CS 325 - Winter 2020
Homework #4

Problem 1.
Write a concrete example of the knapsack problem where you specify a set of *at least* 5 objects, their dollar values (i.e., benefits) and their weights, as well as the weight of the knapsack, denoted W. Now, consider the greedy approach of *sorting items based on decreasing benefit/weight ratios and picking items from the beginning of the list.* In the context of your example, show that

Problem 1.a. (2 points)

- The greedy approach works for fractional knapsack.

Problem 1.b. (2 points)

- The greedy approach may fail for 0-1 knapsack.

Problem 2.
Consider the following symbols with their corresponding frequencies:

$A : 1, B : 1, C : 2, D : 3, E : 5, F : 8, G : 13, H : 21$

Problem 2.a. (3 points)

- Construct the Huffman coding of these symbols along with its optimal coding tree.

Problem 2.b. (3 points)

- Use your coding tree to decode 000100100001000000000001001

Problem 3.
Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

Problem 3.a. (4 points)

- Suppose that the available coins are in the denominations that are powers of c, i.e., the denominations are $c^0, c^1, \cdots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm of *picking the largest denomation first* always yields an optimal solution. You are expected to reason about why this approach gives an optimal solution. (*Hint*: Show that for each denomination c^i, the optimal solution must have less than c coins.)

Problem 3.b. (4 points)

- Design an $O(nk)$ time algorithm that makes change for any set of k different coin denominations, assuming that one of the coins is a penny.

Problem 4. (7 points)
Implementation: Implement the make change algorithm you designed in the previous problem. Your program should read a text file "data.txt" where each line in "data.txt" contains three values c, k and n. Please make sure you take your input in the specified order c, k and n. For example, a line in "data.txt" may look like the following:

3 4 38

where $c = 3, k = 4, n = 38$. That is, the set of denominations is $\{3^0, 3^1, 3^2, 3^3, 3^4\} = \{1, 3, 9, 27, 81\}$, and we would like to make change for $n = 38$. The file "data.txt" may include multiple lines like above.

The output will be written to a file called "change.txt", where the output corresponding to each input line contains a few lines. Each line has two numbers, where the first number denotes a denomination and the second number represents the cardinality of that denomination in the solution. For example, for the above input line '3 4 38', the optimal solution is the multiset $\{27, 9, 1, 1\}$, and the output in the file "change.txt" is as follows:

27 1

9 1

1 2

which means the solution contains 1 coin of denomination 27, one coin of 9 and two coins of denomination 1. You can use a delimiter line to separate the outputs generated for different input lines.

Problem 5. (3 points)
Extra credit: Can you generalize the results you found in the construction of Problem 2 (i.e., the Huffman coding tree)? Write a statement/theorem that captures your generalization.

Submit a copy of all your code files and a README file that explains how to compile and run your code in a ZIP file to TEACH. We will only test execution with an input file named data.txt.