Week 11

MINIMUM SPANNING TREE

Topic covered:

- Data structure for disjoint sets
- Kruskal's algorithm

Supplemental materials

- Class lecture
- disjointsets3.py
- mst_testcases.zip containing the edge list of a simple test graph.
- connectedCheck-template.py
- Example_Python3.py containing example basic codes.

Goals

a) Learn to use disjointsets3.py effectively
b) Develop a Python 3 program of Kruskal's algorithm.

1) You are expected to *self-learn* the following skills about Python 3. *There are numerous materials for these topics online* (www.w3schools.com is a good resource). However, for a quick example, you may also study them from supplemental Example_Python3.py file.
   - Python tuple
   - Sorting a Python list of objects

2) Study the disjoint-set data structure in disjointsets3.py from the provided example code at the end of the program file.
   - The argument required for initializing the disjoint sets is n, the number of sets to begin with.
   - Set identifier will automatically be 0 to n-1
   - Study the result of "findset" and "union" operations

3) Write a program that takes, as input, an undirected graph in the form of edge list. Then the program verifies whether the graph is "connected".
   Hint If the graph is connected, all the vertices can be united down to one set.

4) Test your program on the test cases provided in mst_testcases.zip. The code for reading the edge list into your program is provided in connectedCheck-template.py.

   Modify the input graphs as your prefer (save them as new data files) so that you can check whether your program works correctly for both "connect" and "not connected" cases.

5) If you have passed steps 1 and 4 above, you would have adequate skills to write an MST program that implements Kruskal's algorithm. And you may proceed so. However, if not, go back to step 1 and/or 2 until you this condition is satisfied.

The only required output is the total cost of the MST.

6) Test your program using the provided test cases.