EECS 340, Breakout Session Notes, December 2, 2019

1. The quiz is different this time. Tell the students they have 15 minutes to work with each other to solve the quiz. They are not use computers or phones or consult their notes. They also are not to write anything on the quiz papers. They can write on boards, other paper, etc. But, they will have to put these away when taking the quiz.

2. Now they put everything away and write the quiz for 15 minutes.

3. Here is the final problem to give them:

   A fraternity has a lot of empty drink bottles from all the parties. You, as a new pledge, are given a task. Which algorithm task should you use, and which is NP-hard?

   (a) Place the bottles in a giant circle where adjacent bottles are brands of drink that were served at the same party.

   (b) Fill a bin that can hold up to 10 pounds with bottles to maximize the amount of money back on the deposit. The bottles are different weights and have different amounts of deposit.

   (c) Assign the bottles to different fraternity members to take to recycling such that the bottles are divided as evenly as possible and a member is only assigned bottles of a brand that that member brought to some party.

   (d) Assume the bottles all have twist caps, but they have slightly different sizes. They removed the labels from all the bottles, throw the caps in a big pile, and ask you to correctly recap every bottle.

4. (a) is NP-hard. Take a graph that you want to find a Hamilton Cycle on. Make every vertex its own brand of drink, and have each edge represent a party where both brands were served.

5. (b) is dynamic programming. This can be directly reduced to knapsack.

6. (c) is flow. Link every bottle to a member that brought that brand. Give capacity n/k to each edge from a member to t (where n is the # of bottles and k is the # of fraternity brothers). If that flow is not size n (all bottles assigned), increase the capacity of the brother $\rightarrow t$ edges by 1 until a flow of size n is achievable. Since there are at most nk edges, and we have to run the flow at most n times, we get $O(n^3k)$.

7. (d) is divide and conquer. Take a cap and divide all the bottles into 3 groups: fits the cap, the cap is too small and the cap is too large. Then take a bottle that matches this cap and divide all the caps equivalently. Then you can recurse on each half to get expected time $O(n \log n)$.