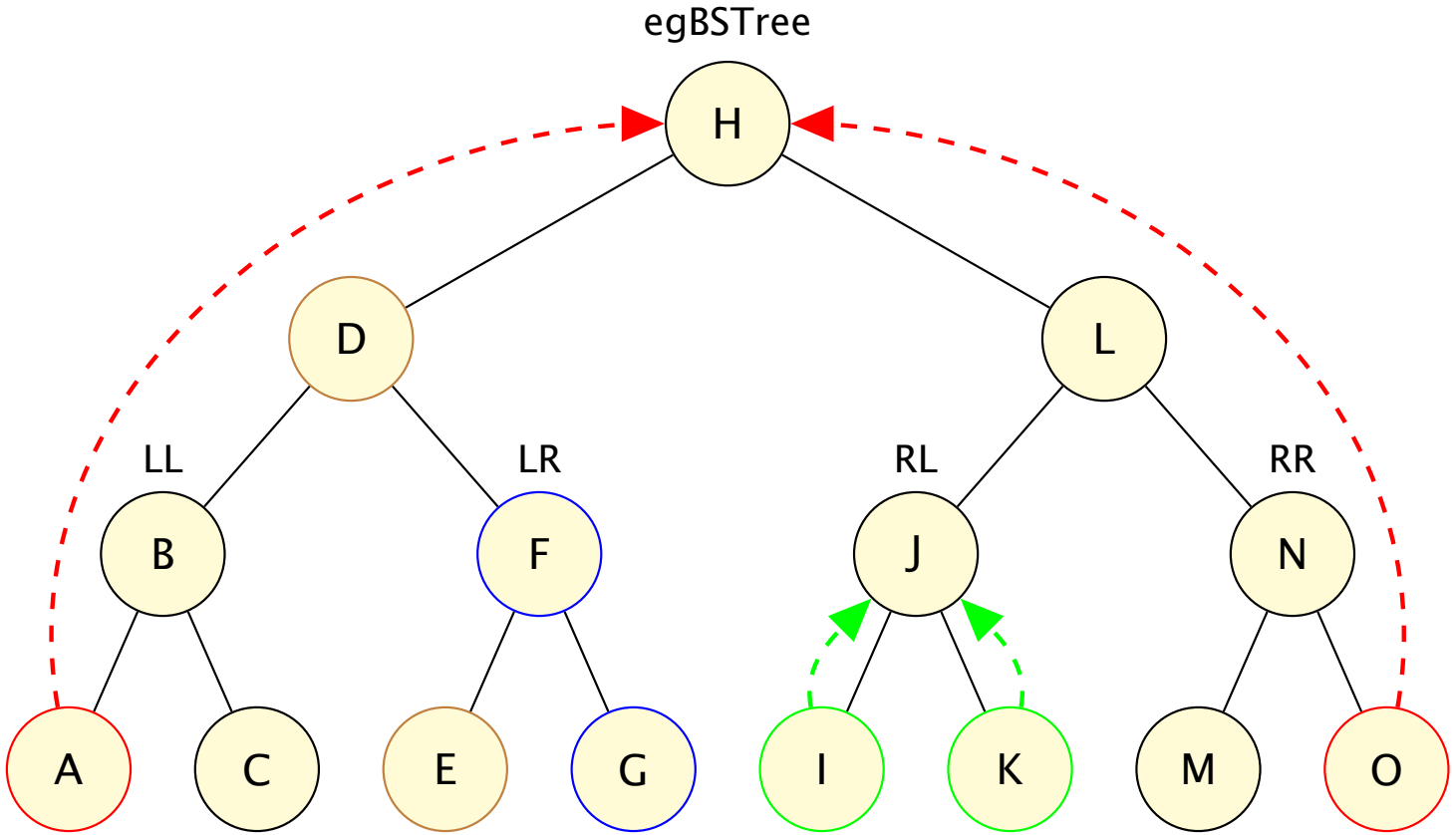
Examples of using `reportLCA()`



- `reportLCA(egBSTree, 'A', 'O')` different branches
- `reportLCA(egBSTree, 'I', 'K')` different branches

