Fall 2017/MA 397 HW 4: Trees and their Algorithms

Remember that, although you are encouraged to work together, all of your write-ups must be your own (no copying someone else's solution - not even with minor wording changes.) **List the names of everyone you worked with on the HW!** You are encouraged to **not** look online for solutions - your time is better spent wrestling with the proof yourself or getting help from the professor, than squandering it online.

1. Reading

- Read Sections 2.1 - 2.3 of the text. You may skip unassigned problems. For now you may also skip the section on "graphic sequences".
- Memorize the definition of **acyclic**, **forest**, **tree**, **leaf**, **distance**, **diameter**, **spanning tree**, **Wiener index**, **Prüfer code**, **in-tree** and **out-tree**, **rooted tree**, **descendents**.
- Study the proofs of Theorems/Propositions 2.1.4, 2.1.8, 2.1.14, 2.2.3, 2.2.4, 2.2.8, 2.2.28, 2.3.7, 2.3.15. We'll go over some of these in class and see below for some exercises associated with some of them.

2. To do

Reading Comprehension (your answers need not be long):

(1) In the proof of Theorem 2.1.4, explain the proof of the implication C implies A.

(2) In the proof of Proposition 2.1.8, what kind of proof is used? Can you find a different proof (perhaps using edge contractions?) Explain how we can show the inequality to be sharp.

(3) Give a 3- 5 point outline of the proof of Theorem 2.1.14.

(4) What is a Prüfer code and how can these codes be used to show that there are n^{n-2} trees with n vertices?

(5) Make up your own example of a connected graph with 8 vertices and use Proposition 2.2.8 to count the number of spanning trees. Substantial extra-credit for writing a computer program to do it for you!

(6) Give a brief explanation of what a minimum spanning tree is, what Kruskal's algorithm is and why it works. Substantial extra-credit for writing a computer program implementing it!

(7) Summarize Dijkstra's algorithm, what it's used for, and why it works. Also explain the connection to breadth first searches.

Problems:

(1) Do problems 2.1.1 - 2.1.4.

(2) Do problem 2.1.12.

(3) Do problem 2.1.14.

(4) Do problem 2.1.42

(5) Do problem 2.2.1 and 2.2.2.

(6) Do problem 2.2.8, but you only need to provide one proof for each.

(7) Do problem 2.3.1, 2.3.3

(8) Do problem 2.3.5

(9) Do problem 2.3.10.

Hint: Let T be the tree produced by Prim's algorithm and let T^* be an optimal spanning tree which agrees with T for the first k steps and disagrees at the $k+1$st step. Show how to create an optimal spanning tree (in particular, it has the same weight as T^*) which agrees with T for the first $(k+1)$st steps.