CS5050 Advanced Algorithms

Spring Semester, 2018

Assignment 5: Dynamic Programming

Due Date: 3:00 pm, Tuesday, Apr. 3, 2018 (**at the beginning of CS5050 class**)

Note: For each of the following problems, you will need to design a dynamic programming algorithm. When you describe your algorithm, please explain clearly the *subproblems* and the *dependency relation* of your algorithm.

1. **(20 points)** The knapsack problem we discussed in class is the following. Given an integer M and n items of sizes $\{a_1, a_2, \ldots, a_n\}$, determine whether there is a subset S of the items such that the sum of the sizes of all items in S is exactly equal to M. We assume M and all item sizes are positive integers.

   Here we consider the following *unlimited version* of the problem. The input is the same as before, except that there is an unlimited supply of each item. Specifically, we are given n item sizes $a_1, a_2, \ldots, a_n$, which are positive integers. The knapsack size is a positive integer M. The goal is to find a subset S of items (to pack in the knapsack) such that the sum of the sizes of the items in S is exactly M and each item is allowed to appear in S multiple times.

   For example, consider the following sizes of four items: $\{2, 7, 9, 3\}$ and $M = 14$. Here is a solution for the problem, i.e., use the first item once and use the fourth item four times, so the total sum of the sizes is $2 + 3 \times 4 = 14$ (alternatively, you may also use the first item 4 times and the fourth item 2 times, i.e., $2 \times 4 + 3 \times 2 = 14$).

   Design an $O(nM)$ time dynamic programming algorithm for solving this unlimited knapsack problem. For simplicity, you only need to determine whether there exists a solution (namely, if there exists a solution, you do not need to report the actual solution subset).

2. **(20 points)** This is a problem from a student during his interview with Goldman Sachs in Salt Lake City.

   Given a set A of n positive integers $\{a_1, a_2, \ldots, a_n\}$ and another positive integer M, find a subset of numbers of A whose sum is *closest* to M. In other words, find a subset A' of A such that the absolute value $|M - \sum_{a \in A'} a|$ is minimized, where $\sum_{a \in A'} a$ is the total sum of the numbers of A'. For the sake of simplicity, you only need to return the sum of the elements of the solution subset A' without reporting the actual subset A'.

   For example, suppose $A = \{1, 4, 7, 12\}$ and $M = 15$. Then, the solution subset is $A' = \{4, 12\}$, and thus your algorithm only needs to return $4 + 12 = 16$ as the answer.

   Let K be the sum of all numbers of A. Design a dynamic programming algorithm for the problem and your algorithm should run in $O(nK)$ time (note that it is not $O(nM)$).

3. **(20 points)** Here is another variation of the knapsack problem. We are given n items of sizes $a_1, a_2, \ldots, a_n$, which are positive integers. Further, for each $1 \leq i \leq n$, the i-th item a_i has a positive value $value(a_i)$ (you may consider $value(a_i)$ as the amount of dollars the item is worth). The knapsack size is a positive integer M.

   Now the goal is to find a subset S of items such that the sum of the sizes of all items in S is **at most** M (i.e., $\sum_{a_i \in S} a_i \leq M$) and the sum of the values of all items in S is **maximized** (i.e., $\sum_{a_i \in S} value(a_i)$ is maximized).

   Design an $O(nM)$ time dynamic programming algorithm for the problem. For simplicity, you only need to report the sum of the values of all items in the optimal solution subset S and you do not need to report the actual subset S.

4. **(20 points)** Given an array $A[1 \ldots n]$ of n distinct numbers (i.e., no two numbers of A are equal), design an $O(n^2)$ time dynamic programming algorithm to find a *longest monotonically increasing subsequence* of A. Your algorithm needs to report not only the length but also the actual longest subsequence (i.e., report all elements in the subsequence).

   Here is a formal definition of a *longest monotonically increasing subsequence of* A (refer to the example given below). First of all, a *subsequence* of A is a subset of numbers of A such that if a number a appears in front of another number b in the subsequence, then a is also in front of b in A. Next, a subsequence of A is *monotonically increasing* if for any two numbers a and b such that a appears in front of b in the subsequence, a is smaller than b. Finally, a *longest monotonically increasing subsequence of* A refers to a monotonically increasing subsequence of A that is longest (i.e., has the maximum number of elements).

   For example, if $A = \{20, 5, 14, 8, 10, 3, 12, 7, 16\}$, then a longest monotonically increasing subsequence is $5, 8, 10, 12, 16$. Note that the answer may not be unique, in which case you only need to report one such longest subsequence.

Total Points: 80