Binary Trees

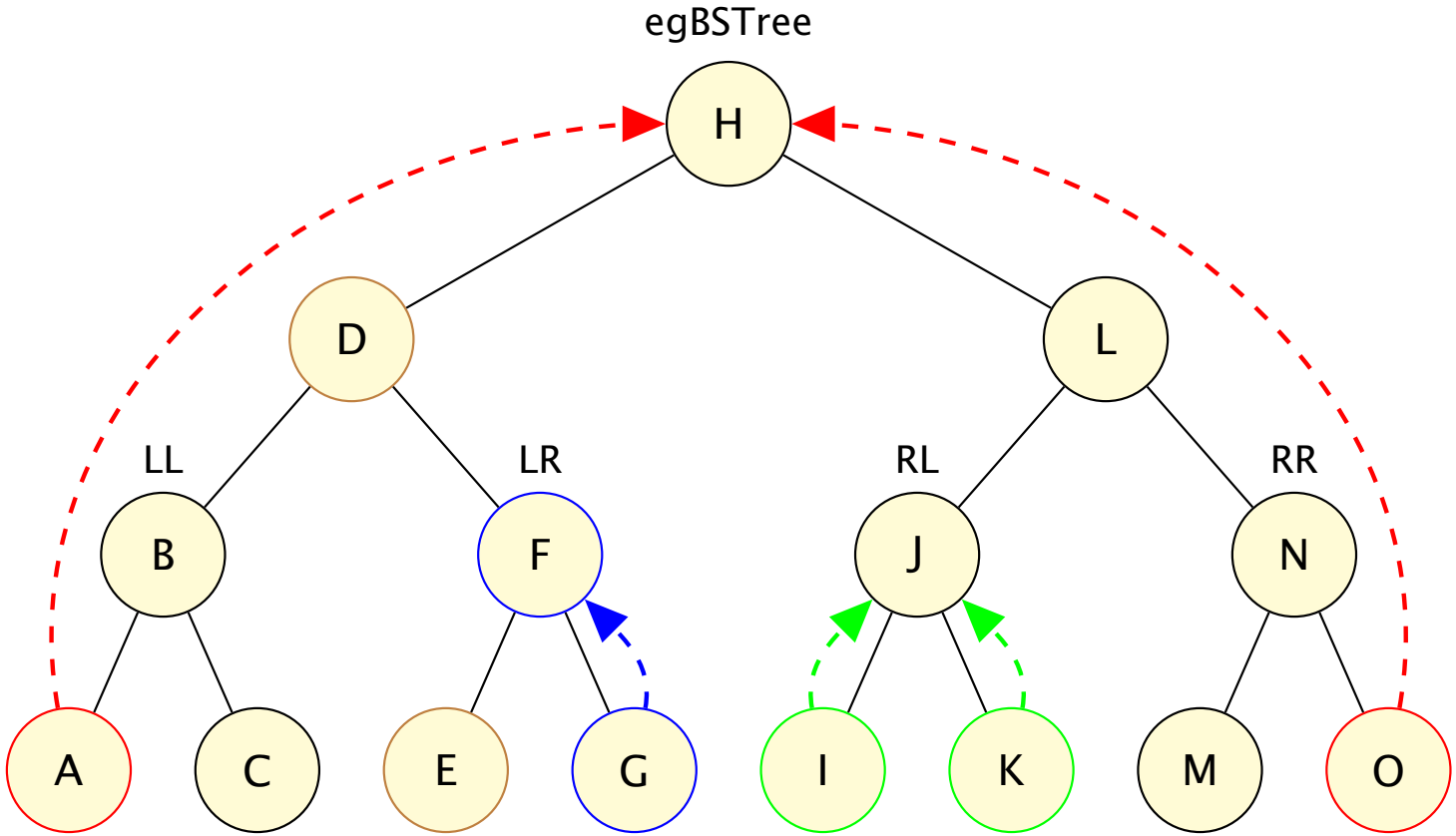
Examples of using `reportLCA()`



- `reportLCA(egBSTree, 'A', 'O')` different branches
- `reportLCA(egBSTree, 'I', 'K')` different branches
- `reportLCA(egBSTree, 'F', 'G')` same branch

