Homework #2[1]

Due by 11:59pm 10/10/2018

Overview

At the heart of a two-player adversarial game is the Minimax Algorithm. It tells us what the current (max) player's best move is. Additionally, the alpha-beta pruning modification makes the overall algorithm more efficient.

This assignment asks you to demonstrate your understanding of both the basic minimax algorithm and alpha-beta pruning. First you need to manually trace through how minimax with alpha beta pruning processes some abstract game trees (similar to those we've done in class). Then, you need to cross-check your answers with a program you write.

Program

You have to write a program called **minimax_a_b.py** that takes an input file as its argument. The input file contains a complete "game tree" (the format is described below). Your program will have to process the tree twice -- once with a vanilla minimax algorithm, and once using minimax with alpha beta pruning (in both cases, it is assumed that the tree is traversed in a *left-to-right* order). For each case, print final utility value propagated to the root node as well as the names of all the nodes visited.

Input File Format

The input tree is represented as a list where the zeroth element gives the name of the parent and the rest of the list elements are that node's children. The leaf nodes of the tree will have both a name and a utility value; a leaf node is represented as a tuple ('name', value). For example, a simple tree with root node named A that has two terminal children nodes named B (with a value of 5) and C (with a value of 10) would be represented as:

```
['A', ('B', 5), ('C',10)]
```

You may import the ast library and use the "literal_eval" function to turn a string into a nested list structure.

Test cases

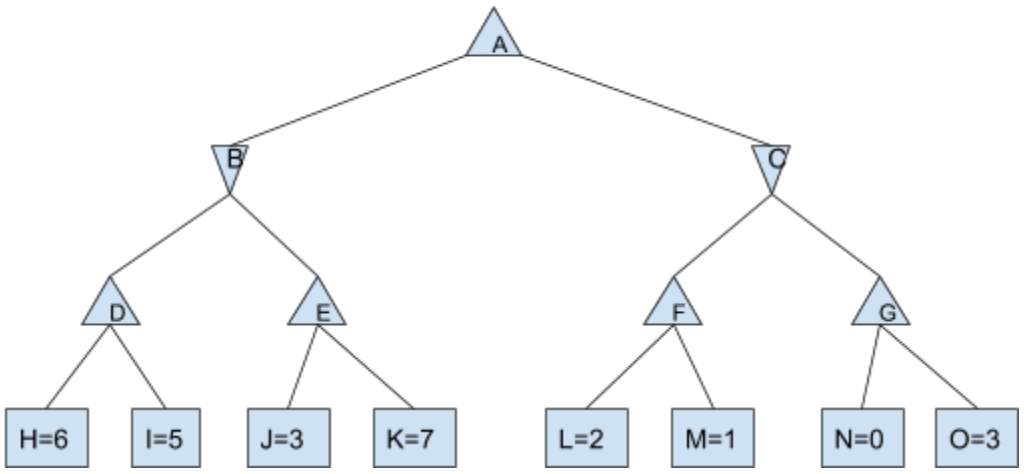
There are three game trees for you to both manually trace through and to give as input for your program. (Note that when we grade your program, we may try an input file with a bigger tree).

---

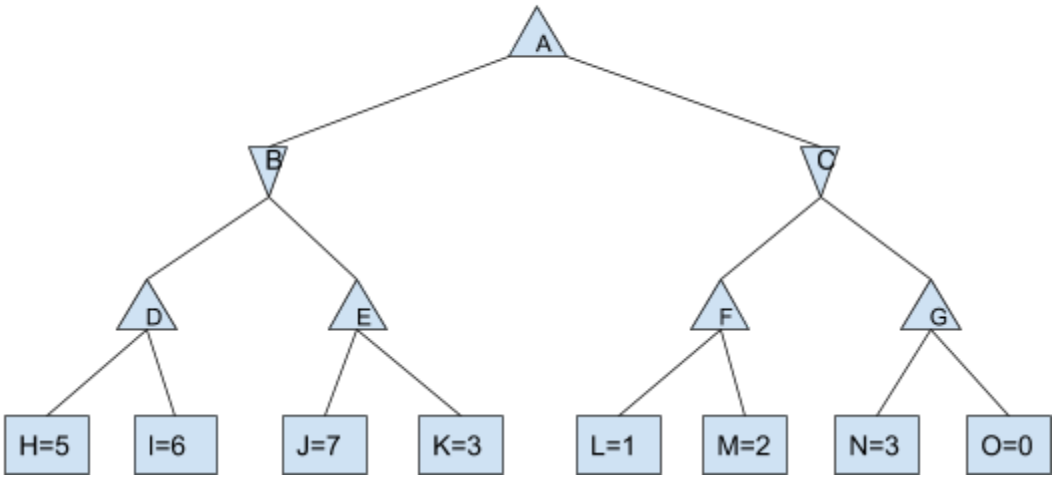
[1] Shared Google directory (all files are initially committed in github repository)

