Final Practice Problems: Dynamic Programming and Max Flow Problems
(I) Dynamic Programming Practice Problems

To prepare for the final first of all ***study carefully all examples of Dynamic Programming which we did in class****. When confronted with a new problem, always first check if this might be something that you already know how to solve (or very similar to something you already know how to solve). For example, try to determine which problems done in the lectures are hiding behind the following two problems: (Number of points indicates approximate difficulty level of each problem.)*

1. Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a_1 \dots a_n$ and n cities on the northern bank with x-coordinates $b_1 \dots b_n$. 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.

2. You have an amount of money of M cents and you are in a candy store. There are n kinds of candies and for each candy you know how much pleasure you get by eating it, which is a number between 1 and 100. You also know the price of each candy, which is an integer number of cents. Your task is to chose which candies you are going to buy to maximise the total pleasure you will get by gobbling them all. (5pt)

   Even if not exactly the same, sometimes a solution to a problem might be obtainable as a relatively simple modification of what you already know how to do. For example, having in mind the construction to generate maximal increasing subsequence done in class, try to solve the following problems.

3. You are given a set of n types of rectangular boxes, where the i^{th} box has height h_i, width w_i and depth d_i. 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. (10pts)

4. You are traveling by a canoe down a river and there are n trading posts along the way. Before starting your journey, you are given for each $1 \leq i < j \leq n$ the fee $F(i, j)$ for renting a canoe from post i to post j. These fees are arbitrary. For example it is possible that $F(1, 3) = 10$ and $F(1, 4) = 5$. You begin at trading post 1 and must end at trading post n (using rented canoes). Your goal is to design an efficient algorithms which produces the sequence of trading posts where you change your canoe which minimises the total rental cost. (10pt)

5. Given a sequence of n real numbers $A_1 \dots A_n$, determine *in linear time* a contiguous subsequence $A_i \dots A_j$ for which the sum of elements in the subsequence is maximised. (10pts)

   Similarly, keeping in mind the matrix multiplication problem try solving the following problem:

6. You are given a boolean expression consisting of a string of the symbols *true* and *false* and with exactly one operation and, or, xor between any two consecutive truth values. Count the number of ways to place brackets in the expression such that it will evaluate to true. For example, there is only 1 way to place parentheses in the expression *true* and *false* xor *true* such that it evaluates to true.

   Sometimes dynamic programming might look like brute force, but closer inspection reveals that only the recursion step involves some exhaustive search. Try for example solving the following problem.

7. You have n_1 items of size s_1, n_2 items of size s_2, and n_3 items of size s_3. You would like to pack all of these items into bins, each of capacity C, using as few bins as possible.

   Sometimes dynamic programming solution is just a straightforward recursion. Try solving the following problems.

8. You are given an $n \times n$ chessboard with an integer in each of its n^2 squares. You start from the top left corner of the board; at each move you can go either to the square immediately below or to the square immediately to the right of the square you are at the moment; you can never move diagonally. The goal is to reach the right bottom corner so that the sum of integers at all squares visited is minimal.

   (a) Describe an algorithm which always correctly finds a minimal sum path and runs in time $O(n^2)$. (5pt)

   (b) Describe an algorithm which computes the number of such minimal paths. (7pt)

9. There are N lily pads in a row. A frog starts on the leftmost lily pad and wishes to get to the rightmost one. The frog can only jump to the right. There are two kinds of jump the frog can make:

   - The frog can jump 3 lily pads to the right (skipping over two of them)
   - The frog can jump 5 lily pads to the right (skipping over four of them)

   Each lily pad has some number of flies on it. Design an algorithm that maximises the total number of flies on lily pads the frog lands on getting to the rightmost lily pad. (10pt)

   Recall that both the Bellman Ford and the Floyd Warshall algorithms involved the idea of relaxation. Try to use the same idea to solve the following problem:

10. Find the number of partitions of n, i.e. the number of sets of integers $\{p_1, p_2, \ldots, p_k\}$ such that such that $p_1 + p_2 + \ldots, p_k = n$. *(Try limiting how large the largest number j appearing in a partition of a number $i \leq n$ can be.)*

   You might want to review the longest common subsequence problem before trying this one.