Binary Trees

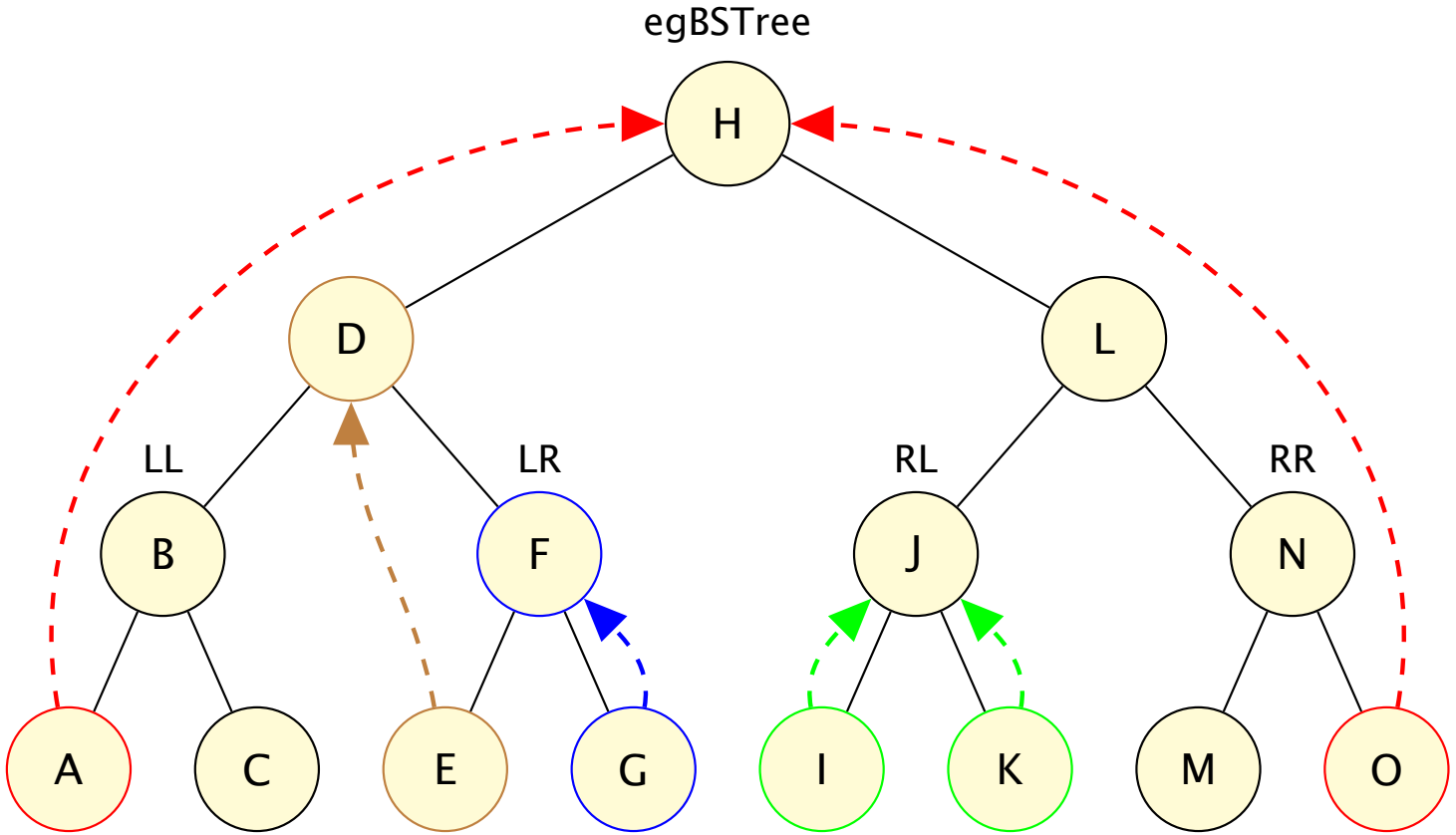
Examples of using `reportLCA()`



- `reportLCA(egBSTree, 'A', 'O')` different branches
- `reportLCA(egBSTree, 'I', 'K')` different branches
- `reportLCA(egBSTree, 'F', 'G')` same branch
- `reportLCA(egBSTree, 'E', 'D')` same branch

Binary Trees

Examples of using `reportLCA()`



- `reportLCA(egBSTree, 'A', 'O')` different branches
- `reportLCA(egBSTree, 'I', 'K')` different branches
- `reportLCA(egBSTree, 'F', 'G')` same branch
- `reportLCA(egBSTree, 'E', 'D')` same branch

Binary Trees

Abstract Data Type

- This example uses the ADT given below — this is different to the M269 representation but equivalent
- This emphasises that a Binary Tree is a union of an Empty Tree and the set of all non-empty nodes, with an item and two subtrees
- It also has the advantage of a default print representation that is useful
- The code is in file M269BinaryTrees2025LCA.py

```
7 from collections import namedtuple

9 EmptyBT = namedtuple('EmptyBT',[])

11 NodeBT = namedtuple('NodeBT'
12                     ,['dataBT','leftBT','rightBT'])

14 # Tree type

16 from typing import TypeVar,Union,NewType

18 T = TypeVar('T')
19 Tree = NewType('Tree',Union[EmptyBT,NodeBT])
```

Binary Trees

Abstract Data Type

- Here are the Binary Tree operations used in these notes

```
23 def mkEmptyBT() -> Tree :
24   return EmptyBT()

26 def mkNodeBT(x : T,t1 : Tree,t2 : Tree) -> Tree :
27   return NodeBT(x,t1,t2)

