School of Computer Science

Carleton University

Name: **A.M.** Problem Set: **5**

Course: **COMP 3804** Week: **1-9**

Consult [1, 2, 3]. The problem set consists of problems on Dynamic Programming.

Problem 1. *Assume that a taxi driver doesn't work for two consecutive days, i.e. after each working day, the driver needs a break of at least one day. For the next n successive days, the driver is provided with a chart showing the amount they can earn each day. Design an efficient algorithm to find this driver's optimal set of non-consecutive working days to maximize their earnings.*

As an example, suppose that the amounts that the driver can make (working on that day) for nine successive days are as follows:

Day #	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Earning ($)	*60*	*80*	*10*	*70*	*20*	*20*	*80*	*30*	*10*

Below, we list four scenarios and the corresponding earnings the driver can make if they work on the following sets of non-consecutive days.

Scenario	*Working days*	*Total Earnings*
A	*1,3,5,7,9*	$60 + 10 + 20 + 80 + 10 = 180$
B	*2,4,6,8*	$80 + 70 + 20 + 30 = 200$
C	*1,4,7,9*	$60 + 70 + 80 + 10 = 220$
D	*2,4,7,9*	$80 + 70 + 80 + 10 = 240$

Execute your algorithm on the above 9-day example to find an optimal schedule.

Problem 2. *Assume that you have a chocolate bar of length n inches. You have a satisfaction chart that indicates that if you eat a piece of length i inches, for $1 \leq i \leq n$, then you get a satisfaction of s_i. We can assume that all the quantities involved $(n, i, s_1, s_2, \ldots, s_n, ...)$ are positive integers. Using dynamic programming, find the best way to make pieces of the chocolate bar to get the maximum possible satisfaction when we want to consume the whole bar. What is the time complexity of the algorithm?*

As an example, suppose that the bar is 5-inches long, with the following satisfaction chart:

Length of Piece in inches (i)	*0*	*1*	*2*	*3*	*4*	*5*
Satisfaction s_i	*0*	*2*	*7*	*9*	*6*	*12*

Possible ways to make pieces of the 5-inch bar with its satisfaction values are as follows:

	Piece Sizes	*Satisfaction Value*		*Piece Sizes*	*Satisfaction value*
I.	$(1, 1, 1, 1, 1)$	$5 \times 2 = 10$	*II.*	$(2, 1, 1, 1)$	$7 + 4 \times 2 = 15$
III.	$(2, 2, 1)$	$2 \times 7 + 2 = 16$	*IV.*	$(3, 1, 1)$	$9 + 2 \times 2 = 13$
V.	$(3, 2)$	$9 + 7 = 16$	*VI.*	$(4, 1)$	$6 + 2 = 8$
VII.	(5)	12			

Observe that the maximum satisfaction achieved is 16 with the pieces of sizes $(2, 2, 1)$ or $(3, 2)$.

Problem 3. *Given two strings $X = x_1x_2 \cdots x_n$ and $Y = y_1y_2 \cdots y_m$, what are the minimum number of edit operations that are required to convert X to Y. The edit distance (also called the Levenshtein distance) allows the insertion, deletion and substitution of symbols in the strings. Design a dynamic programming algorithm to find the minimum number of insertions, deletions, and substitutions that can be done to transform X to Y. Verify that three edit operations are sufficient to change X=SNOWY to Y=SUNNY. How many edit operations can transform TORONTO into HURONTARIO?*

Problem 4. *In the class, we designed a dynamic programming algorithm running in $O(Wn)$ time for the knapsack problem with repetitions. That is given a set of n-items, where each item i has an integer weight $w_i > 0$ kilos and value $v_i > 0$. We want to place items in a knapsack that can hold items with total weight $\leq W$. Our objective was to maximize the value of the items placed in the knapsack, allowing us to use unlimited quantities of each item. Now design an algorithm to maximize the value of the items placed in the knapsack when you can only use an item at most once.*

Problem 5. *Given a set of n positive integers $X = \{x_1, \cdots, x_n\}$, and a positive number t. Design a dynamic programming algorithm running in $O(tn)$ time to find a subset $S \subseteq X$ such that $\sum_{s \in S} s = t$, or report that such a set S does not exist?*

Problem 6. *Consider a queue Q of n people standing shoulder to shoulder. You are asked to pick a subset S of these people so that*
(a) No two people who are consecutive in Q are picked in S.
(b) The sum total of the heights of people in S is maximized.
Design a dynamic programming algorithm for finding an optimal set S.
Do you see a connection with the 1st Problem?

Problem 7. *Let X be a sequence consisting of n elements, where each element is a letter of the English alphabet. Find the longest subsequence of X that is a palindrome. For example, in "Was It A Cat That I Saw", a possible palindromic subsequence is "Was It A Cat I Saw".*

References

[1] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* 3rd. ed., MIT Press, 2009.

[2] S. DasGupta, C. Papadimitriou, V. Vazirani. *Introduction to Algorithms.* McGraw Hill.

[3] A. Maheshwari. *Notes on Algorithm Design*, Chapter 1, `https://people.scs.carleton.ca/~maheshwa/Notes/DAA/notes.pdf`