COMP 285 (NC A&T, Spr '22) Homework 4

Due. Wednesday, February 16th, 2022 @ 11:59 PM!

Homework Expectations: Please see Homework.

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.

Random Fun with Selection, Non-Comparions Sorts, and Data Structures

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

1 Interview Practice: Modifying the Select Algorithm (20 pt.)

After class, Kevin, Tobiloba, and Hassan got together to discuss a few modifications to the 'Select' algorithm we learned in class. Recall that in the 'Select' algorithm, the runtime is represented with the recurrence:

$$T(n) = O(n) + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$$

Here, $T(\frac{n}{5})$ is for selecting the pivot, and $T(\frac{7n}{10})$ is for the recursive call to select the k-th element. If you need a refresher, see Lecture 8 (and maybe Lecture 7).

1.1 Bigger, Bigger, Bigger Chunks! (10 pt.)

Consider the modified version of the select algorithm, where we split our array into $\lceil \frac{n}{7} \rceil$ groups of size ≤ 7 instead. What would be the recurrence relation for this modified version? What would be the running time?[1]

[We are expecting: The recurrence relation for this modified algorithm as well as the running time in $O(\cdots)$ notation.**]**

1.2 Smaller, Smaller, Smaller Chunks! (10 pt.)

Now consider another modified version of the select algorithm, where we split our array into $\lceil n/3 \rceil$ groups of size ≤ 3 instead. What would be the recurrence relation for this modified version? What would be the running time.

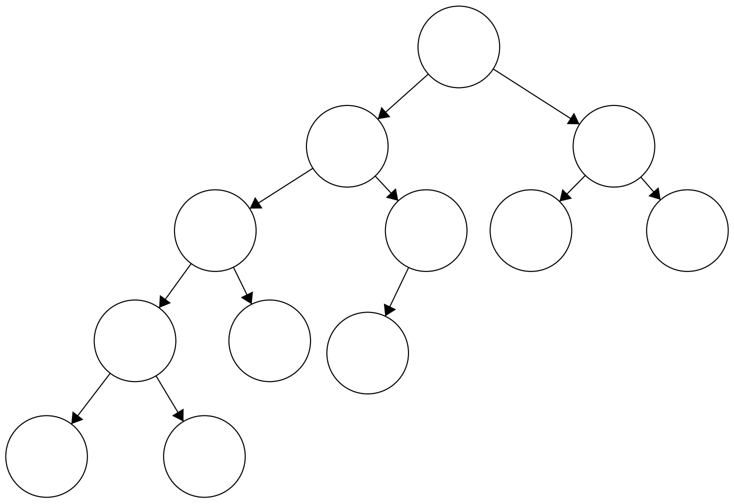
[We are expecting: The recurrence relation for this modified algorithm as well as the running time in $O(\cdots)$ notation.**]**

[1] You can't use the Master Theorem directly, but can you upper bound the recurrence? You can assume, for example, $T(n) = T(2n/5) + T(n/5) \leq T(3n/5)$.

2 Exercise: Practice with Colors and Shakespeare (10 pt.)

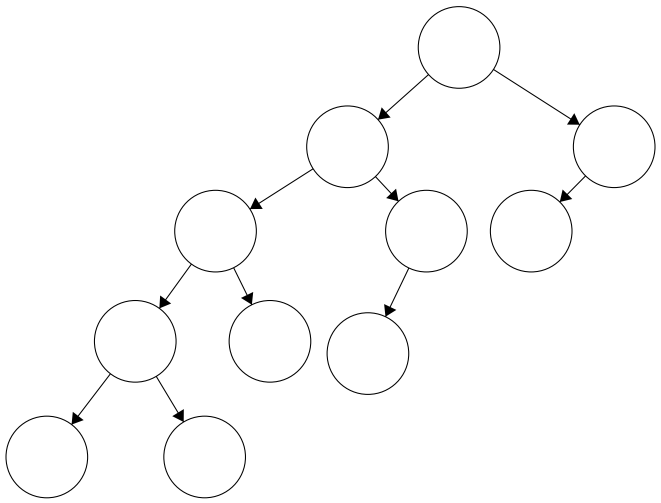
Look at each of the following examples of trees. If you can color the nodes red or black to give a legitimate red-black tree (eg, it must follow our the rules discused in Lecture 12), then provide such a coloring.

2.1 To Tree or not to Tree, that is the question... (5 pt.)



[We are expecting: Either an image of a colored-in red-black tree or a statement that "No such red-black tree exists." No need to justify your answer.**]**

2.2 What's in a tree? That which we call a node, by any other word would color just as well... (5 pt.)



[We are expecting: Either an image of a colored-in red-black tree or a statement that "No such red-black tree exists." No need to justify your answer.**]**

3 Interview Practice: Radix Sort for Strings (20 pt.)

As described in class, radix sort runs in $O(d(n+r))$ where n is the number of elements, r is the base, and d is the max length of the elements. For this problem, assume that radix sort uses counting sort, which does in fact run in $O(n+r)$ time. [**Note**: The question looks long but it is in fact very straightforward.]

