Spanning Trees

The purpose of this project is for you to write codes that find minimum and maximum spanning trees using both Krukal's and Prim's alorithms. 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 `edge_get(graph)` that takes in a weighted graph as its input and returns the list of all the edges of the graph in non-decreasing order. Note that the order of the endpoints of each edge does not matter (so [“*A*”, “*B*”] is the same as [“*B*”, “*A*”]).

- `edge_get(`{“*A*” : [[“*B*”, 10], [“*D*”, 5]], “*B*” : [[“*A*”, 10], [“*C*”, 5]], “*C*” : [[“*B*”, 5], [“*D*”, 15], “*D*” : [[“*C*”, 15], [“*A*”, 5]]}`)` should return [[“*A*”, “*D*”], [“*B*”, “*C*”], [“*A*”, “*B*”], [“*C*”, “*D*”]] (or [[“*B*”, “*C*”], [“*A*”, “*D*”], [“*A*”, “*B*”], [“*C*”, “*D*”]]

You are to write a function `list_join(lst, elt1,elt2)` that takes in three inputs: a main list whose elements are lists, and then two elements that each belong in exactly one of the lists of the main list. This method is then to return a new main list of lists, in which the list containing elt1 and the list containing elt2 are merged if different and unaffected otherwise.

- `list_join(`[[“*A*”, “*B*”], [“*C*”], [“*D*”], [“*E*”, “*F*”]], “*A*”, “*D*”`)` should return [[“*A*”, “*B*”, “*D*”], [“*C*”], [“*E*”, “*F*”]]
- `list_join(`[[“*A*”, “*B*”], [“*C*”], [“*D*”], [“*E*”, “*F*”]], “*A*”, “*B*”`)` should return [[“*A*”, “*B*”], [“*C*”], [“*D*”], [“*E*”, “*F*”]]

You are to write a function `min_kruskal(graph)` that takes in a weighted graph as its input and returns a list of edges that make up a minimum spanning tree IN THE ORDER obtained via Kruskal's algorithm. The edges should be in the form of lists as well.

- `min_kruskal(`{“*A*” : [[“*B*”, 10], [“*D*”, 5]], “*B*” : [[“*A*”, 10], [“*C*”, 5]], “*C*” : [[“*B*”, 5], [“*D*”, 15], “*D*” : [[“*C*”, 15], [“*A*”, 5]]}`)` should return [[“*A*”, “*D*”], [“*B*”, “*C*”], [“*A*”, “*B*”]] (or [[“*B*”, “*C*”], [“*A*”, “*D*”], [“*A*”, “*B*”]])

You are to write a function `min_prim(graph)` that takes in a weighted graph as its input and returns a list of edges that make up a minimum spanning tree IN THE ORDER obtained via Prim's algorithm. The edges should be in the form of lists as well. Note that “*A*” will always be considered the source/root.

- `min_prim(`{“*A*” : [[“*B*”, 10], [“*D*”, 5]], “*B*” : [[“*A*”, 10], [“*C*”, 5]], “*C*” : [[“*B*”, 5], [“*D*”, 15], “*D*” : [[“*C*”, 15], [“*A*”, 5]]}`)` should return [[“*A*”, “*D*”], [“*A*”, “*B*”], [“*B*”, “*C*”]]

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.