CSE 3318 Lab Assignment 3

Due October 31

Goals:

1. Understanding of the dynamic programming solution to the (one room) weighted interval scheduling problem.

2. Understanding of the five steps for developing a dynamic programming solution.

Requirements:

1. Your task is to write a C program to solve the ***two*** room weighted interval scheduling problem by using dynamic programming. The first line of the input will be the number of intervals (`n`) and each of the remaining `n` lines have three non-negative integers (s_i, f_i, v_i) for the start time, finish time, and weight for an interval. `n` will not exceed 50. The intervals will be ordered by ascending finish time. You should echo the input.

2. You should print your dynamic programming table(s) with appropriate labeling.

3. Your solution should be formatted as:

   a. A line with just the number of intervals assigned to the first room.
   b. A line for each of the intervals assigned to the first room: start time, finish time, weight.
   c. A line with just the number of intervals assigned to the second room.
   d. A line for each of the intervals assigned to the second room: start time, finish time, weight.
   e. A line with the total weight that has been achieved

   and appear at the end of your output. No particular ordering of the intervals is required.

4. Submit your C program on Canvas by 5:00 p.m. on Tuesday, October 31 . One of the comment lines should include the compilation command used on OMEGA (5 point penalty for omitting this).

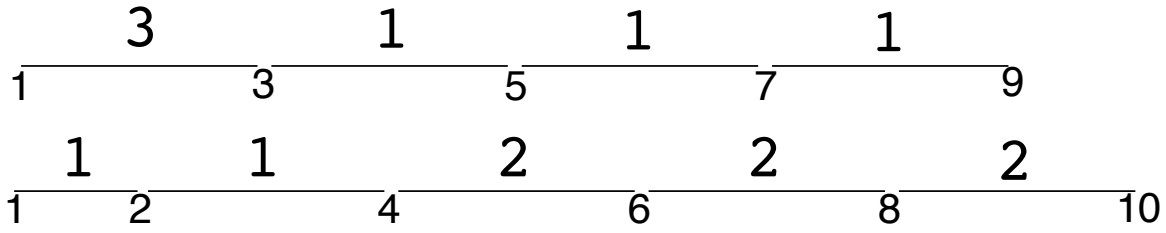
Getting Started:

1. The input should be read from standard input (which will be one of 1. keyboard typing, 2. a shell redirect (<) from a file, or 3. cut-and-paste). Do NOT prompt for a file name! Do not use `fgets()`.

2. All output should go to standard output.

3. Your solution ***must*** use dynamic programming to maximize the total weight of the intervals assigned to the two rooms. The intervals assigned to a room may not overlap (two intervals i and j overlap if either $s_i < s_j < f_i$ or $s_i < f_j < f_i$.).

4. Applying the simple (one room) solution in Notes 7.B to get a maximum solution for one room and then applying the same technique again to the leftover intervals (to assign intervals to the second room) is a ***non-optimal*** greedy technique ***that will receive zero points***.



5. The notion of rightmost preceding intervals is still useful in deriving the detailed cost function.

6. It is useful to have the cost function $M(i, j)$ = Total cost for optimal non-overlapping subsets for the two rooms. Room one may use the first i input intervals. Room two may use the first j input intervals. An interval may only be used once. Some observations:

   a. $M(i, j) = M(j, i)$ COST MATRIX IS SYMMETRIC

   b. $M(i, 0) = M(0, i) = M(i)$ from the one-room version in Notes 7.b. (You may reuse this code.)

   c. $M(i, i) = M(i, i - 1)$ DIAGONAL. INTERVAL i CAN ONLY BE IN 1 OF 2 ROOMS.

RANDOM CHOOSE ROOM 2 DOESN'T GET i

7. (The great point in understanding this assignment). Suppose you are attempting to compute $M(i, j)$ with $i > j$. The only issue is: Should interval i be placed in room one?

INPUT FILES: (ORDERED BY END OF INTERVALS)

- NUMBER OF ROWS (# OF INTERVALS)
- START FINISH VALUE
- :
- :

↑ END ORDERED

OUTPUT:

- INPUT
- DP TABLES
- # INTERVALS IN FIRST ROOM
- EACH FIRST ROOM INTERVAL
- # INTERVALS IN 2ND ROOM
- EACH 2nd ROOM INTERVAL
- TOTAL MONEY (VALUE)