ARIZONA STATE UNIVERSITY

CSE 310, All Sections — **Data Structures and Algorithms** — Fall 2015

Assignment #3

Available Wednesday, 10/14/2015; due electronically before midnight Friday, 11/06/2015

This assignment covers Chapters 15 (Dynamic Programming) and 16 (Greedy Algorithms) of the text [100 marks total].

1. **Minimum-Sum Descent.** [20 marks total]
   Positive integers are arranged in an equilateral triangle with n numbers in its base such as the one shown in Figure 1 for $n = 4$. The problem is to find the smallest sum in a descent from the apex of the triangle to its base through a sequence of adjacent numbers (shown in the figure by the circles).

   (a) Give C/C++ pseudocode for a dynamic programming algorithm to solve this problem.

   (b) Apply your algorithm to the triangle in Figure 1.

   (c) What is the time and space efficiency of your algorithm?

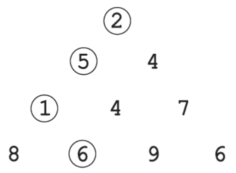

Figure 1: Equilateral triangle with $n = 4$ numbers in its base.

2. **World Series Odds.** [20 marks total]
   Consider two teams, A and B, playing a series of games until one of the teams wins n games. Assume that the probability of A winning a game is the same for each game and equal to p and the probability of A losing a game is $q = 1 - p$. (Hence, there are no ties.) Let $P(i, j)$ be the probability of A winning the series if A needs i more games to win the series and B needs j more games to win the series.

   (a) Set up a recurrence relation for $P(i, j)$ that can be used by a dynamic programming algorithm. *Hint:* In the situation where teams A and B need i and j games, respectively, to win the series, consider the result of team A winning the game and the result of team A losing the game.

   (b) Find the probability of team A winning a seven-game series if the probability of it winning a game is 0.4. *Hint:* Set up a table with five rows ($0 \leq i \leq 4$) and five columns ($0 \leq j \leq 4$) and fill it by using the recurrence derived in part (a).

   (c) Write C/C++ pseudocode for a dynamic programming algorithm to solve this problem and determine its time and space efficiencies. *Hint:* Your pseudocode should be guided by the recurrence you set up in part (a).

3. **Knapsack Problem.** [20 marks]
   Given n items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ and a knapsack of capacity W, the knapsack problem is to find the most valuable subset of the items that fit into the knapsack.

   (a) Write C/C++ pseudocode for a bottom-up dynamic programming algorithm for the knapsack problem.

   (b) Apply your algorithm in part (a) to the following instance of the knapsack problem with a knapsack of capacity $W = 6$:

Item	Weight	Value
1	3	\$25
2	2	\$20
3	1	\$15
4	4	\$40
5	5	\$50

(c) How many different optimal subsets does the instance of part (a) have?

(d) In general, how can we use the table generated by the dynamic programming algorithm to tell whether there is more than one optimal subset for an instance of the knapsack problem?

4. **The Bridge Crossing Problem.** [20 marks total]
There are $n > 1$ people who want to cross a rickety bridge; they all begin on the same side. It is night, and they have one flashlight among them. A maximum of two people can cross the bridge at one time. Any party that crosses, either one or two people, must have the flashlight with them. The flashlight must be walked back and forth; it cannot be thrown, for example. The crossing times in minutes of the n people are $t_1, t_2, \ldots, t_n$, respectively. A pair must walk together at the pace of the slower person. The problem is to minimize the bridge crossing time.

   (a) Give C/C++ pseudocode for a greedy algorithm to get all n people to cross the bridge, and determine how long it will take to cross the bridge by using your algorithm.

   (b) Apply your algorithm to an instance of the problem with $n = 4$ people, with crossing times of $t_1 = 1$ minute, $t_2 = 2$ minutes, $t_3 = 5$ minutes, and $t_4 = 10$ minutes, respectively.

   (c) Does your algorithm yield a minimum crossing time for every instance of the problem? If it does then prove it; if it does not, find a small counterexample.

5. **Scheduling Video Streams.** [20 marks]
Suppose n video streams need to be sent, one after another, over a communication link. Stream i consists of a total of b_i bits to be sent, at a constant rate, over a period of t_i seconds. Two streams cannot be sent at the same time; instead, a *schedule* must be determined, i.e., an order in which to send them. No matter the order, there cannot be any delays between the end of one stream and the start of the next. Assume the schedule starts at time zero (and ends at time $\sum_{i=1}^{n} t_i$, no matter the schedule), and that all the values b_i and t_i are positive integers. In addition, the link imposes the following constraint, using a fixed parameter r: *For each natural number $t > 0$, the total number of bits sent over the time interval from 0 to t cannot exceed rt.* A schedule is *valid* if it satisfies the constraint imposed by the link.

   Given a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, the problem is to determine whether a valid schedule exists.

   Example: Suppose there are $n = 3$ streams with $(b_1, t_1) = (2000, 1)$, $(b_2, t_2) = (6000, 2)$, and $(b_3, t_3) = (2000, 1)$, and the link parameter is $r = 5000$. Then the schedule that runs the streams in the order $1, 2, 3$ is valid because the constraint is satisfied: At $t = 1$: The whole first stream is sent; $2000 < 5000 \cdot 1$. At $t = 2$: Half of the second stream is sent; $2000 + 3000 < 5000 \cdot 2$. Similarly for $t = 3$ and $t = 4$.

   (a) **Claim:** There exists a valid schedule if and only if each stream i satisfies $b_i \leq rt_i$. Prove this claim true or false.

   (b) Give C/C++ pseudocode for a *greedy* algorithm that takes a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, and determines whether a valid schedule exists.

   (c) Clearly explain what makes your algorithm greedy.

   (d) What is the worst case running time of your algorithm?