Computational Thinking – Modelling with Graphs

Summative Assignment

Lecturer: George Mertzios
Assessment: summative
Hand-in method: DUO
Document format: PDF
Deadline: Friday 8th March 2019, 14:00

Description of the Assignment: The Assignment is divided into two parts, Parts A and Part B. Part A concerns *graph coloring* (40% of the total marks) and Part B concerns *graph traversing* (60% of the total marks). Each part is subdivided into Questions.

In each part of the Assignment, you are provided with 5 python (i.e. **.py*) files, each of them containing one specific input graph. Please copy all these files to your working Python folder (such that you can access them in IDLE). Each one of these files contains a function *Graph()* which returns the appropriate graph. Furthermore, these graphs will already have some initialization of all the necessary attributes for the nodes. Suppose that, for example, the file is called *graph1.py*. In this case, if you write the following in IDLE:

```
>>> import graph1
>>> G = graph1.Graph()
```

then *G* will be the graph that is constructed by the function *Graph()* of the file *graph1.py*.

For each Question of Parts A and B you should write some code in Python. Then you should execute your code providing as input the graphs that are created by the corresponding graph files. For each Question of each Part, you are asked to submit both:

- your code **<u>and</u>**
- the output of this code in a separate .txt file. To produce this separate text file with the output of your code, please **<u>just copy-paste your output into this .txt file.</u>** Please note that you should **<u>not</u>** implement any additional functions (such as "file.write" or similar) that automatically prints the output into a file.

Details follow in the specific description of each Part below. Also, **<u>to avoid unnecessary loss of marks, please do not change at all the way the output is produced in the .py files that are provided to you.</u>** It is also important that you prepare and run your code in **<u>Python 3.6</u>** (not in other python versions). Otherwise you may risk unnecessary loss of marks, as I will test your code in Python 3.6.

In the specific description of each Part below, whenever you are asked to submit a file (containing either code or text), it is specified how to include **<u>your 6-digit CIS-username *XXXXXX*</u>** in the name of the file (this code is of the form *abcd12* and you can find it also on your Campus-card).

PART A (40%)

You are given the files *graph1.py*, *graph2.py*, *graph3.py*, *graph4.py* and *graph5.py*. Each of these files contains a function *Graph()* which returns an input graph.

Question A.1 (20%).
You are given a file *greedy_col_basic_incomplete.py*. This file has two (incomplete) functions *find_smallest_color(G,i)* and *greedy(G)*. Complete these two functions and save your code in a file that is called *greedy_col_basic.py* such that, when you run this new file:

- it visits the vertices of the input graph sequentially in the order 1, 2, 3, 4, … (i.e. regardless of whether the next visited node is adjacent or not to any of the previously visited nodes),
- at every step it assigns (in a greedy fashion) to the currently visited node the smallest possible color (where every color is a positive integer 1, 2, 3, …) such that no two adjacent nodes receive the same color, and finally
- it outputs the constructed coloring and the number *kmax* of the different colors in this coloring (using the printing commands that are given within the function *greedy(G)* in the file *greedy_col_basic_incomplete.py*).

Each time the algorithm visits a new node *i* of a graph *G*, the function *find_smallest_color(G,i)* returns the color to be assigned to *i*.

The **output** of ***greedy_col_basic.py*** should be copied to a text file called "***output_greedy_col_basic_XXXXXX.txt***", where *XXXXXX* is your CIS-username. This text file with your output should be submitted together with your file *greedy_col_basic.py*.

Question A.2 (20%).
You are given a file *greedy_col_variation_incomplete.py*. This file has three (incomplete) functions *find_next_vertex(G)*, *find_smallest_color(G,i)* and *greedy(G)*. Complete these three functions and save your code in a file that is called *greedy_col_variation.py* such that, when you run this new file:

- it visits the vertices of the input graph in such an order that always the next visited node is adjacent to at least one previously visited node,
- among all such vertices, the algorithm visits at every step the smallest one (i.e. the vertex that is labeled with the smallest integer),
- at every step it assigns (in a greedy fashion) to the currently visited vertex the smallest possible color (where every color is a positive integer 1, 2, 3, …) such that no two adjacent nodes receive the same color, and finally
- it outputs the constructed coloring and the number *kmax* of the distinct colors in this coloring (using the printing commands that are given within the function *greedy(G)* in the file *greedy_col_variation_incomplete.py*).

At every iteration of the algorithm, the function *find_next_vertex(G)* computes and returns the next visited node. Furthermore, each time the algorithm visits a new node *i* of a graph *G*, the function *find_smallest_color(G,i)* returns the color to be assigned to *i*. Note that the function *find_smallest_color(G,i)* is exactly the same as in Question A.1.

