CS 231, Spring 2024 A. Jamshidpey

ASSIGNMENT 1

Due date: May 29, 2024 16:00 (Waterloo time)

WARNING: To receive credit for this assignment, you checked "I agree" to the academic integrity declaration. Please keep all the rules in mind as you complete your work.

Coverage: Through Module 3

This assignment consists of a written component and a programming component. Please read the instructions on the reference page for assignments carefully to ensure that you submit each component correctly.

Note: In assignment questions that specify the use of a particular paradigm, you are expected to come up with a new algorithm using that paradigm. It is not sufficient to implement a class example as a helper function and declare that the paradigm has been used. For example, using selection sort is not sufficient for a problem asking you to use a greedy approach.

Written component

For full marks, you are expected to provide a brief justification of any answer you provide.

W1. *[7 marks]* For your long-term strategic plan, you would like to identify where to place offices for your business. Although you'd like to have many offices, it is essential that each pair can communicate with each other in a cost-effective manner. Using your knowledge from CS 231, you choose to represent the information as a graph, where potential office locations are represented by vertices and the cost of communication between a pair of locations is represented by a positive weight on an edge.

We use the term *cluster* to refer to a set of vertices (represented by their IDs) such that there is an edge between each pair of vertices in the set. The *total value* of a cluster is the sum of the weights of the edges with both endpoints in the cluster. For example, in Sample graph 4, $\{e\}$ is a cluster of size 1 (with total value 0, since there are no edges with endpoints in the cluster), $\{e, f\}$ is a cluster of size 2 (with total value equal to 3, the weight of the edge ef), and $\{e, f, g\}$ is a cluster of size 3 (with total value equal to the sum of the weights of edges ef, fg, and eg, or $3 + 4 + 9 = 16$). The set $\{e, g, d\}$ is not a cluster, since there is no edge ed.

Please see the reference page on the module `graphs.py` for illustrations of the sample graphs.

(a) *[2 marks]* In Sample graph 4, what is the **maximum** size of a cluster, and which cluster of maximum size has the **minimum** total value? If there is more than one correct answer, any one will do. Briefly justify your answer, explaining why it is a cluster, why its size is the maximum possible, and why its total value is the minimum possible.

(b) *[2 marks]* Give the input and output specifications for an **enumeration problem** for the problem of finding clusters of size k with the minimum total value. You may use the terms defined above without supplying definitions.

(c) *[2 marks]* By adding a bound on the total value, give the input and output specifications for a related **decision problem**.

(d) *[1 mark]* Give **a different** example of a real-world situation to which the decision problem can be applied, this time when the edge weights are all equal to 1.

Your situation does not have to be useful. Even ridiculous situations are acceptable, as long as you justify why the problem applies.

W2. *[5 marks]* In this question, we will consider a vertex v in a graph to be *k-friendly* if v has at least k neighbours, and at least k of those neighbours have at least k neighbours (including v). For example, in Sample graph 3, the vertex with ID c has three neighbours (namely, the vertices with IDs a, d, and e) which have, respectively, 2, 5, and 2 neighbours. The vertex with ID c is both 1-friendly (it has at least one neighbour with at least one neighbour each) and 2-friendly (it has at least two neighbours with at least two neighbours each), but is not 3-friendly (it does not have at least three neighbours with at least three neighbours each).

Write a pseudocode function FRIENDLY that consumes a graph *Instance*, the ID *One* of a vertex in *Instance*, and an integer k. Your function should produce True if the vertex with ID *One* is k-friendly and False otherwise.

Use the methods in `graphs.py` to write your function, but translate them into pseudocode as described on the reference page on pseudocode. Please see the reference page on `graphs.py` for sample graphs as well as the methods.

W3. *[8 marks]* **Use the recipe** for analyzing worst-case running time to determine the worst-case running time of IS_DESCENDANT as a function of n (the number of nodes in *Instance*), expressed in Θ notation. Be sure to refer to each line in the function and justify the values you determine. For full marks, a list of line numbers and running times for each line is **not** sufficient. Make sure to show each step of the recipe.

Please see the reference page on the module `trees.py` for the costs of tree operations, and the reference page on costs for the costs of all other operations. You may also find it helpful to consult the reference page on pseudocode.

```
IS_DESCENDANT(Instance, One, Two)
INPUT:    A nonempty tree Instance and nodes One and Two in Instance
OUTPUT:   True if One is a descendant of Two and False otherwise
1  Root ← TREE_ROOT(Instance)
2  Current ← One
3  while Current $\neq$ Root
```

```
4       Current ← Parent(Instance, Current)
5       if Current == Two
6           return True
7   return False
```

W4. *[6 marks]* **Use the formal definition of** Θ to show that $5\log^3 n - \log n + 10 \in \Theta(\log^3 n)$. Remember that $\log^3 n = (\log n)(\log n)(\log n)$, and that in computer science, log is used to denote $\log_2$. Make sure to use specific values of constants in the definition.

