# Dynamic Programming - Written Problems

1. Suppose you are given a row of $n$ doors $\{d_1, d_2, ..., d_n\}$, each of which can either be locked or unlocked. Your goal is to have exactly $S$ secured doors. The issue is that not all locked doors are secure. A door is secure if and only if:

    1. This door ($d_i$) is locked AND
    2. This door is the first door (i.e., $d_1$) OR The door to this one's left ($d_{i-1}$) is also locked

    Develop an algorithm that, given the number of doors $n$ and a value $S$, counts the total number of unique ways the doors can be locked to achieve EXACTLY $S$ secure doors.

2. You are going skiing this weekend and wish to ski once down every unique run the mountain has to offer. The mountain has $n$ runs given to you in a list called $R = \{r_1, r_2, ..., r_n\}$ where each $r_i$ is a positive integer denoting the number of minutes it will take to ski that run and get back up the lift to the top of the mountain. You want to ski each run in the minimum number of days possible. Each day has $L$ minutes of skiing available. Obviously, you must fit each run into a single day (you can't split a run over two days). You also wish to ski the runs in order.

    In addition to this, you have preferences regarding the amount of unused time at the end of each day. If, at the end of a day, you have $m$ minutes (e.g., only 10 mins) left in the day or less, than you are happy to stop skiing (you have a little time to get home, change, etc.). If, however, you have more than $m$ minutes left in the day and not enough time for another run, you'll feel like you wasted valuable time. We will model your time wasted dissatisfaction ($twd$) as follows:

    $$twd(t) = \begin{cases} 0 & t = 0 \\ -C & 1 \leq t \leq m \\ (t-m)^2 & \text{otherwise} \end{cases}$$

    Where $C$ is some constant. Develop an algorithm that minimizes the number of days needed to ski. If multiple optimal schedules exist, then pick the one that minimizes the sum of the $twd(t)$ values for all days.

3. Suppose you are given a rectangular board of size $2$ by $w \geq 1$ (the units here don't matter). You are also given an endless bag of dominoes, each of size $2$ by $1$. Write a program that accepts the integer $w$ as parameter, and returns the total number of unique ways the $2$ by $w$ board can be fully tiled. When we say *fully tiled* here, we mean that every square on the board is covered by dominoes. State the runtime of your algorithm. **See the next problem for some additional details.*

4. Now solve the exact same problem as above, except with a board of size $4$ by $w$. State the runtime of your algorithm. For example, if the input given is $w = 2$, then the solution is $5$, the following image shows all of the combinations of fully tiled boards of width $w = 2$: