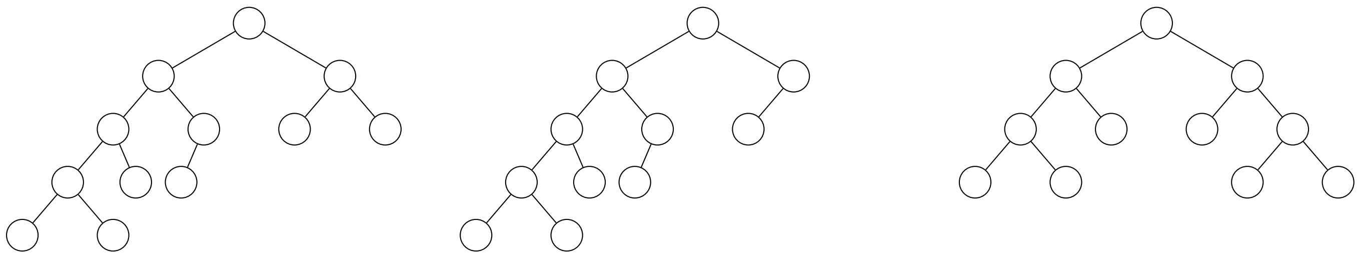
# CS 161 (Stanford, Winter 2026) Homework 4

**Style guide and expectations:** We do NOT accept handwritten solutions. Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we look for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material. Please do not distribute this material on any public forum.
**What we expect:** Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!
**Pair submissions:** You can submit in pairs for this assignment. If you choose to do this, please submit **one** Gradescope assignment per pair and be sure to tag both partners on your submission. Note that we still encourage exercises to be done solo first.

**Exercises.** The following questions are exercises. We suggest you do these on your own. As with any homework question, though, you may ask the course staff for help.

## 1 Red-Black Trees (6 pt.)

For each of the following examples, if the nodes can be colored red or black to make a legitimate red-black tree, then give such a coloring. If not, then say that they cannot.



**[We are expecting:** For each tree, either an image of a colored-in red-black tree or a statement "No such red-black tree." No justification is required. You are welcome to use LaTeX code to color the trees, take a screenshot and then color the trees in MSPaint, or make a hand-drawn copy and take a photo, or...**]**

## 2 Word Representations

As described in class, RadixSort 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 RadixSort uses CountingSort, which does in fact run in $O(n+r)$ time.

[**Note**: This exercise appears long, but your answers should be approximately 1-2 lines each.]

### 2.1 (2 pt.)

We want to use RadixSort 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 $W$, 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.

$W$ = [the, quick, brown, fox, jumps, over, the, lazy, dog]
**[We are expecting:** values and descriptions for $d$, $n$, and $r$.**]**

### 2.2 (2 pt.)

Now suppose we convert each of the strings in $W$ to their ASCII representation. Each character is converted to an 8-bit binary representation, with *space* mapping to 00100000. For example, 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 RadixSort, 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 descriptions for $d$, $n$, and $r$.**]**

### 2.3 (2 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 break ties. 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" (first published November 19, 1562) so that it will be correctly sorted by RadixSort.

For example: let's say we have the word "brainrot" which was first published on November 18th, 2024. How can we ensure this word is sorted after "jumps" since we sort on date (and then to break ties, on the alphabet)?

**Hint:** You can extract numerical information from the date.

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

### 2.4 (2 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). You can assume that all words you are trying to represent are unique.

$V$ has $n$ words in it, where $n > 2$. Give the time complexity of RadixSort 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 of why it is correct.**]**

### 2.5 (2 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 "10000000", since there are eight unique words in the list). Give the new worst-case time complexity of RadixSort 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 of why it is correct.**]**

**Problems.** The following questions are problems. You may talk with your fellow CS 161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

## 3 Lucky's Unlucky Tree (4 pt.)

Lucky comes to you with the following claim. He says that he has come up with a new kind of binary search tree, called `luckyTree`, even better than red-black trees!

More precisely, `luckyTree` is a data structure that stores comparable elements in a binary search tree. It might also store other auxiliary information, but Lucky won't reveal his secrets. Lucky claims that `luckyTree` supports the following operations:

- `luckyInsert(k)` inserts an item with key $k$ into the `luckyTree`, maintaining the BST property. It does not return anything. It runs in time $O(1)$.
- `luckySearch(k)` finds and returns a pointer to node with key $k$, if it exists in the tree. It runs in time $O(\log(n))$.
- `luckyDelete(k)` removes and returns a pointer to an item with key $k$, if it exists in the tree, maintaining the BST property. It runs in time $O(\log(n))$.

Above, $n$ is the number of items stored in the `luckyTree`. Lucky says that all these operations are deterministic, and that `luckyTree` can handle arbitrary comparable objects.

You think Lucky's being too lackadaisical here, as this data structure couldn't possibly work. How do you know Lucky is wrong?

