
CS 161: Fundamentals of Artificial Intelligence

Spring 2023– Assignment 2 – Due 11:59pm, Tuesday, April 18

- Submit your commented Python program in a file named **hw2.py** via **BRUINLEARN**.
- Base your solution on the skeleton file named **hw2-skeleton.py**.
- Your programs should be written in good style. In Python, a comment is any character following a hash character (`#`) on a line. Provide an overall comment explaining your solutions. Furthermore, every function should have a header comment explaining precisely what its arguments are, and what value it returns in terms of its arguments. In addition, you should use meaningful variable names.
- The physical layout of the code on the page is very important for making python programs readable. Make sure that you use blank lines between functions and indent properly. Programming style will be a consideration in grading the assignment.
- You are restricted to using the python built-in functions. All other packages made by `import` (like `numpy queue`, and `collections`) are forbidden.
- You may assume that all input to your functions is legal; i.e. you do not need to validate inputs.
- Do not write any additional helper functions for your code unless this is explicitly allowed.
- Your function declarations should look **exactly as specified** in this assignment. Make sure the functions are spelled correctly, take the correct number of arguments, and those arguments are in the correct order.
- Even if you are not able to implement working versions of these functions, please **include a correct skeleton** of each. Some of these assignments are auto graded and having missing functions is problematic.
- By submitting this homework, you agree to the following honor code.

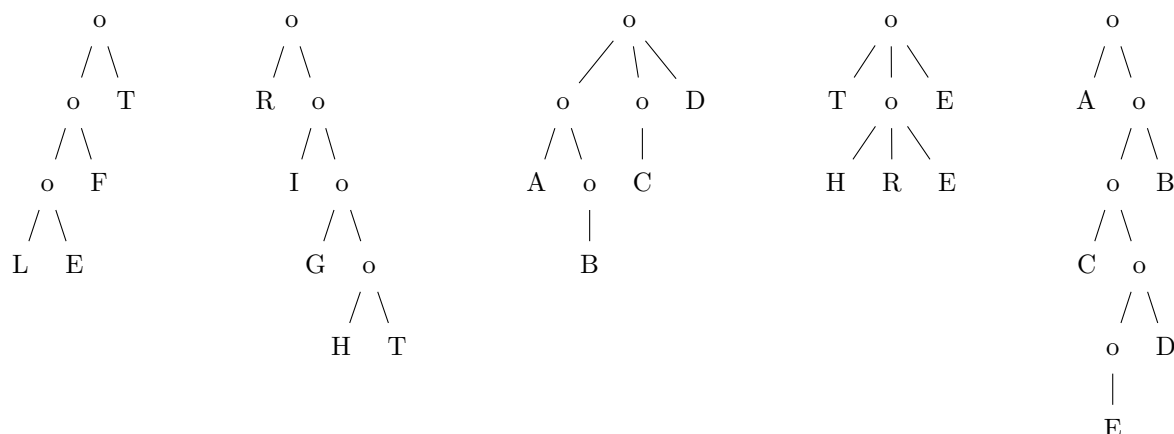
You are encouraged to work on your own in this class. If you get stuck, you may discuss the problem with up to two other students, **PROVIDED THAT YOU SUBMIT THEIR NAMES ALONG WITH YOUR ASSIGNMENT. ALL SOLUTIONS MUST BE WRITTEN UP INDEPENDENTLY, HOWEVER.** This means that you should never see another student's solution before submitting your own. You may always discuss any problem with me or the TAs. **YOU MAY NOT USE OLD SOLUTION SETS UNDER ANY CIRCUMSTANCES.** Making your solutions available to other students, **EVEN INADVERTENTLY** (e.g., by keeping backups on github), is aiding academic fraud, and will be treated as a violation of this honor code.

You are expected to subscribe to the highest standards of academic honesty. This means that every idea that is not your own must be explicitly credited to its author. Failure to do this constitutes plagiarism. Plagiarism includes using ideas, code, data, text, or analyses from any other students or individuals, or any sources other than the course notes, without crediting these sources by name. Any verbatim text that comes from another source must appear in quotes with the reference or citation immediately following. Academic dishonesty will not be tolerated in this class. Any student suspected of academic dishonesty will be reported to the Dean of Students. A typical penalty for a first plagiarism offense is suspension for one quarter. A second offense usually results in dismissal from the University of California.