The input files are named test_treeN and are provided in this repository; they are also drawn below:

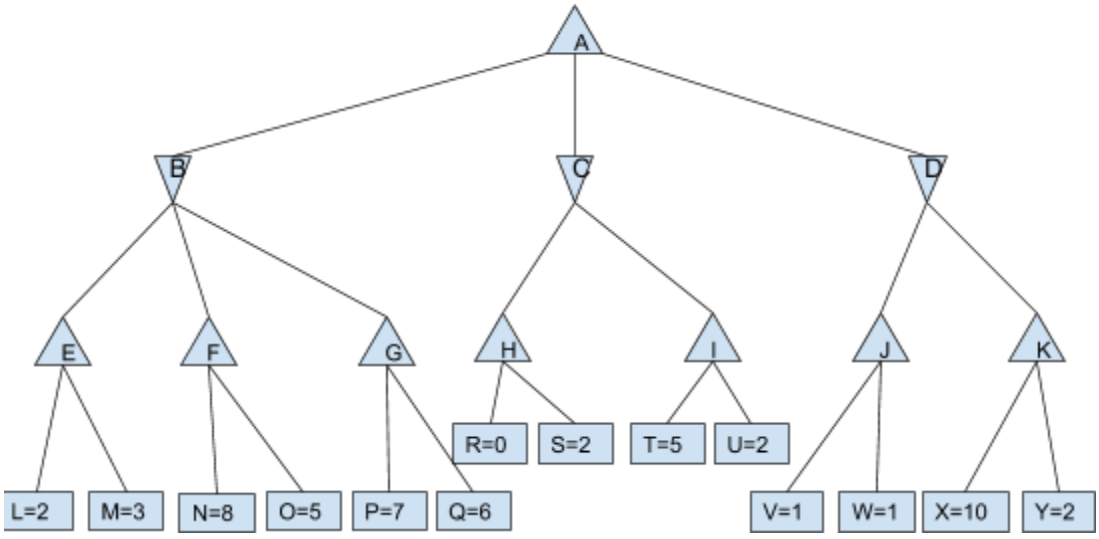
Tree 1



Tree 2



Tree 3



What to submit

- **manual solution:** Show your manual trace of minimax with alpha-beta pruning for the three sample trees. For each tree, show how the alpha and beta values are updated; cross out nodes that are pruned; give the final utility value and which action was taken.
- **minimax_a_b.py**: your source code for minimax with alpha-beta pruning
- **transcript**: show your program's output for the supplied test case files.
- **readme**: Specify the python version you are using. Document anything that's not working, or any other information that you think the grader would need to know in order to test your program.

Grading Guideline

Assignments are graded qualitatively on a non-linear five point scale. Below is a rough guideline:

- 5 (100%): Manual solutions for the three test cases are correct. Both the basic minimax and the alpha-beta pruning version work as expected. There is a clear enough README for the grader and a transcript.
- 4 (93%): Small mistakes in the manual solutions; OR both the basic minimax and the alpha-beta pruning version seem to work, but we found some subtle or minor bugs. There is a clear enough README for the grader and a transcript.
- 3 (80%): Some bigger mistakes in the manual solutions; OR only the basic minimax is working. There is a clear enough README for the grader and a transcript.

- 2 (60%): No manual traces of the three test cases, but the program works. OR the program isn't working, but the manual traces are largely correct.
- 1 (40%): Alpha-beta pruning is not correct (program and manual traces), but manual traces of the basic minimax is correct.