Computational Thinking and Program Design

Assignment Twelve (20 marks)
(Due on 31 May 2022)

Rocky K. C. Chang

26 May 2022

Instructions:

1. Submit your code to the iLearning platform before 23:59 on 31 May.
2. Name your program files for Q1 as `Q1.py`, Q2 as `Q2.py`, and Q3 as `Q3.py`.
3. Compress `Q1.py`—`Q3.py` and `myqueue.py` into a .7z file (using 7-zip). The file name should be "Your student ID".7z.
4. Submit the .7z file to the iLearning platform.
5. Late submission will not be accepted.
6. Observe also the penalty for plagiarism as stated in the Course Overview slides.

Question 1 (From BFS search to a forward BFS tree)

[5 MARKS] As discussed yesterday, we could easily turn the BFS search results (a sequence data type) into a BFS tree. Moreover, we introduced two types of implementations: forward BFS tree and backward BFS tree. Figure 1 shows them for our 10-node graph.

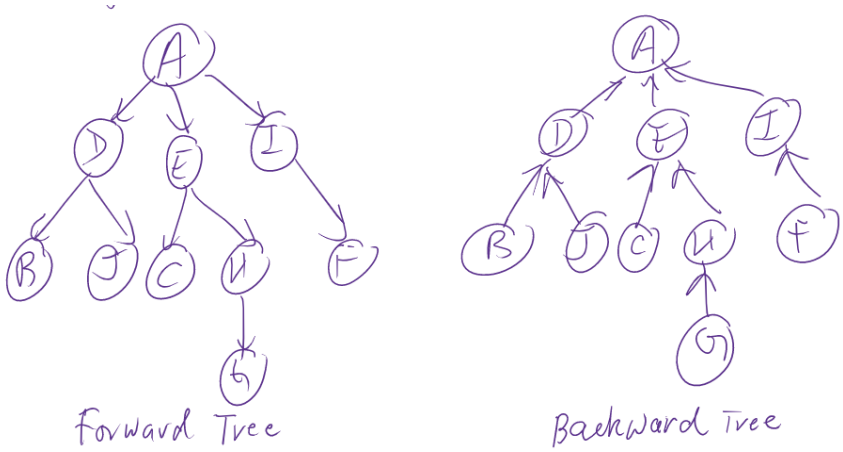

Figure 1: Two types of trees for our 10-node graph.

In this question, you are asked to implement a function that will generate a forward BFS tree from a graph and a starting node. You will implement the tree using Python dictionary, in which the keys are the nodes in the graph, and the values are the sets of children nodes. Please pay attention to the requirements below. No marks will be given if you violate them.

1. You must use the attached `A11_1_Q1_queue.py` to start with.
2. You will remove only the line "`print(s, end=" ")`" and nothing else.
3. Add additional lines to implement the function.

```
def forwardBfsTree(graph, node):
# Input: A graph and a starting node for the BFS search
# Output: Return a forward BFS tree
```

Please include the test code, the 10-node graph and the MCGW graph below for testing. The expected results are given in Figure 2.

```
tree = forwardBfsTree(graph, 'A')
for node in tree.items():
  print(node)
print()
tree = forwardBfsTree(MCGW_graph, 'EEEE')
for node in tree.items():
  print(node)

graph = {
  'A' : ['D', 'E', 'I'],
  'B' : ['D'],
  'C' : ['E', 'H', 'J'],
  'D' : ['A', 'B', 'E', 'J'],
  'E' : ['A', 'C', 'D', 'H'],
  'F' : ['H', 'I'],
  'G' : ['H'],
  'H' : ['C', 'E', 'F', 'G'],
  'I' : ['A', 'F'],
  'J' : ['C', 'D']
}

MCGW_graph ={
  'EEEE':['WEWE'],
  'WEWE':['EEEE','EEWE'],
  'EEWE':['WEWE','WWWE','WEWW'],
  'WWWE':['EEWE','EWEE'],
  'EWEE':['WWWE','WWEW'],
  'WEWW':['EEWE','EEEW'],
  'EEEW':['WEWW','WWEW'],
  'WWEW':['EWEE','EEEW','EWEW'],
  'EWEW':['WWEW','WWWW'],
  'WWWW':['EWEW']
}
```

```
('A', {'E', 'I', 'D'})
('D', {'B', 'J'})
('E', {'C', 'H'})
('I', {'F'})
('B', set())
('J', set())
('C', set())
('H', {'G'})
('F', set())
('G', set())

('EEEE', {'WEWE'})
('WEWE', {'EEWE'})
('EEWE', {'WEWW', 'WWWE'})
('WWWE', {'EWEE'})
('WEWW', {'EEEW'})
('EWEE', {'WWEW'})
('EEEW', set())
('WWEW', {'EWEW'})
('EWEW', {'WWWW'})
('WWWW', set())
```

Figure 2: Expected results for Q1.

Question 2 (From BFS search to a backward BFS tree)

[5 MARKS] In this question you will implement a function to generate a backward BFS tree from a graph and a starting node.

```
def backwardBfsTree(graph, node):
# Input: A graph and a starting node for the BFS search
# Output: Return a backward BFS tree
```

The same requirements for Q1 apply here. Please also include the two Python dictionaries for the 10-node graph and MCGW graph and the testing code below. The expected results are given in Figure 3.

```
tree = backwardBfsTree(graph, 'A')
for node in tree.items():
  print(node)
print()
tree = backwardBfsTree(MCGW_graph, 'EEEE')
for node in tree.items():
  print(node)
```

```
('A', 'A')
('D', 'A')
('E', 'A')
('I', 'A')
('B', 'D')
('J', 'D')
('C', 'E')
('H', 'E')
('F', 'I')
('G', 'H')

('EEEE', 'EEEE')
('WEWE', 'EEEE')
('EEWE', 'WEWE')
('WWWE', 'EEWE')
('WEWW', 'EEWE')
('EWEE', 'WWWE')
('EEEW', 'WEWW')
('WWEW', 'EWEE')
('EWEW', 'WWEW')
('WWWW', 'EWEW')
```

Figure 3: Expected results for Q2.

Question 3 (Finding leaf nodes of BFS trees)

[10 MARKS] In this question you will implement two functions (for forward BFS tree and backward BFS tree) for finding the leaf nodes of a BFS tree. A leaf node is one that has no children.

```
def findLeafNodesfromForwardTree(tree):
    """
    Input: A forward BFS tree
    Output: Return a set of leaf nodes in the BFS tree
    """

def findLeafNodesfromBackwardTree(tree):
    """
    Input: A backward BFS tree
    Output: Return a set of leaf nodes in the BFS tree
    """
```

Include the two Python dictionaries for the 10-node graph and MCGW graph, the code below, `forwardBfsTree()` and `backwardBfsTree()` to test your functions. The expected results are given in Figure 4.

```
print("Forward tree:")
print(findLeafNodesfromForwardTree(forwardBfsTree(graph,"A")))
print(findLeafNodesfromForwardTree(forwardBfsTree(MCGW_graph,"EEEE")))
print()
print("Backward tree:")
print(findLeafNodesfromBackwardTree(backwardBfsTree(graph,"A")))
print(findLeafNodesfromBackwardTree(backwardBfsTree(MCGW_graph,"EEEE")))
```

```
Forward tree:
{'J', 'C', 'F', 'G', 'B'}
{'WWWW', 'EEEW'}

Backward tree:
{'J', 'C', 'F', 'G', 'B'}
{'WWWW', 'EEEW'}
```

Figure 4: Expected results for Q3.