# CSE 2320 Lab Assignment 3

Due July 20, 2011

## Goals:

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

## Requirements:

1. Use Java to implement a $\Theta(kn)$-time, $\Theta(k+n)$-space dynamic programming (not greedy!) algorithm to determine a ***minimum cost sequence*** of parking permits to cover $n$ not-necessarily-adjacent days you need to drive downtown. Each element of a solution sequence will be one of $k$ available permit types, each covering a different number of ***consecutive*** days at some cost.

   The $k+n+1$ input lines will be accessed using `System.in` (use `java.util.Scanner`). The first line of the input will be positive integers for $k$ and $n$. $k \le 10$ and $n \le 100$. The next $k$ lines will be pairs of positive integers for the permit types. The two values of a pair will be the number of days and cost, respectively. Note that the $k$ pairs will appear in strictly increasing order for the number of days and likewise for the costs. (For example, nobody will spend $20 for a three-day permit if a four-day permit is just $15.) Each of the remaining $n$ lines will contain an integer corresponding to a day you must park. These values appear in strictly increasing order.

   The output is 1) the table of subproblems, i.e. their cost and backtrace information, 2) the cost of the final solution, and 3) the sequence of permits needed and the range of days covered by each. The sequence may be output in reverse order. The last day for the last permit should be the last day you need to park. The start date for the first permit may be earlier than the first day you need to park.

2. Email your program to `hafizfahad@mavs.uta.edu` by 12:45 p.m. on July 20, 2011.

## Getting Started:

1. The left column gives an input instance and the right column gives a solution:

```
3 10
1 5
2 7
3 9
1
2
3
4
7
8
9
12
13
15
```

```
DP table:
prefixCost[0]=5 permitTypeUsed[0]=0
prefixCost[1]=7 permitTypeUsed[1]=1
prefixCost[2]=9 permitTypeUsed[2]=2
prefixCost[3]=14 permitTypeUsed[3]=0
prefixCost[4]=19 permitTypeUsed[4]=0
prefixCost[5]=21 permitTypeUsed[5]=1
prefixCost[6]=23 permitTypeUsed[6]=2
prefixCost[7]=28 permitTypeUsed[7]=0
prefixCost[8]=30 permitTypeUsed[8]=1
prefixCost[9]=35 permitTypeUsed[9]=0
Cost is 35
Permits used are:
Permit type 0 cost 5 begin 15 end 15
Permit type 1 cost 7 begin 12 end 13
Permit type 2 cost 9 begin 7 end 9
Permit type 0 cost 5 begin 4 end 4
Permit type 2 cost 9 begin 1 end 3
```

2. This problem is slightly similar to the weighted interval scheduling problem in Notes 7. In particular, an optimal solution is determined for every prefix of the input sequence of $n$ days.

3. Achieving $\Theta(kn)$ time requires maintaining a table with $k$ entries whose entry $i$ indicates the ***subscript*** of the ***latest*** input day such that a permit of type $i$ cannot cover ***both*** this day and the day whose subproblem is under consideration. It is convenient to initialize the entries of this table to `-1`.