Homework Assignment #4

Due Time/Date

2:20PM Tuesday, October 4, 2022. Late submission will be penalized by 20% for each working day overdue.

How to Submit

Please use a word processor or scan hand-written answers to produce a single PDF file. Name your file according to this pattern: "b107050xx-hw4". Upload the PDF file to the NTU COOL site for Algorithms 2022. You may discuss the problems with others, but copying answers is strictly forbidden.

Problems

There are five problems in this assignment, each accounting for 20 points. (Note: problems marked with "(X.XX)" are taken from [Manber 1989] with probable adaptation.)

1. (5.7) Write a program (or modify the code discussed in class) to recover the solution (i.e., enumerate the elements in the solution) to a knapsack problem using the *belong* flag. You should make your algorithm as efficient as possible.

2. (5.8) In algorithm *Knapsack*, we first checked whether the ith item is unnecessary (by checking $P[i-1,j]$). If there is a solution with the $i-1$ items, we take this solution. We can also make the opposite choice, which is to take the solution with the ith item if it exists (i.e., check $P[i-1,j-k_i]$ first). Which version do you think will have a better performance? Redraw Fig. 5.11 (see notes/slides) to reflect this choice.

3. (5.17) The Knapsack Problem that we discussed in class is defined as follows. Given a set S of n items, where the ith item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

   We have described in class an algorithm to solve the problem. Modify the algorithm to solve a variation of the knapsack problem where each item has an *unlimited* supply. In your algorithm, change the type of $P[i,k].belong$ into integer and use it to record the number of copies of item i needed.

4. (5.20) Let x_1, x_2, …, x_n be a set of integers, and let $S = \sum_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm (presented in suitable pseudocode) should run in time $O(nS)$.

5. (5.23) Write a non-recursive program (in suitable pseudocode) that prints the moves of the solution to the towers of Hanoi puzzle. The three pegs are respectively named A, B, and C, with n (generalizing the original eight) disks of different sizes stacked in decreasing order on peg A.