The **output** of ***greedy_col_variation.py*** should be copied to a text file called "***output_greedy_col_variation_XXXXXX.txt***", where *XXXXXX* is your CIS-username. This text file with your output should be submitted together with your file *greedy_col_variation.py*.

PART B (60%)

You are given the files *graph6.py, graph7.py, graph8.py, graph9.py* and *graph10.py*. Each of these files contains a function *Graph()* which returns an input graph. Furthermore, you are given the file *depth_first.py*, which includes an implementation of the "Depth-First Search (DFS)" algorithm for traversing a graph. In particular, *depth_first.py* contains only one recursive function *dfs(G,u)*, which is called recursively to continue the DFS-traversing of the graph *G*, starting at node *u*.

After the code for the function *dfs(G,u)*, you can find (in the main part of *depth_first.py*) the calls of this function for all of the graphs returned by the files *graph6.py*, …, *graph10.py*. Therefore, if you run the file *depth_first.py* (for example by pressing F5), you see the output of the function *dfs()* for each of these 5 graphs, each time for *u*=1.

Question B.1 (20%).
You are given a file *breadth_first_incomplete.py*. This file has one (incomplete) function *bfs(G,a,b)*, where *G* is the input graph and *a,b* are two given nodes of *G*. Complete this function and save your code in a file that is called *breadth_first.py* such that, when you run this new file:

- it performs a Breadth-First Search (BFS), starting from node *a* and ending when it reaches node *b*,
- it outputs the *distance* (i.e. the length of a shortest path) between the given pairs of nodes for the 5 input graphs (as specified in the main part of the file *breadth_first_incomplete.py*).

The **output** of ***breadth_first.py*** should be copied to a text file called "***output_breadth_first_XXXXXX.txt***", where *XXXXXX* is your CIS-username. This text file with your output should be submitted together with your file *breadth_first.py*.

Question B.2 (25%).
You are given a file *depth_first_pair_nodes_incomplete.py*. This file has one (incomplete) function *dfs(G,a,b,u)*, where *G* is the input graph and *a,b* are two given nodes of *G*. This function *dfs(G,a,b,u)* is called recursively to continue the DFS-traversing of *G*, starting at node *u*. Complete this function and save your code in a file that is called *depth_first_pair_nodes.py* such that, when you run this new file:

- it performs a Depth-First Search (DFS), starting from node *a* and ending when it reaches node *b*,
- at every step, among the neighbors of the currently visited vertex, the algorithm chooses the smallest one (i.e. the vertex that is labeled with the smallest integer) to continue the exploration from it,

- in addition, it adds a label to each of the visited nodes, such that if a vertex v receives the label *i*, then this Depth-First Search has reached vertex *v* with a path of length *i* (starting form *a*); note that the label of vertex *a* is 0.

The **output** of ***depth_first_pair_nodes.py*** should be copied to a text file called "***output_depth_first_pair_nodes_XXXXXX.txt***". This text file with your output should be submitted together with your file *depth_first_pair_nodes.py*.

Hint: Recall that in the given file *depth_first.py* we used the integer variable *visited_counter* in order to count how many nodes we have visited so far (we stop our DFS when we have visited all nodes of the graph, i.e. when *visited_counter* = *n*). However, in the file *depth_first_pair_nodes.py* we do not need such a counter: we now stop our DFS when we have visited the specific node *b* (i.e. we do not necessarily need to visit all nodes of the graph).

Question B.3 (15%).

You are given a file *diameter_incomplete.py*. This file has one (incomplete) function *max_distance(G)*, where *G* is the input graph. After function *max_distance(G)*, you can find (in the main part of *diameter_incomplete.py*) the calls of this function for all of the graphs returned by the files *graph6.py*, …, *graph10.py*. Complete the function *max_distance(G)* and save your code in a file that is called *diameter.py* such that, when you run this new file:

- it outputs the greatest distance between any pair of vertices in the input graph (this is also called the *diameter* of the input graph).

The **output** of ***diameter.py*** should be copied to a text file called ***"output_diameter_XXXXXX.txt"***, where *XXXXXX* is your CIS-username. This text file with your output should be submitted together with your file *diameter.py*.

<u>In summary, you are expected to submit the following files:</u>

- *greedy_col_basic.py* (Question A.1)
- *output_greedy_col_basic_XXXXXX.txt* (Question A.1)
- *greedy_col_variation.py* (Question A.2)
- *output_greedy_col_variation_XXXXXX.txt* (Question A.2)
- *breadth_first.py* (Question B.1)
- *output_breadth_first_XXXXXX.txt* (Question B.1)
- *depth_first_pair_nodes.py* (Question B.2)
- *output_depth_first_pair_nodes_XXXXXX.txt* (Question B.2)
- *random_distance.py* (Question B.3)
- *output_random_distance_XXXXXX.txt* (Question B.3)