DS 4300 - Spring 2025

Homework 01

Extra Credit Deadline: Sunday, January 12 2024 @ 11:59 pm
Regular Credit Deadline: Tuesday, January 14 2024 @ 11:59 pm

Directions:
1. From the File Menu, make a copy of this document in your own Google Drive account OR download an MS Word version.
2. Provide **professional-quality, type-set** solutions for each of the stated questions.
   - Do not delete the actual question statement. However, you may add additional space between the questions to accommodate your answers.
   - ***Your solutions must be typeset***. The solutions you incorporate into this document MAY NOT be handwritten. This includes not writing them on an iPad and then copying them (or a screen shot) into this document. I suggest you use some type of equation editor in your writing tool of choice or use LaTeX to typeset your responses where appropriate.
   - Any source code or pseudocode should be typed in a *monospaced font* like Courier New or Fira Mono, both available in Google Docs. **DO NOT paste screenshots of your code.**
   - To reiterate, handwritten solutions in any form will receive NO CREDIT.
3. **Generate a PDF of your document with your solutions**. DO NOT upload screenshots, images, or pictures. Reminder, DO NOT delete the question statements. If you compose your solutions in LaTeX, please retype the questions.
4. Upload your solutions document to GradeScope by the due date. **It is your responsibility to associate each question with your solution in GradeScope and click the submit button afterwards.** Failure to do so will result in no credit for any questions not linked with your solution *and will not be a valid reason to request a re-grade*.

Academic Collaboration Reminder:
Remember that you may not look at, copy, capture, screenshot, or otherwise take possession of any other students' solutions to any of these questions. Further, you may not provide your solutions in part or in whole to any other student. Doing any of the above constitutes a violation of academic honesty which could result in an F in this class and a referral to OSCCR.

What is permissible? You are free and encouraged to talk to your peers about the conceptual material from the lectures or the conceptual material that is part of this assignment. I'm confident you know where the line between collaborative learning and cheating sits. Please don't cross that line.

Question 1:

Linear search and Binary search aim to find the location of a specific value within a list of values (sorted list for binary search). The next level of complexity is to find two values from a list that together satisfy some requirement.

Propose an algorithm to search for a pair of values in an unsorted array of *n* integers that are closest to one another. Closeness is defined as the absolute value of the difference between the two integers. Your algorithm should not first sort the list. [10 points]

```
def find_closest_unsorted_pair(array):
      # checks if the array has at least two values
      n= len(array)
      if n < 2:
            return "Array must have more than two values"

      # defines variables for the desired values and the minimum difference
      min_diff= np.inf
      val1= 0
      val2= 0

      # compares all pairs within the list
      for i in range(n-1):
            for j in range(i+1, n):
                  # calculates the difference
                  diff = abs(array[i]-array[j])
                  # updates diff and values when diff is smaller than the current value
                  if diff < min_diff:
                        min_diff = diff
                        val1 = array[i]
                        val2 = array[j]

      # return closest values in the array
      return val1, val2
```

Next, propose a separate algorithm for a sorted list of integers to achieve the same goal. [10 points]

```
def find_closest_pair_sorted(array):
    # checks if the array has at least two values
    n = len(array)
    if n < 2:
        return "Array must have at least two elements"

    # defines variables for the desired values and the minimum difference
    min_diff = np.inf
    val1 = 0
    val2 = 0

    # compares all adjacent values (since the array is sorted)
    for i in range(n - 1):
        diff = array[i + 1] - array[i]
        # calculates the difference
        if diff < min_diff:
            # updates diff and values when diff is smaller than the current value
            min_diff = diff
            val1= array[i]
            val2 = array[i + 1]

    # return closest values from the array
    return val1, val2
```

Briefly discuss which algorithm is more efficient in terms of the number of comparisons performed. A formal analysis is not necessary. You can simply state your choice and then justify it. [5 points]

The second algorithm is more efficient than the first because it just needs to pass through the array once to compare adjacent values instead of finding the difference between every combination of values within the array.

Question 2:

Implement in Python an algorithm for a level order traversal of a binary tree. The algorithm should print each level of the binary tree to the screen starting with the lowest/deepest level on the first line. The last line of output should be the root of the tree. Assume your algorithm is passed the root node of an existing binary tree whose structure is based on the following BinTreeNode class. You may use other data structures in the Python foundation library in your implementation, but you may not use an existing implementation of a Binary Tree or an existing level order traversal algorithm from any source.

