University of Toronto at Scarborough

**CSCC73 - Algorithm Design and Analysis**

# Assignment #5: Dynamic Programming 2

Due: Feb 25, 2023 at 11.59pm This exercise is worth 5% of your final grade.

**Warning:** Your electronic submission on MarkUs affirms that this exercise is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on Gradescope. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

1. Suppose that you have a computer that can process a certain amount of data each day and further that the more consecutive days that the computer runs, the worse it's performance. For each of $n$ days, you are given $x_i$ terabytes of data to process on day $i$. For each terabyte of of data processed you receive a fixed amount of revenue but any unprocessed data becomes unavailable at the end of the day (so you cannot process it the next day).

   Since you are limited by the capabilities of your computer which decreases with each passing day since the last reboot, you want to find a schedule to process the data and reboot the machine to maximize the revenue. For your computer you know the amount of data it can process $j$ days after the last reboot. So for example, after a reboot, the next day it can process $s_1$ terabytes of data. On the second day after reboot, it can process $s_2$ terabytes of data etc. We assume that $s_1 > s_2 > \ldots > s_n > 0$. To return to a processing power of $s_1$ you can reboot, but on that day, no data will be processed.

   **Example**.

   Suppose $n = 4$ and the values of $x_i$ and $s_i$ are given below:

   | | Day 1 | Day 2 | Day 3 | Day 4 |
   |---|---|---|---|---|
   | x | 10 | 1 | 7 | 7 |
   | s | 8 | 4 | 2 | 1 |

   For this example, the best schedule would be to reboot on day 2 since then $8+0+7+4 = 19$ terabytes would be processed whereas if no reboot occurred then only $8 + 1 + 2 + 1 = 12$ terabytes would have been processed.

   Formally, given a schedule of data $x_1, x_2, \ldots, x_n$ for $n$ days and efficiency of your computer $s_1, s_2, \ldots, s_n$ where $s_i$ is the processing power $j$ days *after* a reboot, give an efficient algorithm that takes the values and returns the number of terabytes processed by an optimal solution. Be sure to justify the correctness of your algorithm as discussed on piazza for dynamic problems and give the complexity of your algorithm.

2. Question 15 from the text, given below.

   Consider the problem faced by a stockbroker trying to sell a large number of shares of stock in a company whose stock price has been steadily falling in value. It is always hard to predict the right moment to sell stock but owning a lot of shares in a single company adds an extra complication: selling many shares in a single day will have an adverse effect on the price.

   Since future market prices, and the effect of large sales on these prices, are very hard to predict, brokerage firms use models of the market to help them make such decisions. In this problem, we will consider the following simple model. Suppose we need to sell $x$ shares of stock in a company, and suppose that we have an accurate model of the market: it predicts that the stock price will take the values $p_1, p_2, \ldots, p_n$ over the next $n$ days. Moreover, there is a function $f(\cdot)$ that predicts the effect of large sales: if we sell $y$ shares on a single day, it will permanently decrease the price by $f(y)$ from that day onward. So, if we sell $y_1$ shares on day 1, we obtain a price per share of $p_1 - f(y_1)$, for a total income of $y_1 \cdot (p_1 - f(y_1))$. Having sold $y_1$ shares on day 1, we can then sell $y_2$ shares on day 2 for a price per share of $p_2 - f(y_1) - f(y_2)$. This yields an additional income of of $y_2 \cdot (p_2 - f(y_1) - f(y_2))$. This process continues over all $n$ days.

Design an algorithm that takes the prices $p_1, \ldots, p_n$ and the function $f(\cdot)$ (written as a list of values $f(1), f(2), \ldots, f(x)$) and determines the best way to sell $x$ shares by day $n$. In other words, find natural numbers $y_1, y_2, \ldots, y_n$ so that $x = y_1 + \cdots + y_n$ and selling $y_i$ on day $i$, $1 \leq i \leq n$, maximizes the total income achievable. You should assume that the share value $p_i$ is monotone decreasing and $f(\cdot)$ is monotone increasing, ie., selling a larger number of shares causes a larger drop in the price. Your algorithm's running time can have a polynomial dependence on $n$ (the number of days), $x$ (number of shares) and $p_1$ (peak price of the stock). Be sure to justify the correctness of your algorithm as discussed on piazza for dynamic problems and give the complexity of your algorithm.

**Example**.

Let $n = 3$, $(p_1, p_2, p_3) = (90, 80, 40)$ and $f(y) = 1$ for $y \leq 40,000$ and $f(y) = 20$ for $y > 40,000$. Assume you start with $x = 100,000$ shares. Selling all of them on day 1 would yield a price of 70 per share. For a total income of $7,000,000$. On the other hand, selling $40,000$ shares on day 1 yeilds a price of 89 per share, and selling the remaining 60,000 shares on day 2 results in a price of 59 per share for a total income of 7,100,000.