

## 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 (in Python) 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 closest_pair_unsorted(unsorted_array):
    ''' Search for pair of values in unsorted array that are closest to one another.
    Closeness is defined as absolute value of difference between two integers. '''
    # if length of array is less than 2, return error message
    if len(unsorted_array) < 2:
        return "Given array does not contain at least two values."

    # initialize variables for closest pair(s) and their difference
    closest_pairs = []
    closest_difference = float("inf")

    # loop through array to find two values that are closest
    for i in range(len(unsorted_array)):
        for j in range(i + 1, len(unsorted_array)):
            pair_difference = abs(unsorted_array[i] - unsorted_array[j])

            # if closer pair is found, updated closest_difference and reset
            # closest_pairs to new pair
            if pair_difference < closest_difference:
                closest_difference = pair_difference
                closest_pairs = [(unsorted_array[i], unsorted_array[j])]
            # if another pair has the same difference as closest_difference, add to
            # closest_pairs list
            elif pair_difference == closest_difference:
                closest_pairs.append((unsorted_array[i], unsorted_array[j]))
    return closest_pairs
```

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

```
def closest_pair_sorted(sorted_array):
    ''' Search for pair of values in sorted array that are closest to one another.
    Closeness is defined as absolute value of difference between two integers. '''
    # if length of array is less than 2, return error message
    if len(sorted_array) < 2:
        return "Given array does not contain at least two values."

    # initialize variables for closest pair(s) and their difference
    closest_pairs = []
    closest_difference = float("inf")

    # loop through array to find two values that are closest
    for i in range(len(sorted_array)):
        # make sure loop does not go out of bounds/beyond array
        if i < len(sorted_array) - 1:
            pair_difference = abs(sorted_array[i] - sorted_array[i + 1])
```

```

# if closer pair is found, updated closest_difference and reset
# closest_pairs to new pair
if pair_difference < closest_difference:
    closest_difference = pair_difference
    closest_pairs = [(sorted_array[i], sorted_array[i + 1])]
# if another pair has the same difference as closest_difference, add to
# closest_pairs list
elif pair_difference == closest_difference:
    closest_pairs.append((sorted_array[i], sorted_array[i + 1]))
return closest_pairs

```

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 sorted array algorithm is more efficient in terms of the number of comparisons performed. This is because the sorted array algorithm is only comparing integers that are adjacent to one another whereas the unsorted array algorithm is comparing all possible pairs of integers in the array. This is necessary because an integer in an unsorted array could be closest to a value that is not directly next to it, while an integer in a sorted array is going to be closest to the value next to it because the array has already been sorted by value. Therefore, the sorted array algorithm doesn't need to perform as many comparisons to find the closest pair of integers as the unsorted array algorithm and is thereby more efficient.**

## 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.

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

def level_order_traversal(root):
    ''' Algorithm for level order traversal of binary tree given root node of BinTreeNode
    class binary tree. Algorithm will print each level of binary tree starting with
    lowest/deepest level and ending with the root of the tree. '''
    # if no root is given, return error message
    if not root:
        return "No root was given."

    # initialize list to store nodes by level
    tree_levels = []

    # initialize queue for nodes to be processed (starting with root)
    queue = [root]

    # search tree level by level
    while queue:
        # set up variables for current level node values and size
        current_level_values = []
        level_size = len(queue)

        # process current level nodes
        for _ in range(level_size):
            # remove node being processed from front of queue and add value to
            # current_level_values
            node_to_process = queue.pop(0)
            current_level_values.append(node_to_process.value)

            # check if node being processed has children and if so, add to queue
            if node_to_process.left:
                queue.append(node_to_process.left)
            if node_to_process.right:
                queue.append(node_to_process.right)

        # add current_level_values to tree_levels
        tree_levels.append(current_level_values)

    # reverse order of tree_levels to print out deepest level first
    for level in reversed(tree_levels):
        print(" ".join([str(node) for node in level]))
```


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

```
# construct a non-complete binary tree of at least 5 levels
root = BinTreeNode(1)

root.left = BinTreeNode(2)
root.left.left = BinTreeNode(4)
root.left.left.left = BinTreeNode(6)
root.left.left.left.left = BinTreeNode(8)
root.left.left.left.left.left = BinTreeNode(10)

root.right = BinTreeNode(3)
root.right.right = BinTreeNode(5)

# call level order traversal algorithm and show output is correct
level_order_traversal(root)
```



```
[38]: # call level order traversal algorithm and show output is correct
      level_order_traversal(root)

      10
      8
      6
      4 5
      2 3
      1
```

**Screenshot of output when level order traversal algorithm is called on tree created above (output is correct)**


---

<sup>1</sup> 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.

### 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



**Sophie Sawyers**

[Edit](#)

Student

she/her · so-fee soy-yers

●

Active

🕒

9:00 PM local time


Set a status

View as ▾

⋮

Contact information

[Edit](#)



Email Address  
sawyers.s@northeastern.edu

[+ Add Phone](#)

About me

[Edit](#)

[+ Add Start Date](#)