

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]

```
# function that searches for a pair of values in an unsorted array of n integers that are closest #
(numerically)
def find_close(array):
    pair = []
    min_diff = 10**100
    for i in range(len(array)):
        for j in range(i+1, len(array)):
            diff = abs(array[i] - array[j]) # calculate closeness
            if diff < min_diff: # if pair is closest so far:
                pair = [array[i], array[j]] # update pair
                min_diff = diff # update closest difference
    return pair
```

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

```
# function that searches for a pair of values in a sorted array of n integers that are closest #
(numerically)
def find_close_sorted(array):
    pair = []
    min_diff = 10**100
    for i in range(len(array)):
        if i+1 == len(array): # stop when at last element of array
            return pair
        elif abs(array[i] - array[i+1]) < min_diff: # if current difference is the closest
            pair = [array[i], array[i+1]] # update pair
            min_diff = abs(array[i] - array[i+1]) # update closest difference
    return pair
```

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 (`find_close_sorted()`) is more efficient. This algorithm only performs $n - 1$ comparisons. On the other hand, the first algorithm (`find_close()`) performs $(n(n-1)) / 2$ comparisons. For example, on an array of size 10, the first algorithm would perform 45 comparisons and the second algorithm would perform 9 comparisons.

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.

```
# helper function to print traversal on multiple lines and in reverse order
def print_traversal(traversal):
    reversed_traversal = sorted(traversal, key=lambda x: x[1], reverse=True) # order traversal by desc. level
    output = ''
    levels = [x[1] for x in reversed_traversal] # get only the levels
    level = max(levels) # get maximum level

    for tup in reversed_traversal:

        if level == tup[1]:
            output = output + str(tup[0]) + ' '

        else:
            level = level - 1
            output = output + '\n' + str(tup[0]) + ' '

    print(output)

from collections import deque

# perform level order traversal from lowest level to root
def level_order_traversal(bst_root):
    de = deque([(bst_root, 0)]) # add root and level to a deque
    traversal = [] # initialize list to hold node values and levels

    # base case
    if bst_root is None:
        return []

    # loop while there are items in the deque
    while len(de) != 0:
        btnode, level = de.popleft()
        traversal.append((btnode.value, level)) # add node value and its level to traversal list

        if btnode.left is not None:
            de.append((btnode.left, level + 1)) # add left subtree and its level to deque

        if btnode.right is not None:
            de.append((btnode.right, level + 1)) # add right subtree and its level to deque

    return print_traversal(traversal) # send traversal to function to be reversed/printed
```

Construct a non-complete binary tree¹ 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

# level 1
root = BinTreeNode(22)
# level 2
root.left = BinTreeNode(10)
root.right = BinTreeNode(28)
# level 3
root.left.left = BinTreeNode(5)
root.left.right = BinTreeNode(11)
root.right.left = BinTreeNode(25)
root.right.right = BinTreeNode(33)
# level 4
root.left.left.left = BinTreeNode(2)
root.left.left.right = BinTreeNode(7)
root.left.right.right = BinTreeNode(15)
root.right.left.right = BinTreeNode(26)
root.right.right.left = BinTreeNode(30)
root.right.right.right = BinTreeNode(45)
# level 5
root.left.right.right.left = BinTreeNode(14)
root.right.right.left.left = BinTreeNode(32)
```

Output:

```
level_order_traversal(root)
```


```
14 32
2 7 15 26 30 45
5 11 25 33
10 28
22
```

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.

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 ✕



Caroline Han Edit

[+ Add name pronunciation](#)

[+ Add pronouns](#)

● Active

🕒 8:09 PM local time

Set a status

View as ▼

⋮

Contact information Edit

✉ Email Address

han.car@northeastern.edu

[+ Add Phone](#)

About me Edit

[+ Add Start Date](#)