11. A palindrome is a sequence of at least three letters which reads equally from left to right and from right to left.

    (a) Given a sequence of letters S, find efficiently its longest subsequence (not necessarily contiguous) which is a palindrome. Thus, we are looking for a longest palindrome which can be obtained by crossing out some of the letters of the initial sequence without permuting the remaining letters. (20pt)

    (b) Find the total number of occurrences of all subsequences of S which are palindromes of any length ≥ 3. (25pt)

    Here are some more problems to try.

12. Consider a row of n coins of values $v_1 \ldots v_n$, 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.

13. Devise a dynamic programming algorithm that counts the number of non-decreasing sequences of integers of length N, such that the numbers are between 0 and M inclusive.(20pt)

14. You are given a rooted tree. Each edge of the tree has a cost for removing it. Devise an algorithm to compute the minimum total cost of removing edges to disconnect the root from all the leaves of the tree. (20 pt)

15. 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 minimised. (30pt)

(II) Max Flow Problems

For some flow networks we do not need sophisticated algorithms to find a max flow, as in the following problem.

16. You are given a tree which is a source, capacities of all edges of the tree and all leaves acting as sinks. Find a fast algorithm for finding the max flow in such a flow network.

    One can add capacity constraint to vertices as well, not only edges:

17. You are given a flow network where not only edges have capacities, but also each node has a capacity of flow that can go through the node. Design an algorithm which finds a maximal flow through such a network.

    Next problem is related to the previous one.

18. (Escape Problem) This is a problem which is faced by people who design printed circuit boards. An $n \times n$ grid is an undirected graph consisting of n rows, each row containing n vertices, with vertices connected to all of their immediate neighbours (2 at all of the 4 corners, 3 at all of the 4 sides and 4 in the interior of the grid. The escape problem is, given $m \leq n^2$ many vertices in the grid, connect them with m many distinct vertices located on the 4 sides by non intersecting paths or return "impossible" when there is no such a solution.

    Sometimes, some of the information is actually spurious, as in the following example (do you really need the capacities of edges?)

19. Assume that you are the administrator of a network of computers; each computer is connected by unidirectional fiberoptic cables of the same capacity to a few other computers on the same network (so the network can be modelled by a directed graph). You noticed that computers $P_1, P_2, ...P_n$ are mounting an attack on computers $Q_1, Q_2, ..., Q_m$. The total number of computers on the network is $N > m + n$. Since it is a real emergency, you must disconnect some of the optical cables of the network so that none of computers $P_1, P_2, ...P_n$ can send packets to any of $Q_1, Q_2, ..., Q_m$. Since you must send crews to disconnect some of the fiberoptic cables, for each cable c_{ij} for traffic from a computer X_i to a computer X_j there is an associated cost c_{ij} for disconnecting it. Your task is to design an algorithm for determining which cables to disconnect to isolate computers $Q_1, Q_2, ..., Q_m$ from all of the computers $P_1, ..., P_n$ so that the total cost incurred is minimal.

    Some very standard problems reducible to max flow, as the following ones.

20. You work for a new private university which wants to keep the sizes of classes small. Each class is assigned its maximal capacity - the largest number of students which can enrol in it. Students pay the same tuition fee for each class they get enrolled in. Students can apply to be enrolled in as many classes as they wish, but each of them will eventually be enrolled to at most 5 classes at any given semester. You are given the wish lists of all students, containing for each student the list of all classes they would like to enrol this particular semester and you have to chose from the classes they have put on their wish lists in which classes you will enrol them, without exceeding the maximal enrolment of any of the classes and without enrolling any

student into more than 5 classes. Your goal is, surprisingly, to maximise the income from the tuition fees for your university. Design an efficient algorithm for such a task.

21. 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.

22. 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.