Tutorial Problem Set #3

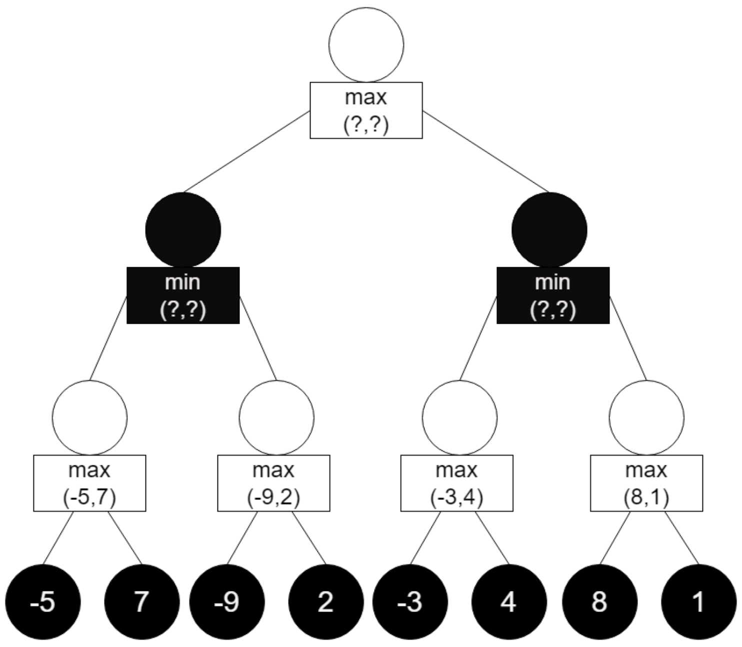
Due: Wednesday, October 2, 2024, 11:59 PM

Policy

- Piazza questions on tutorial problems will be ignored or deleted. Questions will only be answered in your assigned tutorial section.
- Sample executables can be found in your 1249 git repository directory (run `git pull`).
- Completing the problem set will reduce the weight of the final exam by 0.5%. To complete a problem set, you must pass at least 50% of the public and secret tests.
- You may assume all input is valid. Tutorial problems **NEVER** require checking for invalid inputs.

Question 1

When writing a program that can play a turn-based game (e.g., chess), we must write a search algorithm that allows the program to look ahead at possible positions before deciding what move to make. An alternative is implementing the Minimax algorithm. It is a recursive algorithm that evaluates the values associated with each move or position of the game, indicating how good such a position would be for a player. The algorithm explores the nodes of a binary tree holding integer numbers; large values favour player A, while smaller values favour player B. The leaf nodes represent the end states of the game, and each level of the tree represents a turn taken by one of the players. Player A attempts to maximize the result of the move evaluation, while Player B tries to minimize it.



In this exercise, you must debug and fix a faulty implementation of the Minimax algorithm. The provided program ends with a `Segmentation fault`. Use gdb to investigate the problem and test your solution. During the exercise, you will be asked to inspect specific information related to the execution of the code. Create a file called `gdbOutput.txt` in which you are supposed to write your answers to the questions listed below. You should write one answer per line with no extra space. Please write the names of files and functions as you see in gdb. Do not capitalize lowercase letters. You should not need to change the code in the files `main.cc` or `minimax.h`.

1. In which file and line does gdb stop the execution saying that the `Segmentation fault` signal was received? (Write your response as it appears in gdb `filename:linenumber`)
2. In which function does the `Segmentation fault` occur? (Write the function's name as output by gdb; do not include parentheses or capital letters.)
3. Before investigating the faulty control flow, inspect the values in the tree the faulty function uses. Place a breakpoint at the function where the `Segmentation fault` occurs and rerun the program. Each time the execution is paused, inspect the tree node passed to the function to figure out where in the tree we are. How many nodes can we traverse without running into any errors or `nullptrs`? (Use numbers to answer this question.)

The algorithm should compute a tree like the one shown above. If you find any inconsistency with the size or content of the tree, try debugging the Node constructor to fix those issues. Remember to recompile the code if you introduce any changes. Once you know the tree is correct, try running the program with gdb again.

4. When gdb pauses the execution because of the `Segmentation fault`, inspect the control flow backtrace at that point in the program. What are the unique names of all functions executed before the Segmentation fault? List them on a single line, starting from `main`.

The `Segmentation fault` seems to be caused by an infinite recursion within the code. Inspect the functions causing the infinite recursion and try to modify the code to fix the problem. Remember that the result of max and min evaluations should only be assigned to values of internal nodes of the tree. When leaf nodes are visited, their current value should be returned. After fixing the segmentation fault, run the program with gdb again. Your program should print `4` as the value of the tree root.

5. Inspect the values of all the tree internal nodes and add them to your text file (no need to include the leaf values). Distribute the node values so that numbers in the same tree level are in the same line, separated by whitespace. For example:

   7

   7 3

   9 7 3 2

Submission

Submit a zip file called `tutorial3.zip` to Marmoset with your `gdbOutput.txt` and `minimax.cc`.