

Interview Technical Questions

Report

Prateek Kacker | Nanodegree for Machine Learning Engineer | 05/01/17It'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

## **Project Submission**

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?

### Submission Instructions

* For each question, create a solution in Python (version 2). All solutions should be functions named as “question1”, “question2”, et cetera. Feel free to make additional helper functions or classes as needed. Code solutions must be in a file called "solutions.py".
* In the same .py file, include at least 3 test cases for each solution. For each test case, write the function call with the input you want to test and print it to the console, like "print question1()". On the next line, comment out the output you expect to see from that function call. At least 2 of these must be edge cases, testing inputs such as null values, empty inputs, unusually large values, et cetera.
* Write up an explanation for each question in a single separate text file (called "explanations.txt"). Your paragraph should not be a detailed walkthrough of the code you provided, but provide your reasoning behind decisions made in the code. For example, why did you use that data structure? You also need to explain the efficiency (time and space) of your solution.
* Compress your one Python and one text file into a .zip, and submit.

### Evaluation

Check out the rubric [here](https://review.udacity.com/#!/projects/7736707697/rubric)!