3.1 Radix Sort on Characters (4 pt.)

Pretend we want to use radix sort to sort a list of words W in lexicographical (alphabetical) order. All words are padded with space characters (assume space comes before 'a' lexicographically) to make them the same length. For the following set W, describe what the three variables d, n, and r refer to (your explanation should include some reference to W or its elements) and what their values would be for this problem.

$$W = [\text{the, quick, brown, fox, jumps, over, the, lazy, dog}]$$

[We are expecting: values and descriptions for d, n and r**]**

3.2 Radix Sort on Bit Strings (4 pt.)

Now suppose we convert each of the strings in W to their ASCII representations (8-bit binary strings, with space mapping to 00100000, so the word 'the' becomes 40 digits long - 8 digits per character plus 8 digits per padding space to make it as long as the longest word in W). We still want to use radix sort, and we want to treat these bit strings as literal strings (i.e., do not try to interpret the 8 bit strings into decimal numbers). Now what are the values for d, n, and r?

[We are expecting: values and description for d, n and r**]**

3.3 Sorting with Dates (4 pt.)

Now we're back to using the character strings from part (a), but you happen to have the date (day, month, year) that each word was first published in an English dictionary. You want to sort first by date, then use lexicographical ordering to breakties. You will do this by converting each of the original words in W into words with date information (digits) prepended, appended, or inserted somewhere in the string. Write the string you would use to represent the word "jumps"jumps (first published November 19, 1562) so that it will be correctly sorted by radix sort for the given objective.

[We are expecting: a string**]**

3.4 Binary Conversions Revisited (4 pt.)

You decide that because you only ever use the words in a certain list V in everyday speech, you would like to save space and simply represent the first word in V with the binary value ‘0’, the second word with ‘1’, the third with ‘10’, etc., continuing to increment by one in binary (and no longer including date information). All subsequent occurences of a particular word w receive the same binary assignment as the first occurence of w, all strings are padded with ‘0’s to make them equal length. V has n words in it, where $n > 2$. Give the time complexity of radix sort on the list V with all words converted to their 0-padded binary strings and explain (informally) why that is correct. Simplify your answer as much as possible where values of d, n, or r are known.

[We are expecting: A runtime, and a brief justification**]**

3.5 One-Hot Encodings (4 pt.)

Not wanting to mess with binary conversions, you decide instead to represent the words in your vocabulary V with "one-hot" vectors (vectors of length n with all 0's except for a single ‘1’ in a position corresponding to a particular word. For example, in W, the word 'the' would be represented as the vector ‘10000000’, since there are eight unique words in the list). Give the new worst-case time complexity of radix sort on the list V, again simplifying as much as possible and explaining (informally) why that is the correct complexity.

[We are expecting: A runtime, and a brief justification**]**

4 Majority Vote in a Private Setting (10 pt.)

Difficulty This problem is meant to be challenging. You should actively discuss ideas with your class mates and help each other find a good solution. However, at the end of the day, you should be able to explain each step of the algorithm yourself and each character of your submission must be typed by you, personally.

The awesome country of Algolandia recently held an election. Every vote cast was electronically and on the block-chain, but unfortunately, there was a malfunction wiht the system and all the information is now private. We know the following facts:

- Every citizen of Algolandia voted.
- There are exactly n citizens.
- Exactly one candidate received strictly greater than $n/2$ votes.
- We don't know how many candidates there are.

Algolandia's government has heard about your algorithmic expertise, since you took COMP 285. As a result, they've hired you to help them! However, because of the private nature, Algolandia's cybersecurity software is only allowed to open a single ballot! Therefore, you must find a ballot that voted for the winning candidate. The only function you have unlimited access to is:

`ballotsMatch`(ballotA, ballotB)

This function returns True if ballotA and ballotB cast a vote for the same candidate, and False otherwise.

Design a deterministic divide-and-conquer algorithm which uses O(n log n) calls to `ballotsMatch` and returns the winning candidate.

[We are expecting: Pseudocode that calls `ballotsMatch` AND a clear English description of what your algorithm is doing as well as why your algorithm calls `ballotsMatch` $O(n \log n)$ times.**]**

BONUS (optional)! Is $O(n \log n)$ the fastest algorithm you can come up with? Can you give a better one? If so, provide a description of your algorithm and the pseudo-code. If not, why not?

5 Feedback: Homework Thoughts (10 pt.)

In order to improve the homework in future iterations, complete this form.

[We are expecting: You should submit the form with your @aggies.ncat.edu email to track your submission.**]**

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

6 Coding (50 pt.)

(50 pt.) After completing the written portion of the assignment, you should submit to Gradescope.

You can get your starter code for the coding portion here.

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.

The "Homework 4: Random Fun with Selection, Non-Comparions Sorts, and Data Structures" assignment is the written portion, for which you should submit a **typed** response to the non-coding questions (questions 1-**??**). 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 4: 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. You can upload the assignment as many times as you want.