***Computer Science 180 Programming Project***

This will be a multiway choice problem. We need to find locations to break up the lectures to get minimal dissatisfaction. We will solve it with a similar method as the matrix problem in class. We define all topics as ni, where i is the topic number. We define dissatisfaction(i,n) to be the minimum dissatisfaction for a lecture from topic i to topic n. The dissatisfaction for any given subset n will be DIS(n) = dissatisfaction(i,n) + DIS(i – 1). Then, our problem is essentially, for each subset of topics, we want to find the minimum dissatisfaction.

We will create some two dimensional array of to represent the topics presented in a lecture. We define the array to hold the amount of dissatisfaction for a given set of topics within a single lecture, where rows will be the starting topic and columns will be the ending topic. To fill this in, the base case on the diagonal is maxLectureLength – topicLength[i], which will calculate the amount of leftover time after the first topic. Then, for each spot below, it will be the timeLeft – topicLength[row]. Then, we take the same array and iterate through every position. If it is negative, we know it is impossible to present, so we just set it to -1 to make the array easier to read. Then, if it is in the last row, this means it is the last lecture, so we set the dissatisfaction to zero. If there is between zero and five minutes left over, there is also zero dissatisfaction. If there is over five minutes, we set the dissatisfaction to (timeLeftOver – 5)4, as defined in the spec. This goes through every position once, so it has O(2n2) = O(n2) run time.

We will now use a recurrence to find the minimum dissatisfaction for given subsets. Since it is a multiway choice solution, we will use the recurrence, which iterates through all possibilities of where to split up topics, and returns the minimal dissatisfaction. We choose to store the dissatisfaction in a numTopics length array, which we simply call dissatisfaction. The optimal solution for each topic is stored in another numTopics length array, which we call optimal. Now, we write our full recurrence for finding the minimum in the multiway choice, which in turn gives us our iterative algorithm which will heavily use the recurrence. This is an O(n2) operation since it goes through numTopics at most twice in a nested for loop and the calculations within can be done in O(1) run time.

for all j = 0 … numTopics

if i equals 0 /base case

dissatisfaction[0] = lectures[0][0]

optimal[0] = 0

else

//using optimal[j] =

for i = 0 … j

if lectures[j][i] != 0 //valid topic set

if lectures [j][i] + dissatisfaction[i-1] < currentMin

currentMin = lectures [j][i] + dissatisfaction[i-1]

minVal = i

dissatisfaction[j] = currentMin;

optimal[j] = minVal

Finally, we print out the solution. We start at the last position of optimal and using the optimal topic value, move the current position to the optimal topic value. We continually do this until we reach topic 0. Since we want to output the lectures in the correct time order, we first place the outputs into an array and then output them in reverse order. We could count the number of lectures we need during these iterations, and the total dissatisfaction is stored at dissatisfaction[numTopics – 1]. It has an O(n) run time since it traverse optimal, which will take at most n iterations, and the calculations within are O(1).

We know our algorithm is correct because we can prove our recurrence relation. Our recurrence relation is essentially OPT(j) = , which uses the dissatisfaction equation DIS(n) = dissatisfaction(i,n) + DIS(i – 1). The base case is for topic zero, which is clear since it would be the only topic in the only lecture possible. If we let DIS(n) equal the dissatisfaction, then the dissatisfaction for a given subset will be the dissatisfaction for a lecture with topics i through n (based on time left over), plus the dissatisfaction for the subset of topics 1 to i. Since we know how to find the dissatisfaction, we can use it to find the minimal dissatisfaction in a multiway choice. By finding the minimum dissatisfaction for all possible i < j, where j is the last lecture in a given subset, we can find the optimal choice for the starting topic given the last topic j in the subset. Since we prove the recurrence, we therefore know that our algorithm is correct.

The total run time of our algorithm is O(n2). The run time for filling in the lectures array is O(n2), the run time for the recurrence, or iterative algorithm is also O(n2), and the run time for outputting the results is O(n). The explanations for each of the sections’ run times are listed in their respective descriptions.

After thinking of the algorithm, there were a few minor problems while implementing the project. The first was doing the result output. Since the answers are given in reverse order, we have to put them in another array and then output them in the correct order afterwards. The second was when we have numbers that are larger than INT\_MAX. To circumvent this, we use doubles instead of ints and just in case the numbers get even larger, we set any number #INF (infinity) to DBL\_MAX.