Construct a non-complete binary tree[1] of at least 5 levels. Call your level order traversal algorithm and show that the output is correct. [20 points]

```
class BinTreeNode:
    def __init__(self, value=0, left=None, right=None):
        self.value = value
        self.left  = left
        self.right = right

def level_order_reverse(root):
    # checks if the tree contains values
    if not root:
        return ""

    # initialize queue with the root as the first value
    queue = deque([root])
    # initialize list to store levels
    level_list = []

    # go through entire queue
    while queue:
        # number of nodes at current level
        num_nodes = len(queue)
        # initialize list to store values of nodes at current level
        level_values = []

        # go through each value within the current level
        for val in range(num_nodes):
            # take the first value in the queue and add to the list of values
            current_node = queue.popleft()
            level_values.append(current_node.value)

            # add the current nodes left then right sub-nodes to the queue if they have
them
            if current_node.left:
                queue.append(current_node.left)
            if current_node.right:
                queue.append(current_node.right)
```

[1] A complete binary tree is a binary tree where every level except possibly the last level is full, meaning no additional nodes could fit on that level.

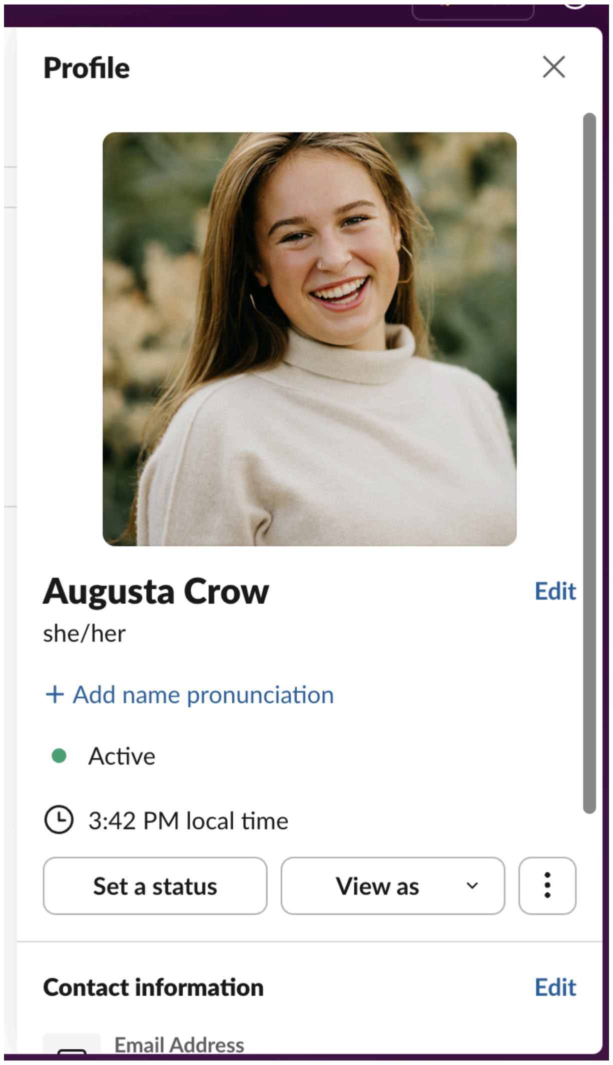
```
        # add the values of the current level into the level list
        level_list.append(level_values)

    # change the levels list into string values to print the tree
    result = "\n".join(" ".join(map(str, level)) for level in level_list[::-1])
    return print(result)
```

[53]:
```
# example of algorithm usage in a reverse traversal

# create binary tree
root = BinTreeNode(20)
root.left = BinTreeNode(15)
root.left.left = BinTreeNode(12)
root.left.right = BinTreeNode(19)
root.left.right.left = BinTreeNode(16)
root.left.right.left.right = BinTreeNode(17)
root.right = BinTreeNode(25)
root.right.left = BinTreeNode(23)
root.right.left.right = BinTreeNode(24)

# call algorithm
level_order_reverse(root)
```

```
17
16 24
12 19 23
15 25
20
```

Question 3:

Fill out your profile on Slack including a clear head shot. This will help the TAs and I, as well as you all, associate names with faces quicker. Paste a screen shot of your completed Slack profile below. [5 points]



Profile

Augusta Crow

she/her

+ Add name pronunciation

● Active

3:42 PM local time

Contact information

Email Address