29 def isEmptyBT(t : Tree) -> bool:
30   return t == EmptyBT()

32 def getDataBT(t : Tree) -> T:
33   if isEmptyBT(t):
34     raise RuntimeError("getDataBT applied to EmptyBT()")
35   else:
36     return t.dataBT

38 def getLeftBT(t : Tree) -> Tree :
39   if isEmptyBT(t):
40     raise RuntimeError("getLeftBT applied to EmptyBT()")
41   else:
42     return t.leftBT

44 def getRightBT(t : Tree) -> Tree :
45   if isEmptyBT(t):
46     raise RuntimeError("getRightBT applied to EmptyBT()")
47   else:
48     return t.rightBT
```

Lowest Common Ancester

Development

- If `t` is empty then return an empty tree (representing fail)
- If one of the two input keys equals the key at `t` then return `t`
- Otherwise recurse with two further calls to `lca(getLeftBT(t), k1, k2)` and `lca(getRightBT(t), k1, k2)`

  Either the two nodes are in separate branches or the same branch
- Note that the following code cannot be used directly in any M269 exercise (since it uses different notation)

LCA

The LCA code

```
136 def lca(t  Tree, k1: T, k2: T) -> Tree :
137   if isEmptyBT(t) :
138     return mkEmptyBT()
139   elif (getDataBT(t) == k1 or getDataBT(t) == k2) :
140     return t
141   else :
142     leftLCA  = lca(getLeftBT(t),k1,k2)
143     rightLCA = lca(getRightBT(t),k1,k2)
144     if (not (isEmptyBT(leftLCA))
145         and not (isEmptyBT(rightLCA))) :
146       return t
147     else :
148       return (leftLCA if not (isEmptyBT(leftLCA))
149                       else rightLCA)
```

