Digraphs and Relations

The purpose of this project is for you to write methods that determine the properties of a given relation. In addition, given a directed graph you will construct its transitive closure. 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_reflex(ground, relation)` that takes in two inputs: the first is the ground set, and the second is the relation. It should then determine whether the given relation is reflexive.

- `is_reflex(`[“*A*”, “*B*”, “*C*”, “*D*”, “*E*”]`,` [[“*A*”, “*A*”], [“*A*”, “*D*”], [“*B*”, “*C*”], [“*B*”, “*D*”], [“*C*”, “*E*”], [“*D*”, “*A*”], [“*E*”, “*E*”]]`)` should return False.
- `is_reflex(`[“*A*”, “*B*”, “*C*”]`,` [[“*A*”, “*A*”], [“*A*”, “*B*”], [“*A*”, “*C*”], [“*B*”, “*B*”], [“*B*”, “*A*”], [“*C*”, “*C*”], [“*C*”, “*A*”]]`)` should return True.

You are to write a function `is_sym(ground, relation)` that takes in two inputs: the first is the ground set, and the second is the relation. It should then determine whether the given relation is symmetric.

- `is_sym(`[“*A*”, “*B*”, “*C*”, “*D*”, “*E*”]`,` [[“*A*”, “*A*”], [“*A*”, “*D*”], [“*B*”, “*C*”], [“*B*”, “*D*”], [“*C*”, “*E*”], [“*D*”, “*A*”], [“*E*”, “*E*”]]`)` should return False.
- `is_sym(`[“*A*”, “*B*”, “*C*”]`,` [[“*A*”, “*A*”], [“*A*”, “*B*”], [“*A*”, “*C*”], [“*B*”, “*B*”], [“*B*”, “*A*”], [“*C*”, “*C*”], [“*C*”, “*A*”]]`)` should return True.

You are to write a function `is_antisym(ground, relation)` that takes in two inputs: the first is the ground set, and the second is the relation. It should then determine whether the given relation is antisymmetric.

- `is_antisym(`[“*A*”, “*B*”, “*C*”, “*D*”, “*E*”]`,` [[“*A*”, “*A*”], [“*A*”, “*D*”], [“*B*”, “*C*”], [“*B*”, “*D*”], [“*C*”, “*E*”], [“*D*”, “*A*”], [“*E*”, “*E*”]]`)` should return False.
- `is_antisym(`[“*A*”, “*B*”, “*C*”, “*D*”]`,` [[“*A*”, “*A*”], [“*A*”, “*B*”], [“*A*”, “*C*”], [“*A*”, “*D*”][“*B*”, “*D*”], [“*C*”, “*D*”], [“*C*”, “*C*”]]`)` should return True.

You are to write a function `is_trans(ground, relation)` that takes in two inputs: the first is the ground set, and the second is the relation. It should then determine whether the given relation is transitive.

- `is_trans(`[“*A*”, “*B*”, “*C*”, “*D*”, “*E*”]`,` [[“*A*”, “*A*”], [“*A*”, “*D*”], [“*B*”, “*C*”], [“*B*”, “*D*”], [“*C*”, “*E*”], [“*D*”, “*A*”], [“*E*”, “*E*”]]`)` should return False.
- `is_trans(`[“*A*”, “*B*”, “*C*”, “*D*”]`,` [[“*A*”, “*A*”], [“*A*”, “*B*”], [“*A*”, “*C*”], [“*A*”, “*D*”][“*B*”, “*D*”], [“*C*”, “*D*”], [“*C*”, “*C*”]]`)` should return True.

You are to write a function `trans_clos(digraph)` that takes in a directed graph as its input and returns the directed graph that is its transitive closure.

- `trans_clos(`{“*A*” : [“*B*”], “*B*” : [“*C*”], “*C*” : [“*B*”], “*D*” : [“*A*”, “*C*”]}`)` should return {“*A*” : [“*B*”, “*C*”], “*B*” : [“*B*”, “*C*”], “*C*” : [“*B*”, “*C*”], “*D*” : [“*A*”, “*B*”, “*C*”]}
- `trans_clos(`{“*A*” : [“*B*”, “*C*”], “*B*” : [“*A*”], “*C*” : [“*A*”]}`)` should return {“*A*” : [“*A*”, “*B*”, “*C*”], “*B*” : [“*A*”, “*B*”, “*C*”], “*C*” : [“*A*”, “*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.