Question 1: Explanation

In Question1, the faction takes two strings (s and t) and find out whether one (t) is anagram of the other (s). Here, in the string t, all the different letter combinations are considered to find out anagrams. If an anagram is found, the program returns true.

The efficiency of the program depends on how long are the given strings. However, to increase the time efficiency itertools.permutations() function is used to generate different letter combinations in a word.

word\_combination\_list = list(itertools.permutations(t\_list, len(t\_list)))

This function is helpful to eliminate unnecessary ‘for’ loop iterations and complexity. Also, it is written in lower level languages like C, it is faster than conventional Python ‘for’ loops. Since I am using minimum number of variable to store data during the program running, the space efficiency also in good standing.

Question 2: Explanation

In this function, the longest palindromic substring is found in a given string. To achieve this, a substring is divided in to two from the center and check whether they are backwardly matching or not.

The overall efficiency of this function also depends on the length of the string. Minimum number of variables used to increase the space efficacy. Using isOdd() function, code repetition is minimized. Also, minimize the unnecessary looping, following string reverse method is used.

second\_part = substring[center:][::-1]

These measures helped to improve time efficiency.

Question 3: Explanation

In this question, minimum spanning path for the given graph is determined. First the dictionary form of the graph is converted to Graph object. The Graph object contains the sun objects Nodes and Edges. The path finding is carried out. The total value of the edges (lowest values) for each possible path that contain all the nodes is considered to find the minimum spanning path.

The time taken to run the program is depend on how many nodes and edges are in the graphs. The high numbers of nodes and edges can be increase the computational time exponentially (typical for graphs).

To increase the time efficacy, some iterations are minimized by using inbuilt methods like in following code line. Depth first search algorithm used to find paths.

path[::-1]

Several functions are used to minimize the code repetition.

Following data structures used to define Graph object.

class GNode(object):

def \_\_init\_\_(self, value):

self.value = value

self.edges = []

self.visited = False

class Edge(object):

def \_\_init\_\_(self, value, node\_from, node\_to):

self.value = value

self.node\_from = node\_from

self.node\_to = node\_to

class Graph(object):

def \_\_init\_\_(self, nodes=None, edges=None):

self.nodes = nodes or []

self.edges = edges or []

self.node\_names = []

self.\_node\_map = {}

Question 4: Explanation