1. (20pts) A search tree can be represented as *tuples* in Python as follows:

- (a) if the search tree contains a single leaf node L , it can be represented by a non-tuple object L
- (b) if the search tree has more than one node and is rooted at N , then it can be represented by a tuple (S_1, S_2, \dots, S_k) where S_i represents the i th subtree of N .

Consider for example the following search trees, whose Python representations are shown later.



Write a single Python function, called **BFS**, that takes a single argument **TREE** that represents a tree, and returns a tuple of leaf nodes in the order they are visited by **left-to-right** breadth-first search. Test your program on at least these inputs:

```
>>> BFS("ROOT")
('ROOT',)
>>> BFS((((("L", "E"), "F"), "T")))
('T', 'F', 'L', 'E')
>>> BFS(("R", ("I", ("G", ("H", "T")))))
('R', 'I', 'G', 'H', 'T')
>>> BFS(("A", ("B",)), ("C",), "D"))
('D', 'A', 'C', 'B')
>>> BFS(("T", ("H", "R", "E"), "E"))
('T', 'E', 'H', 'R', 'E')
>>> BFS(("A", (("C", (("E",), "D")), "B")))
('A', 'B', 'C', 'D', 'E')
```

2. (20pts) Write a single Python function, called **DFS**, that takes a single argument **TREE** that represents a tree, and returns a tuple of leaf nodes in the order they are visited by **left-to-right** depth-first search. Test your program on at least these inputs:

```
>>> DFS("ROOT")
('ROOT',)
>>> DFS((((("L", "E"), "F"), "T")))
('L', 'E', 'F', 'T')
>>> DFS(("R", ("I", ("G", ("H", "T")))))
('R', 'I', 'G', 'H', 'T')
>>> DFS(("A", ("B",)), ("C",), "D"))
('A', 'B', 'C', 'D')
```

```

('A', 'B', 'C', 'D')
>>> DFS(("T", ("H", "R", "E"), "E"))
('T', 'H', 'R', 'E', 'E')
>>> DFS(("A", (("C", (("E",), "D")), "B")))
('A', 'C', 'E', 'D', 'B')

```

3. (20pts) Write a **set** of Python functions that implement depth-first iterative-deepening search. The top-level function, called **DFID**, takes two arguments, a tuple **TREE** representing a tree and an integer **D** representing the depth of **TREE**, and returns a tuple of leaf nodes in the order they are visited by a **right-to-left** depth-first iterative-deepening search. Note that those nodes that are visited in multiple iterations will appear multiple times in the returned tuple. Test your program on at least these inputs:

```

>>> DFID("ROOT", 0)
('ROOT',)
>>> DFID((((("L", "E"), "F"), "T"), 3)
('T', 'T', 'F', 'T', 'F', 'E', 'L')
>>> DFID(("R", ("I", ("G", ("H", "T")))), 4)
('R', 'I', 'R', 'G', 'I', 'R', 'T', 'H', 'G', 'I', 'R')
>>> DFID((((("A", ("B",)), ("C",), "D")), 3)
('D', 'D', 'C', 'A', 'D', 'C', 'B', 'A')
>>> DFID(("T", ("H", "R", "E"), "E"), 2)
('E', 'T', 'E', 'E', 'R', 'H', 'T')
>>> DFID(("A", (("C", (("E",), "D")), "B")), 5)
('A', 'B', 'A', 'B', 'C', 'A', 'B', 'D', 'C', 'A', 'B', 'D', 'E', 'C', 'A')

```

4. (40pts) This question aims to solve a problem stated by Homer Simpson in “*Gone Maggie Gone*”, The Simpsons, Season 20, Episode 13. Homer wants to cross a river with his baby (Maggie), his dog (Santa’s Little Helper), and a jar of poison. Homer is the only one who can operate the boat, and he can take at most one thing with him. The dog cannot be left alone with the baby, because he tries to bite her. The baby cannot be left alone with the poison, because she tries to eat it. The goal is to get everybody on the other side of the river.

The file **hw2-skeleton.py** contains a framework for solving this puzzle using depth-first search. Implement the functions in it as described in the comments. **DO NOT CHANGE THE FUNCTION NAMES OR PARAMETERS. DO NOT WRITE ANY ADDITIONAL FUNCTIONS.**