- Note this code cannot be used directly in any M269 exercise since it will be rejected

LCA

isInTree, reportLCA

- We initially have to check both nodes are in the tree and print some sensible string as a result

```
151 def isInTree(t: Tree, k: T) -> bool :
152   if isEmptyBT(t) :
153     return False
154   elif (getDataBT(t) == k) :
155     return True
156   else :
157     return (isInTree(getLeftBT(t), k)
158             or isInTree(getRightBT(t), k))


161 def reportLCA(t: Tree, k1: T, k2: T) -> str :
162   if (not isInTree(t, k1) or not isInTree(t, k2)) :
163     return ("Not␣both␣k1␣\'" + k1 + "\'␣and␣k2␣\'"
164             + k2 + "\'␣are␣in␣the␣tree")
165   else :
166     valueLCA = lca(t, k1, k2)
167     if isEmptyBT(valueLCA) :
168       return str(valueLCA)
169     else :
170       reportStr = ("Key␣value␣of␣LCA␣is␣\'"
171                    + str(getDataBT(valueLCA)) + "\'")
172       return reportStr
```

LCA

Future Work & References

- We should now work through the above examples to see what happens
- We should also indicate what happens if we run `lca` with one or both nodes not in the initial tree
- **LCA References**
- Wikipedia: Lowest common ancestor
- Lowest Common Ancestor in a Binary Tree
- Wikipedia: Schieber-Vishkin algorithm

Recursive Thinking

Example Expression Evaluation

- **Evaluate** showing which line in the code was used at each stage (or *reduction* step)
- `lca(egBSTree, 'A', 'O')`
- `lca(egBSTreeL, 'D', 'E')`
- `lca(egBSTreeL, 'X', 'Y')`
- `lca(egBSTree, 'A', 'X')`

Expression Evaluation

`lca(egBSTree, 'A', 'O')`

```
lca(egBSTree, 'A', 'O')
= lca(egBSTreeL,'A','O')                    # by line 141
  lca(egBSTreeR,'A','O')

lca(egBSTreeL,'A','O')
= lca(egBSTreeLL,'A','O')                   # by line 141
  lca(egBSTreeLR,'A','O')

lca(egBSTreeLL,'A','O')
= lca(egBSTreeLLL,'A','O')                  # by line 141
  lca(egBSTreeLLR,'A','O')

lca(egBSTreeLLL,'A','O')
= egBSTreeLLL                               # by line 139
```

```
lca(egBSTreeLR,'A','O')
  = mkEmptyBT()                             # by lines 141*3, 147*2
```

```
lca(egBSTreeR,'A','O')
= egBSTreeRRR                               # by similar reasoning
```

```
lca(egBSTree, 'A', 'O')
= egBSTree                                  # by line 144
```

Expression Evaluation

Sample Session

```
Python3>>> from M269BinaryTrees2025LCA import *
Python3>>> reportLCA(egBSTree, 'A','O')
"Key value of LCA is 'H'"
Python3>>> reportLCA(egBSTree, 'D','E')
"Key value of LCA is 'D'"
Python3>>> reportLCA(egBSTree, 'I','K')
"Key value of LCA is 'J'"
Python3>>> reportLCA(egBSTree, 'F','G')
"Key value of LCA is 'F'"
Python3>>> lca(egBSTree, 'Y','X')
EmptyBT()
Python3>>> lca(egBSTree, 'A','X')
NodeBT(dataBT='A', leftBT=EmptyBT(), rightBT=EmptyBT())
Python3>>> reportLCA(egBSTree,'Y','X')
"Not both k1 'Y' and k2 'X' are in the tree"
Python3>>> reportLCA(egBSTree,'A','X')
"Not both k1 'A' and k2 'X' are in the tree"
Python3>>>
```

