EECS 340, Breakout Session Notes, October 12, 2018

1. Begin by handing out the quiz. The students are to work alone with no notes, phones, etc. Collect the quiz at 3:40 so that you have 30 minutes to do the exercise below.

2. Place the students into groups of 3 and give them this problem (taken from the International Programming Competition): John is going fishing. He has h hours available, and there are n lakes available all reachable on a single, one-way road. John starts at lake 1 and can finish at any lake he wants. He can only travel from one lake to the next one, but he does not have to stop at a lake unless he wishes to. For each $i = 1, \dots, n-1$, t_i is the number of minutes it takes to travel from lake i to lake $i+1$. All times are in 5-minute intervals. For each lake i, f_i is the number of fish he expects to catch in his first 5 minutes of fishing at the lake, ($f_i \geq 0$). Each 5 minutes of fishing decreases the number of fish expected to be caught in the next 5 minute interval by a constant amount d_i, $d_i \geq 0$. If the number of fish expected to be caught in a 5-minute interval is less than d_i then no fish will be caught at this lake after this interval ends.

   Design an algorithm to plan John's trip in order to maximize the expected number of fish that he will catch.

3. Let them know that this is a dynamic programming question. They need to think about what they need to store in the table, what choices they have to make, and what information they need to make that choice.

   What are the choices that John must make? At each 5 minute interval, John must decide if he will stay at this lake or move on to the next lake. These are the only two choices.

   To get an dynamic programming algorithm they must do the following:

   - Show that for each choice they want to do the optimal on the resulting subproblem.
   - Give a recurrence relation.
   - Prove the recurrence relation is correct.
   - Show how to extract the solution from the table.
   - Give the running time of the algorithm.

   They should now try to complete these tasks.

4. Here is a likely recurrence. If they do not come up with it, or something like it, show it to them so they can see that dynamic programming requires more than "optimize each subproblem and take the max". We actually have to think about how to combine the optimal subproblems.

   Let $T[l,t]$ be the maximum number of fish caught if we are at lake l and time t. So

   $$T[l,t] = max\{T[l-1,t-1], T[l,t-1] + \text{ fish caught at this lake}\}$$

   If we maximize each subproblem, do we get the maximum number of fish? In order words, if we are at lake l at time t, do we want to maximize the number of fish caught before this time?

   No! The amount of fish we catch at this lake depends on how long we have been here. If we just arrived, we will catch more fish, and so it may make sense to not maximize the number of fish if we can get to this lake later. We need to adjust how we are thinking about the problem.

   However, we can change the argument: If we are at lake l at time t, and we have been at this lake for d intervals already, then we do want to maximize the amount of fish caught before. That is because the amount of fish we can catch at this moment is fixed, and if we catch less than the maximum up to this point, we can replace that with a plan that catches the maximum and we catch more fish. That tells us we need a 3-dimensional table!

5. Here is a correct recurrence

   Let $T[t, l, d]$ be the number of fish caught if John is at lake l at time interval t, and has been at this lake for d intervals. For $T[t, l, 0]$ we want to come from lake $l-1$ at t_l intervals ago, but we don't know how we should have stayed at that lake, so we look at all possibilities. For $T[t, l, d]$ with $d > 0$ we know we stayed at the lake.

$$T[t, l, 0] = \max_{d \in \{0, \ldots, 12h\}} \{T[t - t_{l-1}, l-1, d]\}$$
$$T[t, l, d] = T[t, l, d-1] + f_l - \max\{f_l, d_l * (d-1)\}$$

   We need some initial values in our table. We both need to set that we are starting at lake 1 at time interval 0. We also need to prevent John going to any lake faster than he can travel there. We can either have if statements in the loops to prevent illegal values, or we can put error values into the table.

$$T[0, 1, 0] = 0$$
$$T[t, l, d] = -\infty \quad \text{if } t < \sum_{i=1}^{l-1} t_i$$

6. The proof of correctness is as follows:

   Assume $T[x, l-1, d]$ stores the maximum number of fish caught at time x, lake $l-1$ and y intervals spent at lake $l-1$ for all $x < t$ and all d. The maximum number of fish at time t, lake l, and interval 0 is the maximum we could have achieved from the previous lake. This is $\max\{T[t - t_{l-1}, l-1, d]\}$, and by the I.H. the table stores the maximum value so $T[t, l, 0]$ stores the maximum value.

   Likewise, assume $T[t-1, l, d-1]$ stores the maximum number of fish we can catch by time $t-1$ if we are at lake t and we have been nere for $d-1$ intervals. If we are at time t, lake l and inteval d for $d > 0$, we will catch $f_l - (d-1)d_l$ fish this interval, and to maximize the total number of fish, we need to maximize the fish caught at time $t-1$, lake l, and duration $d-1$. By the induction hypothesis, this value is stored in $T[t-1, l, d-1]$.

7. How do we get the solution out of the table?

   The maximum number of fish John can expect to catch is found by searching the entire table for the largest value. Then, we can work backwards from that value. Suppose $T[t, l, d]$ is the largest value. Then we know John ended his day spending d intervals at lake l. That means John arrived at lake l at time $t-d$, and we record that John must leave lake $l-1$ at time $t - d - t_l$. At what time did he arrive at lake $l-1$? Let $t' = t - d$, the time that John arrived at lake l. We look at $T[t' - t_{l-1}, l-1, d^*]$ for all possible d^* and find the one that has the same value as $T[t', l, 0]$. That tells us how long we were at lake $l-1$. Repeating this process will give us the schedule for John.

8. What is the running time?

   The table is size $n \times 12h \times 12h$ (because there are 12 5-minute intervals in an hour) and it takes up to $12h$ steps to fill in each table value. Thus the running time is $\Theta(nh^3)$.