Algorithms Final Practice Problems

1. You are given an ordered sequence of n cities, and the distances between every pair of cities. You must partition the cities into two subsequences (not necessarily contiguous) such that person A visits all cities in the first subsequence (in order), person B visits all cities in the second subsequence (in order), and such that the sum of the total distances travelled by A and B is minimized. Assume that person A and person B start initially at the first city in their respective subsequences. Design a polynomial time algorithm for producing such a partition.

2. You have n_1 items of size s_1 and n_2 items of size s_2. You must pack all of these items into bins, each of capacity C, such that the total number of bins used is minimized. Design a polynomial time algorithm for such packaging.

3. The Integer Knapsack Problem (Duplicate Items Permitted). You have n types of items, where the i^{th} item type has an integer size s_i and a real value v_i. You are trying to fill a knapsack of total capacity C with a selection of items of maximum value. You can add multiple items of the same type to the knapsack.

4. Maximum Value Contiguous Subsequence. Given a sequence of n real numbers $A_1 \ldots A_n$, determine a contiguous subsequence $A_i \ldots A_j$ for which the sum of elements in the subsequence is maximized.

5. Making Change. You are given n types of coin denominations of values $v_1 < v_2 < \ldots < v_n$ (all integers). Assume $v_1 = 1$, so you can always make change for any amount of money C. Give an algorithm which makes change for an amount of money C with as few coins as possible.

6. Longest Increasing Subsequence. Given a sequence of n real numbers $A_1 \ldots A_n$ determine a subsequence (not necessarily contiguous) of maximum length in which the values in the subsequence form a strictly increasing sequence.

7. Box Stacking. You are given a set of n types of rectangular 3-D boxes, where the i^{th} box has height h_i, width w_i and depth d_i (all real numbers). You want to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is also allowable to use multiple instances of the same type of box.

8. Building Bridges. Consider a 2-D map with a horizontal river passing through its center.

There are n cities on the southern bank with x-coordinates $a_1 \ldots a_n$ and n cities on the northern bank with x-coordinates $b_1 \ldots b_n$. You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cities, you are only allowed to connect the the i^{th} city on the northern bank to the i^{th} city on the southern bank.

9. Integer Knapsack Problem (Duplicate Items Forbidden). This is the same problem as the example above, except here it is forbidden to use more than one instance of each type of item.

10. Balanced Partition. You have a set of n integers each in the range 0 ... K. Partition these integers into two subsets such that you minimize |S1 — S2|, where S1 and S2 denote the sums of the elements in each of the two subsets.

11. Edit Distance. Given two text strings A of length n and B of length m, you want to transform A into B with a minimum number of operations of the following types: delete a character from A, insert a character into A, or change some character in A into a new character. The minimal number of such operations required to transform A into B is called the edit distance between A and B.

12. Counting Boolean Parenthesizations. You are given a boolean expression consisting of a string of the symbols 'true', 'false', 'and', 'or', and 'xor'. Count the number of ways to parenthesize the expression such that it will evaluate to true. For example, there is only 1 way to parenthesize 'true and false xor true' such that it evaluates to true.

13. The emergency services are responding to a major earthquake that has hit a wide region, and left *n* people injured who need to be sent to a hospital. Let *P* be the set of *n* people and *H* be the set of *k* hospitals. Several hospitals are available to treat these people, but there are some constraints:

    (a) Each injured person needs to be sent to a hospital no further than one hour drive away. Let H_p be the set of hospitals that are within range for person *p*.

    (b) Each hospital *h* has a capacity c_h, the maximum number of people that the hospital can receive.

    Develop an efficient algorithm that determines whether it is possible to assign each person to a hospital in a way that satisfies these constraints, and returns such an assignment if so.

14. Assume each student can borrow at most 10 books from the library, and the library has three copies of each title in its inventory. Each student submits a list of books he wishes to borrow. You have to assign books to students, so that a maximal number of volumes is checked out.

15. Given a flow graph with *n* vertices and *e* edges with their capacities, a source *s* and a sink *t,* design an efficient algorithm which finds a minimal cut in the graph.

16. Optimal Strategy for a Game. Consider a row of n coins of values v(1) ...
v(n), where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

17. **(For the extended classes)** Assume you are given a table *n* by *n* containing integers. At every step you can flip the sign of all numbers in a single column or a single row. Design an algorithm that transforms this table into a table that has the property that the sum of numbers in every row and every column is non-negative.