McMaster University
MECHTRON 2MD3: Data Structures and Algorithms for Mechatronics

Winter 2023, Assignment 5

Instructions
Questions annotated "short answer" should be typed and submitted in a single pdf document with the file name "1234-asg5.pdf" where 1234 is your student ID. For questions with programming, submit C++ source files as per the question instructions. All code in programs and short answers should be formatted and commented. Please ensure your source files compile and run properly before submitting.

Questions 1 [programming + short answer, 6 marks]
Check your assignment 4 solution for memory leaks using valgrind on the pascal server. If you find leaks, debug you code until valgrind reports no leaks. You should follow the "rule of three" to make sure your LinkedBinaryTree class has working destructor, copy constructor, and overloaded copy assignment operator. Note that code supplied for question 2 below has a leak-free LinkedBinaryTree class. You may use this as a reference but please work to modify your own code rather than copying the class directly. You should submit your debugged code as "1234-asg4-debugged.cpp" where 1234 is your student ID. Please also write a brief description (1-3 sentences) about what was wrong, if anything, and how you fixed it. If you were not able to fix it, discuss where you got stuck.

Questions 2 [programming + short answer, 30 marks]
Note that this project will be discussed in detail during the lecture on March 23, 2023.

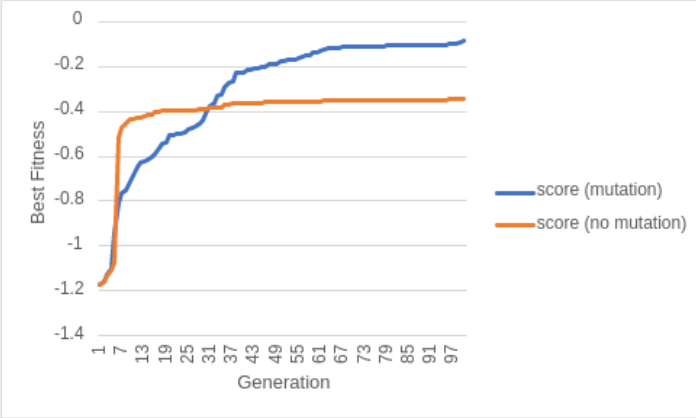
Download (or pull) the code directory from:
https://gitlab.cas.mcmaster.ca/kellys32/2md3_2023/-/tree/main/assignment_5

This directory contains a near-complete implementation of a Genetic Programming (GP) experiment to solve the Cart Centering (isotropic rocket) problem from lecture 2MD3_2023_TreeGP-2023-03-09. The genetic_programming_01.cpp program will evolve trees for 100 generations and print statistics on the best tree at each generation. Your task involves 3 parts as follows;

Part 1. Implement the following two tree mutation operators:

`LinkedBinaryTree::deleteSubtreeMutator(mt19937& rng)`
This function should delete a randomly selected subtree from the tree.
`LinkedBinaryTree::addSubtreeMutator(mt19937& rng)`
This function should add a randomly created subtree to the tree.

Place holders for these functions already exist. Your task is to fill in their code. You may reuse any code existing in the class (e.g. the randomExpressionTree function). You should see a significant improvement in the best fitness after implementing the mutation operators. When plotted, the data should look like something like this (red line is the original code without mutation):



Plot your results in the above format before and after implementing your mutation operators. I used Exel but any plotting tool is fine). Your submission for part 1 must include your plot and the printout of the best individual evolved using mutation. Here's mine:

Best tree:
(((a > a) + (((b > b) * abs(((abs((b / abs(a))) * (((a / b) / a) / b)) > ((abs((a * (b / b))) - (abs(a) / a)) + a)))) * a)) - (b + a))
Generation: 99
Size: 42
Depth: 11
Fitness: -0.0889936

Part 2. Implement a comparator ADT called LexLessThan which performs a lexicographic comparison of two trees *a* and *b* as follows: If the scores of *a* and *b* differ by less than 0.01, then tree *a* is "less than" tree *b* if *a* has more nodes than *b*. Otherwise, compare the trees by their score. The goal of using this comparator is to favour simpler trees only when their scores are similar. Repeat the experiment form part 1 using the new comparator. To do so, you must uncomment the line under "sort using comparaor class". After doing so, does evolution produce simpler trees? If not, try to increase the number of generations.

Part 3. Modify *anything* in genetic_programming_01.cpp in order to evolve a tree with high fitness and low complexity. What is the best fitness and complexity you can get in the single best tree?

Your submission for this question must include:

1. For part 1, a fitness plot showing the effect of your mutation operators and a printout of the best tree statistics.

2. For part 2, a printout of the best tree evolved using LexLessThan comparator ADT

3. For part 3, a printout of your single best tree (highest fitness and lowest complexity) and a brief description of your experiments (what you tweaked and why).

4. Your modified code named "1234-asg5-gp.cpp" where 1234 is your student ID.

Animation: If you uncomment the lines below “Evaluate best with animation“, the program will display a simple animation of the best tree interacting with the Cart Centering simulator. This can be very useful for debugging. However, note that animating the best tree will reset its score. For best results, do not animate the tree prior to generating its printout.

Question 3 [short answer, 3 marks]
Give an example of a worst-case sequence with n elements for insertion-sort using a priority queue. Show that insertion-sort runs in n^2 time on such a sequence.

Question 4 [short answer, 3 marks]
At which nodes of a heap can an entry with the largest key be stored?

Question 5 [short answer, 3 marks]
Let T be a complete binary tree such that node v stores the key-entry pairs $(f(v),0)$, where $f(v)$ is the level number of v. Is tree T a heap? Why or why not?

Question 6 [short answer, 3 marks]
Review Section “Down-Heap Bubbling after a Removal“ in the Goodrich test chapter 8. Explain why the case where the right child of r is internal and the left child is external was not considered in the description of down-heap bubbling.

Question 7 [short answer, 3 marks]
Is there a heap T storing seven distinct elements such that a preorder traversal of T yields the elements of T in sorted order? How about an inorder traversal? How about a postorder traversal?

Question 8 [short answer, 3 marks]
Bill claims that a preorder traversal of a heap will list its keys in nondecreasing order. Draw an example of a heap that proves him wrong.

Question 9 [short answer, 3 marks]
Hillary claims that a postorder traversal of a heap will list its keys in non-increasing order. Draw an example of a heap that proves her wrong.

Question 10 [short answer, 3 marks]
Draw the 11-entry hash table that results from using the hash function, $h(i) = (3i + 5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

Question 11 [short answer, 3 marks]
What is the result of the previous exercise, assuming collisions are handled by linear probing?

Question 12 [short answer, 3 marks]
What is the worst-case running time for inserting n items into an initially empty hash table, where collisions are resolved by chaining? What is the best case?