- The above is why we need `isInTree(t,k)` and `reportLCA(t,k1,k2)`

Question 4

Q 4(a),(b),(c)

- **Q 4(a)** State base case
- **Q 4(b)** Recursion equations
- **Q 4(c)** Code

Question 5

Q 5(a),(b),(c),(d),(e),(f)

- **Various Type of Graphs**
- Path Graph
- Neither Path nor Cycle Graph
- Cycle Graph
- Complete Graph
- Not a special Graph
- Null Graph

Question 6

Q 6(a)

- Implement a **Greedy** algorithm to find a (possible) maximum of the treasure to be collect
- Note that greedy algorithms hardly ever work so we *should* be obliged to produce a proof in this case but we are not asked for this (and the course has not done much on proofs)
- We should be prepared to write some extra code to investigate the data

Question 6

Q 6(a)

- Provided code for Trove

```
class Trove:
    """A treasure trove."""

    def __init__(self, trove_name: str, weight: int, value: int,
                 number_of_chests: int) -> None:
        """Create the treasure trove."""
        self.trove_name = trove_name
        self.weight = weight
        self.value = value
        self.number_of_chests = number_of_chests
        self.chest_weight = weight /  number_of_chests
        self.chest_value = value / number_of_chests
```

Question 6

Q 6(a)

- Provided code for `Plan`

```
class Plan:
    """A plan for getting the most treasure.

    - troves should be a list of the names of the troves the pirate will visit.
    - total_value should be the total value taken from all troves.
    - Note: a float is a number with a decimal point. You will not
      need to format this number in any way and it can be used just
      like an int. We round it down in the initialisation.
    - Note: we have added __str__ and __eq__ methods here which you do not
      need to understand; they support testing.
    """

    def __init__(self, troves: list, total_value: float) -> None:
        """Create a plan for the pirate to follow."""
        self.troves = troves
        self.total_value = round(total_value) # allowed

    def __str__(self) -> str:
        """Return a human readable output."""
        return "Troves: "+", ".join(self.troves)+" Value:"+str(self.total_value)

    def __eq__(self, other: dict) -> bool:
        """Allow equality comparison."""
        return self.__dict__ == other.__dict__ # allowed
```

Question 6

Q 6(a)

- Provided code for `treasure_map`

```
treasure_map = [
    Trove("Black Powder Bay", 61, 1832, 7),
    Trove("Rum Runner's Cove", 77, 1851, 10),
    Trove("Dead Man's Lagoon", 23, 2560, 2),
    Trove("Cutlass Reef",96, 607, 8),
    Trove("the Siren's Snare", 46, 793, 5),
    Trove("Gold Tooth Isle", 89, 2577, 2)
]
```

- `__dict__` A dictionary or other mapping object used to store an object's (writable) attributes. Not all instances have a `__dict__` attribute; see the section on `__slots__` for more details.

Question 6

Q 6(a)

- Code to see the ordering of the `treasure_map` (`trove_list`) before and afte `greedy_trove_selection`

```
def treasMapReview(treasMap: list) -> list :
  revList = [(tv.trove_name,
              tv.chest_weight,tv.chest_value,tv.number_of_chests,
              tv.chest_value/tv.chest_weight)
              for tv in treasMap]
  return revList

treasMpRevInitial01 = treasMapReview(treasure_map)
```

- Python Tutorial: 7, Input and Output
- Python Tutorial: 7.1.1 Formatted String Literals
- Python Tutorial: 7.1.2 The String format() Method
- `str.format()`
- Format String Syntax

Question 6

Q 6(b)

- **Q 6(b)(i)** Describe criteria for greedy choice — highest ratio of value to weight — where in the code ?
- **Q 6(b)(ii)** Compatibility — total weight must not exceed maximum weight — where in the code ?