

**Honor Code Notice.** This document is for exclusive use by Fall 2025 CS3100 students with Professor Bloomfield and Professor Floryan at The University of Virginia. Any student who references this document outside of that course during that semester (including any student who retakes the course in a different semester), or who shares this document with another student who is not in that course during that semester, or who in any way makes copies of this document (digital or physical) without consent of the instructors is **guilty of cheating**, and therefore subject to penalty according to the University of Virginia Honor Code.

For this problem set, you will be given two problems. For each, it can either be solved using a **Greedy Algorithm** or with **Dynamic Programming**. Read each problem and provide an algorithm for each. Part of the challenge this time around is that you do not know which problem should utilize which technique, so you should think carefully about which approach works best in each scenario.

**PROBLEM 1** *Skiing (again!)*

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  $R = \{r_1, r_2, \dots, r_n\}$  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) and because your favorite celebrity on TikTok recommended to, you 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.

**Solution:**

To find the minimum number of days, simply use the obvious greedy algorithm of fitting as many ski runs into your schedule as possible. This provides an upper bound for when we can stop any recursive DP solution (if greedy solution is  $X$  days, then stop if DP have reaches the  $X+1$  day).

We then need to minimize TMD. For each of the  $X$  days, there is some run that should be the last run of the day. Let  $Ti(i, j)$  be the sum of the minutes from  $r_i$  to  $r_j$ . Let  $Opt(i, d)$  be the optimal

satisfaction when skiing the first  $i$  runs in only  $d$  days.  $\text{Opt}(0,0) = 0$ . Then, for any arbitrary  $i$  and  $d$ :

$$\text{Opt}(i,d) = \min_{1 \leq x \leq i} (\text{twd}(\text{Ti}(x,i)) + \text{Opt}(i-x, d-1))$$

In other words, for all possible ways to plan the last day, find the  $\text{twd}$  of that day and then optimal ski the remaining runs in one fewer day. Solution is  $\text{Opt}(n,D)$  where  $D$  is number of days found in greedy solution.

## PROBLEM 2 Word Layouts

You are given an ordered list of  $n$  words (some text), and you want to lay them out in a document (the layout) in such a way that the lines are visually pleasing (the length of each line is approximately the same). The length of the  $i$ th word is  $w_i$ , that is the  $i$ th word takes up  $w_i$  spaces on its line. (For simplicity assume that there are exactly one space between each word.). The goal is to break this ordered list of words into lines, this is called a layout.

You can not reorder the words. The length of a line is the sum of the lengths of the words (plus the one space in between each) on that line. The ideal line length is  $L$ . No line may be longer than  $L$ , although it may be shorter. The penalty for having a line of length  $K < L$  is  $L - K$ . The total penalty of the layout is the **sum of all the line penalties**. Provide an algorithm that minimizes the **sum of all the line penalties**.

**Solution:** This is greedy and the greedy choice is simple: Put as many words as you can on the first line to get as close to  $L$  characters on that line as possible. Then, repeat for the next line, etc.

Rough proof outline: Consider the last word greedy puts on the first line (let's say it is word  $i$  for  $w_i$  spaces), and suppose f.s.o.c. that you should NOT put this word on the first line. Instead it goes on the second line. This creates an extra  $w_i$  penalty on the first line but saves  $w_i$  spaces on the second line (if solution is two lines). Trading these words and putting it back the way it was gains  $w_i$  penalty on line 2 but saves  $w_i$  penalty on line 1, keeping solution the same overall. The same logic holds recursively if line 2 pushes to line 3, etc.

A second case occurs if not putting word  $i$  on the first line causes the whole layout to add an extra line overall. In this case, the total penalty goes significantly up so the exchange makes the layout more optimal.