CSCI 705 Algorithms
Homework 7
Due 4/12/2024

Saad Mneimneh
Computer Science
Hunter College of CUNY

Readings
Based on lectures for Dynamic programming and greedy algorithms.

Problem 1
Consider the problem of partitioning an integer m into a sequence of k integers, each in $\{1,\ldots,n\}$. For instance, here are the partitions of 8 into 3 parts, each in $\{1,2,3,4\}$.

$$1,3,4$$
$$1,4,3$$
$$2,2,4$$
$$2,3,3$$
$$2,4,2$$
$$3,1,4$$
$$3,2,3$$
$$3,3,2$$
$$3,4,1$$
$$4,1,3$$
$$4,2,2$$
$$4,3,1$$

Let $p(m, k)$ be the number of partitions given m, k, and n.

(a) Find a recursive definition of $p(m, k)$

(b) Compute $p(m, k)$ by implementing a recursive function that mimics the recursive definition.

(c) Compute $p(m, k)$ using memoization. You will need a two dimensional array to store the values of $p(m, k)$.

Problem 2

Consider a coin system with the d values $c_0 = 1 < c_1 < \ldots < c_{d-1}$. Given a target n, we would like to find the smallest number of coins that add up to n. In other words, we would like to determine k_i, $i = 0 \ldots d-1$ such that

$$\sum_{i=0}^{d-1} c_i k_i = n$$

and $\sum_{i=0}^{d-1} k_i$ is minimized.

(a) Let $C[m]$, $m \leq n$, be the minimum number of coins that add up to m. Explore the optimal substructure of this problem and come up with a dynamic programming algorithm to solve the coin problem described above. Provide

- pseudocode for the algorithm to determine the number of coins
- pseudocode for the backtracking
- analysis of the algorithm in terms of time and space

(b) Show that when $c_0 = 1, c_1 = 5, c_2 = 10, c_3 = 25$, we have a greedy strategy of choosing the largest coin first. Convert your dynamic programming solution to a greedy algorithm that runs in $O(n)$ time (and not $O(dn)$ time).

(c) Do some research online for a condition that determines, given $c_0, \ldots, c_{d-1}$, whether the coin system admits a greedy strategy or not. You may consult `http://www.cs.cornell.edu/~kozen/papers/change.pdf`. Describe an algorithm to test the condition that runs in $O(dc_{d-1})$ time.

Problem 3

There are n people teaming up for a hiking trip. Each person can carry two bags, so there are $2n$ bags in total. Assume bag i has weight b_i. There are many reasonable (or unreasonable) ways one could optimize the assignment of bags to people. Parts (a) and (b) explore two alternatives, the two questions are independent of each other.

(a) Describe an algorithm to assign each person two bags in such a way to minimize the heaviest load. In other words, if person k is carrying bag i and bag j, let $w_k = b_i + b_j$ be the sum of the two weights. We want to keep the largest w at a minimum.

(b) Every person will carry one bag in each hand. We want both hands to be well balanced, so if person k is carrying bag i and bag j, let $d_k = |b_i - b_j|$ be the difference between the two weights. Describe an algorithm that minimizes the largest difference, i.e. that keeps the largest d at a minimum.