5 Greedy Problems: Due Weds May 10 in Lab

1. Does Prim's algorithm always work correctly on a connected graph with negative edge weights? yes/no?

   If not give a counterexample. If it does justify your answer.

2. Dr. Genius suggests the following algorithm for finding the shortest path from node s to node t in a graph with some negative edge weights. Add a large constant to each edge weight (same constant for every edge) so that all the weights in the graph become positive. Then on this new graph run Dijkstra's algorithm starting at node s until you find the shortest path to t. Does this algorithm always give the shortest path tree? yes/no?

   If not give a counterexample. If it does justify your answer.

3. Suppose you are given n ropes of different lengths, you need to connect these ropes into one rope. The cost to connect two ropes is equal to sum of their lengths. You want to find the minimum cost way to do this. For example: Suppose you have three ropes with lengths 2, 5, and 8. If you chose first to connect the length 5 and 8 ropes, then connect the length 2 and 13 ropes, the total cost would be (5 + 8) + (13 + 2) = 28. However, if you first chose to connect the length 2 and 5 ropes, then the length 7 and 8 ropes, the total cost would be (2 + 5) + (7 + 8) = 22 (which happens to be optimal).

   Specify with pseudo code a greedy algorithm to connect the ropes with minimum cost.

   Prove you algorithm always finds the least cost solution if the lengths of the ropes are distinct.

   Analyze your algorithms complexity.

4. There is a long straight country road with expensive houses scattered along it. The houses are owned by affluent stock traders who require cell phone service. You consult for the company that needs to provide the cell service to every house without exception. The towers only have a range of four miles. So you want to place the cell phone towers at locations along the road so that no house is more than four miles from the nearest cell phone tower. You know exact mileage along the road where each house is located. Design a greedy algorithm that will determine the set of locations for the cell towers that requires the fewest cell tower. Denote the location of the i^{th} house as M_i .
   Give an efficient algorithm.
   Specify (pseudo code) an efficient greedy algorithm to achieve this goal with the fewest cell towers.

   Prove you algorithm always finds the optimal solution.

   Analyze your algorithms complexity.