**Notes:**

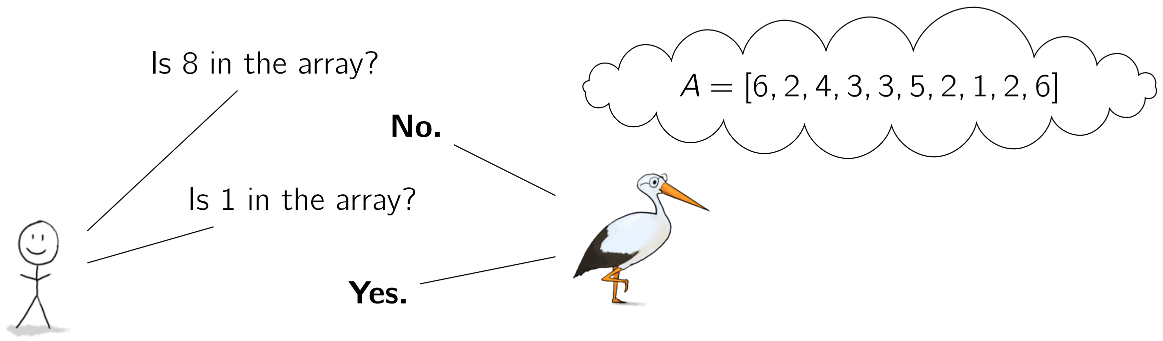
- Since the `luckyTree` is still a type of binary search tree, you can access the root of `luckyTree` by calling `luckyTree.root()`
- You may use any results or algorithms that we have seen in class without further justification. For example, you can traverse the tree however you like, since you have access to the left, right, and parent nodes at a given point.
- Proofs that just say "since it's impossible to do these operations that fast in BST, this data structure can't exist" are not sufficient.

**[We are expecting:** Formally prove that Lucky is wrong by showing that we can solve an algorithmic problem with a known lower bound with this data structure.**]**

## 4 Studious Stork (4 pt.)

Siggi the Studious Stork has knowledge of an array $A$ of length $n$, so that $A[i] \in \{1, \ldots, k\}$ for all $i$ (note that the elements of $A$ are not necessarily distinct). You don't have direct access to the list, but you can ask Siggi *any* yes/no questions about it. For example, you could ask "If I remove $A[5]$ and swap $A[7]$ with $A[8]$, would the array be sorted?" or "Are storks vegetarians?"

You came prepared with a paper and pencil, and your job is to write down all of the elements of $A$ in sorted order.[1] You are allowed to take all the time you need to do any computations on paper with Siggi's answers, but the studious stork charges one ice cream cone per question.



Design an algorithm which outputs a sorted version of $A$ and uses $O(k \log n)$ ice cream cones. You may assume that you know $n$ and $k$, although this is not necessary.

[1]Note that you don't have any ability to change the array $A$ itself, you can only ask Siggi about it.

**Hint:** One approach is to think first about what you would do for $k = 2$: that is, when $A$ contains only the numbers 1 and 2. What information do you need to write down a sorted version of $A$ in this case?

**[We are expecting:** Pseudocode and a clear English explanation of what it is doing. An explanation of why the algorithm uses $O(k \log n)$ ice cream cones. You do not need to prove that your algorithm is correct.**]**

## 5 Ternary Search Trees

In this problem, we will be talking about ternary search trees (TST). Ternary search trees are similar to binary search trees but rather than having just 2 pointers (left and right), they have 3 pointers (left, middle, and right).

A TST $T$ is a ternary tree in which each node contains a point of the form $(x, y)$ for some $x, y \in \mathbb{Z}$. Moreover, no two points in $T$ can share the same $x$ value and no two points can share the same $y$ value. A TST should satisfy the following three properties for every node $u$ with key $(x, y)$:

- Any point $(x_L, y_L)$ in the left-subtree of $u$ has $x_L < x$ and $y_L < y$.
- Any point $(x_M, y_M)$ in the middle-subtree of $u$ has $x_M > x$ and $y_M < y$.
- Any point $(x_R, y_R)$ in the right-subtree of $u$ has $x_R > x$ and $y_R > y$.

### 5.1 (3 pt.)

Prove that for any set of $n$ distinct points (where no two points share the same $x$-coordinate or $y$-coordinate), there exists a ternary search tree $T$ on these $n$ points that satisfy all three properties.

**[We are expecting:** A formal argument at the same level of rigor that we saw for the height of the Red-Black Tree in class.**]**

### 5.2 (3 pt.)

Prove or disprove that any set of $n$ distinct points (where no two points share the same $x$-coordinate or $y$-coordinate), there exists a ternary search tree $T$ on these $n$ points with height $h(T)$ such that $h(T) = O(\log(n))$.

**Hint**: Is there an arrangement of points in which you can only make a tree of height $O(n)$?

**[We are expecting:** A formal argument at the same level of rigor that we saw for the height of the Red-Black Tree in class.**]**

## 6 The Hiring Committee

The highly prestigious University of West Palo Alto is hiring a new computer science professor, and youre on the hiring committee. The committee is committed to hiring the best researcher and teacher they can find, someone who will have world-class research output and be an inspiration to their students.

The committee has received hundreds of applications, and the chair of the committee wants you to rank the applicants from best to worst so the committee can decide who to interview. You have been given a list of applicants that includes, for each applicant, the number of research papers they have published and their average score on teaching evaluations.

### 6.1 Imperfect proxy (4 pt.)

Pick one of the measures you have been given (number of research papers or average teaching evaluation score) and give at least one reason why that measure might not be a perfect proxy for the value it is intended to track.

**[We are expecting:** A few sentences in which you clearly state which measure you have chosen, the value that measure is intended to track, and at least one reason why that measure might not perfectly track that value.**]**

### 6.2 Insufficient information (4 pt.)

Using terminology introduced in class, explain why the list you have been given is insufficient to rank the applicants from best to worst *even if we assume* that number of research papers is a perfect proxy for research promise and average teaching evaluation score is a perfect proxy for teaching skill.

**[We are expecting:** A few sentences that answer the question using terminology introduced in class. Your answer should define the relevant terminology and explain how it applies to this case.**]**