**COMP 285 (NC A&T, Spr ‘22)Homework 5**

#### Due.

Wednesday, February 23rd, 2022 @ 11:59 PM!

#### Homework Expectations:

Please see [Homework](https://www.comp285.ml/homework/#general-homework-information).

#### Exercises

The following questions are exercises. We encourage you to work with a group and discuss solutions to make sure you understand the material.

#### Points

This assignment is graded out of 100 points. However, you can get up to 120 points if you complete everything. These are not bonus points, but rather points to help make-up any parts you miss.

# Fun with Trees, Hashing, and Graphs

#### Written Problems

The following questions are to be submitted in written/typed form to gradescope.

# Interview Practice: Moose Tree

A moose comes to you with the following claim. They say that they have come up with a new kind of binary search tree, called mooseTree, even better than red-black trees!

More precisely, mooseTree is a data structure that stores comparable elements in a binary search tree. It might also store other auxiliary information, but the moose won’t tell you how it works. The moose claims that mooseTree supports the following operations:

* mooseInsert(k) inserts an item with key into the mooseTree, maintaining the BST property. It does not return anything. It runs in time .
* mooseSearch(k) finds and returns a pointer to node with key , if it exists in the tree. It runs in time .
* mooseDelete(k) removes and returns a pointer to an item with key , if it exists in the tree, maintaining the BST property. It runs in time .

Above, is the number of items stored in the mooseTree. The moose says that all these operations are deterministic, and that mooseTree can handle arbitrary comparable objects.

## Making Moose Trees

You want to get a better understanding of the mooseTree. To do so, you think it’d be helpful to write an algorithm that takes as input a vector of unsorted elements and creates a mooseTree using the methods above.

We’ll get you started.

algorithm makeMooseTree(A):  
 // Input: A is a vector of n comparable elements.  
 T = mooseTree() // creates an empty mooseTree.  
 // FILL THIS IN.  
 return T

## From Moose Trees to Sorted Vectors

Now that you understand how to make mooseTrees, you consider the opposite. How can you go from a mooseTree to an **sorted** vector?[[1]](#footnote-28)

We’ll get you started [[2]](#footnote-29).

algorithm getElements(T):  
 // Input: T is a mooseTree with n elements.  
 A = [] // an empty vector  
 // FILL THIS IN.  
 return A

## Wait a second...

You think the moose’s logic is a bit loosey-moosey. How do you know the moose is wrong?

**Notes:**

* You may use results or algorithms that we have seen in class without further justification.
* Proofs that just say "since it’s impossible to do these operations that fast in BST, this data structure can’t exist" is not sufficient.

# Interview Practice: Wise Duck

A wise duck has knowledge of an array of length , so that for all (note that the elements of are not necessarily distinct). You don’t have direct access to the list, but you can ask the wise duck *any yes/no questions about it. For example, you could ask “If I remove and swap with , would the array be sorted?” or “are ducks related to grebes?”*

This time you did bring a paper and pencil, and your job is to write down all of the elements of in sorted order.[[3]](#footnote-33) You are allowed to take all the time you need to do any computations on paper with the wise duck’s answers, but the wise duck charges one bandito burrito per question.

Design an algorithm which outputs a sorted version of which uses badito burritos. You may assume that you know and , although this is not necessary.

# Exercise: Hash Functions

With 217 students currently enrolled in computer science[[4]](#footnote-35) at North Carolina A&T University, we would like to keep track of student records.

The key for each person will be their 9-digit Banner ID.

## Our own hash functions!

Consider using a hash table of size and the hash function sum of each digits of . For example, .

Is this a good hash function to use? List two reasons supporting your decision.

## Good hash families

Consider a hash table of buckets. For , let last two digits of . For example, last two digits of . Define .

Is a good family of hash functions? Justify your decision.

# Interview Practice: DFS Basics

Consider the following directed acyclic graph (DAG):

In class, we saw how to use DFS to find a topological ordering of the the vertices; in the graph above, the unique topological ordering is . We saw an example where we happened to start DFS from the first vertex in the topological order. In this exercise we’ll see what happens when we start at a different vertex. Recall that when you run DFS, if it reached a node with no children (i.e. can’t go any further), then it will resume the search at an unvisited vertex.

Run DFS starting at vertex , breaking any ties by alphabetical order.[[5]](#footnote-40)

1. What do you get when you order the vertices by **ascending** start time?
2. What do you get when you order the vertices by **descending** finish time?

Run DFS starting at vertex , breaking any ties by **reverse** alphabetical order.[[6]](#footnote-42)

1. What do you get when you order the vertices by **ascending** start time?
2. What do you get when you order the vertices by **descending** finish time?

# Feedback: Homework Thoughts

In order to improve the homework in future iterations, complete this [form](https://forms.gle/LwMVLqf3FFuhJxVv7).

#### Coding Problems

The following questions are to be submitted as a ".zip" file on Gradescope.

# Coding

After completing the written portion of the assignment, you should submit it to [Gradescope](https://www.gradescope.com/courses/350304).

For the coding portion, tou can get your starter code [here](https://replit.com/team/COMP285/HW5-Code).

Note that the starter code also include a few test cases you can run on repl.it. However, the full test suite is the one run on Gradescope.

Please reference the README.md included in your starter code for detailed instructions.

# Submitting the Assignment

This assignment is a combination of written and programming questions. Both portions of the assignment should be submitted through [Gradescope](https://www.gradescope.com/courses/350304).

The "Homework 5: Fun with " assignment is the written portion, for which you should submit a **typed** response to the non-coding questions (questions 1-[4](#sec:last)). Each response should clearly be marked with its corresponding number. You are free to use the provided templates, print the questions and write your answers, or to simply type your responses on a blank document (whatever works for you).

The "Homework 5: Coding" is the programming portion of the assignment. For this portion, download the ".zip" file from replit and upload this ".zip" file as your answer to [Gradescope](https://www.gradescope.com/courses/350304). You can upload the assignment as many times as you want.

1. If you’re having trouble, take a look at Lecture 12, Slides 15-27 [↑](#footnote-ref-28)
2. Since the mooseTree is still a kind of binary search tree, you can access the root of mooseTree by calling mooseTree.root(). [↑](#footnote-ref-29)
3. Note that you don’t have any ability to change the array itself, you can only ask the wise duck about it. [↑](#footnote-ref-33)
4. See [here](https://www.ncat.edu/coe/departments/cs/undergrad-programs/index.php) if you’d like to see other years! [↑](#footnote-ref-35)
5. For example, if DFS has a choice between or , it will always choose . This includes when DFS is starting a new tree in the DFS forest. [↑](#footnote-ref-40)
6. For example, when DFS has a choice between or , it will always choose . This includes when DFS is starting a new tree in the DFS forest. [↑](#footnote-ref-42)