t's time to show off what you've learned about technical interviewing!

For this project, you will be given five technical interviewing questions on a variety of topics discussed in the technical interviewing course. You should write up a clean and efficient answer in Python, as well as a text explanation of the efficiency of your code and your design choices. A qualified reviewer will look over your answer and give you feedback on anything that might be awesome or lacking—is your solution the most efficient one possible? Are you doing a good job of explaining your thoughts? Is your code elegant and easy to read?

Answer the following questions:

**Question 1**

Given two strings s and t, determine whether some anagram of t is a substring of s. For example: if s = "udacity" and t = "ad", then the function returns True. Your function definition should look like: question1(s, t) and return a boolean True or False.

## Question 2

Given a string a, find the longest palindromic substring contained in a. Your function definition should look like question2(a), and return a string.

**Question 3**

Given an undirected graph G, find the minimum spanning tree within G. A minimum spanning tree connects all vertices in a graph with the smallest possible total weight of edges. Your function should take in and return an adjacency list structured like this:

{'A': [('B', 2)],

'B': [('A', 2), ('C', 5)],

'C': [('B', 5)]}

Vertices are represented as unique strings. The function definition should be question3(G)

**Question 4**

Find the least common ancestor between two nodes on a binary search tree. The least common ancestor is the farthest node from the root that is an ancestor of both nodes. For example, the root is a common ancestor of all nodes on the tree, but if both nodes are descendents of the root's left child, then that left child might be the lowest common ancestor. You can assume that both nodes are in the tree, and the tree itself adheres to all BST properties. The function definition should look like question4(T, r, n1, n2), where T is the tree represented as a matrix, where the index of the list is equal to the integer stored in that node and a 1 represents a child node, r is a non-negative integer representing the root, and n1 and n2 are non-negative integers representing the two nodes in no particular order. For example, one test case might be

question4([[0, 1, 0, 0, 0],

[0, 0, 0, 0, 0],

[0, 0, 0, 0, 0],

[1, 0, 0, 0, 1],

[0, 0, 0, 0, 0]],

3,

1,

4)

and the answer would be 3.

**Question 5**

Find the element in a singly linked list that's m elements from the end. For example, if a linked list has 5 elements, the 3rd element from the end is the 3rd element. The function definition should look like question5(ll, m), where ll is the first node of a linked list and m is the "mth number from the end". You should copy/paste the Node class below to use as a representation of a node in the linked list. Return the value of the node at that position.

**class** **Node**(object):

**def** **\_\_init\_\_**(self, data):

self.data = data

self.next = **None**

Q1

To check if whether two strings are anagram, all the charaters in both the strings are to be counted. If the count happens to be same, then both the strings are anagram.

Fisrt, a count dictionary for t is set to check with every substring set in s.the function returns true is set is anagram of t, else False. Comparing of the substrings can be done using loops. Looping through all the possible substring will consider the worst case of O(len(s)). The time complxity is O(len(s)).Since there are limited number of characters the space complexity is O(1).Empty strings are not considered anagram.

Q2

A Palindrome is a string, which is same compared from both the ends.

Ex: ‘madam’ = ’madam’.

The Problem requires us to find the longest possible substring palindrome in a given string.Lets, find all the possible substrings og a string and check if a substring is a palindrome will solve the problem.If a palindrome is found, returns the longest palindrome found.

Q3

Minimum spannning tree from an unidirected graph based on the weight of the edges between the vertices to be found. A Minimum spanning tree includes the edges with minimum weights from one vertex to another and must not have any cycles. Input is a graph represented in the form of dictionary.We need to iterate through the dictionary to read and then save all the edges to sort them by weight.we need to check if the path with a lesser wright is possible and if an edge exists in the minimum spanning tree.

Q4

We need to find the least common ancestor of two nodes in a binary search tree.

Binary search tree is representas as a 2D matrix , where each node is an integer.Given both the nodes given are in Bst.Lets take T as a tree and r as an integer valur of root, and n1 and n2 are non-negative integers. The approach is simple and can be recursively used considering Bst is sorted.

Q5

To solve this problem,we will loop through the linked list twice. The first iteration will provide its length . Then the second iteration can determine how many elements to traverse, to get the mth element from the back. The time will still be O(n) for traverse through ll twice. Since we only store the length, the space complexity will be O(1). One special case is to deal with circular linked list. If that case is not treated, then the get\_length() function might stuck in a infinite loop when it hits a circular linked list. We can use two nodes each traverse at different rate to deal with the case. If both of them ever meet, then we know the linked list is circular and we can terminate the loop. This part will not take additional time in traversal and will only need to store one additional node durning the traversal. Therefore the overall time and space complexity will not be affected.