Colorings

The purpose of this project is for you to write code that tests whether or not a given coloring is proper (this includes both vertex colorings and edge colorings). In addition, you will create a function that determines whether or not a given graph has chromatic number at most 3.

Note that when you are outputing a coloring, the order does not matter. For example, {"A" : 1, "B" : 2} will be considered the same as {"B" : 2, "A" : 1}. The ONLY IMPORT ALLOWED is 'copy', and you are ONLY allowed to use the copy.deepcopy() method from this package. All of your code should be 'from scratch.'

Objectives

You are to write a function `is_proper(graph,color)` that has two inputs: one a graph, and the other a labelling of the vertices, and determines whether or not the labelling is a proper vertex-coloring of the given graph. In other words, return the Boolean value True if it is, and False if it is not.

- `is_proper`({"A" : ["B", "C"], "B" : ["A", "C"], "C" : ["A", "B"]}, {"A" : 1, "B" : 2, "C" : 3}) should return True
- `is_proper`({"A" : ["B", "C"], "B" : ["A", "C"], "C" : ["A", "B"]}, {"A" : 1, "B" : 1, "C" : 3}) should return False

Write a function `three_color(graph)` that takes in a graph as its input, and then returns all possible vertex-colorings (NOT necessarily proper!) of that graph.

- `three_color`({"A" : ["B"], "B" : ["A"]}) should return {"A" : 1, "B" : 1}, {"A" : 1, "B" : 2}, {"A" : 1, "B" : 3}, {"A" : 2, "B" : 1}, {"A" : 2, "B" : 2}, {"A" : 2, "B" : 3}, {"A" : 3, "B" : 1}, {"A" : 3, "B" : 2}, {"A" : 3, "B" : 3}

Write a function `is_three_color(graph)` that takes in a graph as its input, and then determines whether or not the chromatic number of the graph is at most three. In other words, it will return the Boolean value True if it is, and False if it is not.

- `is_three_color`({"A" : ["B", "C"], "B" : ["A", "C"], "C" : ["A", "B"]}) should return True
- `is_three_color`({"A" : ["B", "C", "D"], "B" : ["A", "C", "D"], "C" : ["A", "B", "D"], "D" : ["A", "B", "C"]}) should return False

Write a function `is_proper_edge(graph)` that takes in a weighted graph (remember that this just a labelling of the edges) and then determines whether or not the labelling is a proper edge-coloring. In other words, it will return the Boolean value True if it is, and False if it is not.

- `is_proper_edge`({"A" : [["B", 1], ["C", 2]], "B" : [["A", 1], ["C", 3]], "C" : [["A", 2], ["B", 3]]}) should return True
- `is_proper_edge`({"A" : [["B", 1], ["C", 2]], "B" : [["A", 1], ["C", 2]], "C" : [["A", 2], ["B", 2]]}) should return False

Write a function `greedy(graph, order)` that takes in two inputs, one a graph and the other an ordering of the vertices, and returns the proper vertex-coloring produced by the greedy algorithm.

- `greedy`({"A" : ["B", "C"], "B" : ["A"], "C" : ["A"]}, ["A", "B", "C"]) should return {"A" : 1, "B" : 2, "C" : 2}.
- `greedy`({"A" : ["B"], "B" : ["A", "C"], "C" : ["B", "D"], "D" : ["C"]}, ["A", "D", "B", "C"]) should return {"A" : 1, "B" : 2, "C" : 3, "D" : 1}.

Grading Rubric

Your functions will be tested using a collection of pre-made test cases that I will create. Your grade will be based on how often your code produces correct results and on the quality of the descriptions that you provide for your functions. Full credit will be given to a notebook whose functions work 100% of the time. Also each function has a description that clearly explains how the function works.