W5. *[4 marks]* In this problem, we consider the use of exhaustive search to solve the following problem:

Input: A set U, a set $\mathcal{S}$ of subsets of U, and a positive integer k

Output: Yes or no, answering "Does there exist a subset T of U of size k such that for each set in $S \in \mathcal{S}$, S contains at least one element of T?"

For example, for the instance $U = \{a, b, c, d, e\}$, $\mathcal{S} = \{\{a, b\}, \{c, d\}\}$, and $k = 2$, the answer is yes, since $T = \{a, c\}$ is a subset of size 2, and T contains at least one element of the set $\{a, b\}$ (namely, a) and at least one element of the set $\{c, d\}$ (namely, c). Other examples of such subsets include $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$.

Notice that T does not need to be a member of $\mathcal{S}$; it can be any subset of U. In this example, the element e of U does not appear in $\mathcal{S}$, as allowed by the input specification.

For the same U and $\mathcal{S}$ but with the value $k = 1$, we have a no-instance, since there is no subset consisting of a single element that appears in both $\{a, b\}$ and $\{c, d\}$.

(a) *[3 marks]* Using a few sentences (no pseudocode is required), describe an algorithm that solves the problem using exhaustive search. Make sure to define the possibilities and explain when you produce "Yes" and when you produce "No".

(b) *[1 mark]* How many possibilities are there? Express your answer in order notation, using Θ, as a function of u (the size of U), s (the size of $\mathcal{S}$, that is, the number of sets in $\mathcal{S}$), and k. Provide a brief justification of your answer.

Programming component

Please read the information on assignments and Python carefully to ensure that you are using the correct version of Python and the correct style. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

Although submitting tests is not required, it is highly recommended that you test your code. For each assignment question, create a testing file that imports your submission and tests the code. Do not submit your testing file.

For any of the programming questions in this assignment, you may import any of the following files: `check.py`, `grids.py`, `trees.py`, `graphs.py`, and `equiv.py`, as well as built-in modules such as `math`, `itertools`, and `copy`.

Please see the reference pages for information on the modules `grids.py`, `trees.py`, `graphs.py`, and `equiv.py`.

P1. *[5 marks]* Write a function `num_locations` that consumes a grid `grid` and a string `item`, and produces the number of entries in `grid` that contain `item`.

For example, in Sample grid 1, for the item `"red"`, the function will produce 3, as there are three entries with the item `"red"`. For the item `"mauve"`, the function will produce 0, as there are no entries with the item `"mauve"`.

Submit your work in a file with the name `numlocations.py`.

P2. *[6 marks]* Write a function `ancestor_colours` that consumes a tree `tree` and the ID `choice` of a node in the tree, and produces a list of strings of colours of ancestors of the node with ID `choice` in `tree`. Your output should be in order from the parent of the node up through to the root.

For example, in Sample tree 3, for the ID `"e"`, the function will produce the list `["green", "red"]` since the ancestors of the node with ID `"e"` are the node with ID `"c"`, which has the colour green, and the node with ID `"a"`, which has the colour red. For the ID `"a"`, the function will produce the empty list, since the root has no ancestors.

Submit your work in a file with the name `ancestorcolours.py`.

P3. *[14 marks]* In this question, we solve the problem of finding the minimum total value for a cluster of a specified size. Write a function `cluster_exhaustive` that consumes a non-empty graph `graph` and a positive integer `size` (of size at most the number of vertices in `graph`) and produces the minimum total value of any cluster of size `size`, or 0 if there is no cluster of size `size`. Use exhaustive search to solve the problem. Note: Be careful to keep your test examples small in order to keep running time reasonable.

For example, for Sample graph 2 the output for size 2 will be 5, since the cluster containing the vertices with IDs `"a"` and `"c"` has minimum total value of any cluster of size 2 (that is, the endpoints of a single edge in the graph). For a cluster of size 4, the output will be 111, as the cluster containing all four vertices has a total value of 111, the sum of the weights of all of the edges in the graph. The output for a cluster of size 0 or 1 will be 0, since a cluster of size 0 or 1 has no edges. As another example, for Sample graph 5 the output for size 6 will be 0, since the graph contains no cluster of size 6.

To create example graphs for tests, create text files and then convert them into graphs using the function `make_graph`, provided in the module `graphs.py`. You can use the function `make_graph_text_file` (also provided in the module `graphs.py`) to speed up the process of creating text files. You will need to add weights by hand, but at least part of the work can be automated.

Submit your work in a file with the name `clusterexhaustive.py`.