Design and Analysis of Algorithms
Problem Sheet 2

Dynamic Programming:

Problem 1:
Given an array of n positive integers $a_1, a_2, \ldots, a_n$, give an algorithm to find the length of the longest subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ where $1 \leq a_{i_1} < a_{i_2} < \ldots < a_{i_k} \leq n$ such that each term divides the next term, that is, a_{i_j} divides $a_{i_{j+1}}$ for all $1 \leq j < k$.

Problem 2:
Given a $n \times m$ grid of integers where the rows are numbered from 1 to n and the columns are numbered from 1 to m. Your task is to answer queries of the form (i, j, k, l) for different $1 \leq i, j \leq n, 1 \leq k, l \leq m$. The answer to such a query is the sum of the numbers in the sub-grid bounded by the i, jth rows and the k, lth columns. Preprocess the grid so as to efficiently answer each such query in constant time.

Problem 3:
Given an array of n integers $a_1, a_2, \ldots, a_n$. You start with a score of 0. In each step, you are allowed to choose an index i different from 1 and n and delete a_i from the array. The value of $a_{i-1} \times a_{i+1}$ gets added to your score. After deleting a_i, the new array is of length $n-1$ and is $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$. The process stops when you have only 2 elements remaining. Compute the maximum possible score you can get.

Problem 4:
Given a set S of n distinct positive integers $a_1, \ldots, a_n$ and a positive integer M. Give an algorithm to compute the number of subsets T of S such that the sum of elements of T is exactly M.

Problem 5:
Given an array of n distinct integers $a_1, a_2, \ldots, a_n$, give an algorithm to find the longest continuous mountain present in the array, that is, the longest sub-array $a_i, \ldots, a_j$ such that there exists an index $k, i \leq k \leq j$ such that the sequence $a_i, \ldots, a_k$ is strictly increasing and the sequence $a_k, \ldots, a_j$ is strictly decreasing.
A sub-array is a contiguous portion of the array.

Greedy:

Problem 6:
Given n integers on the real line, $a_1, a_2, \ldots, a_n$, you need to cover all of them using closed intervals of length 1. ie. you need to place some intervals on the real line such that every a_i is contained within at least

one of the closed intervals. Give an algorithm to find the minimum number of such intervals that you need.

Problem 7:
You work in a pizza shop which offers a wide range of toppings. The toppings are stored in separate jars which are kept in a cupboard (assume that the jars have unlimited capacity). But the cupboard is far from the counter, and so you keep some jars in a nearby desk. Unfortunately the desk is small and you can keep at most S jars on it at any given time. So when a customer asks for a topping which is not on your desk, and if your desk is full, you will have to return some jar back to the cupboard and replace it with the necessary jar. Miraculously you know the exact sequence of toppings that will be demanded by your customers, which is $a_1, a_2, \ldots, a_n$. Each a_i corresponds to a topping. Describe an algorithm to find the least number of trips that you will have to make to the cupboard. You can start off with any S jars on the desk. Assume $1 \leq a_i \leq n$.

Problem 8:
You are given 2 integers, l and r. You need to find two numbers L and R, which satisfy $l \leq L \leq R \leq r$, and which maximize the bit-wise XOR of L and R.

A bitwise XOR takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1.

Divide and Conquer:

Problem 9:
Given n integers, $a_1, a_2, \ldots, a_n$, you need to find the sub-array with the maximum sum. A sub-array is a contiguous portion of the array, and it's sum is just the sum of all the array values which it covers.

Give a Divide and Conquer based algorithm to solve this. Try to come up with a linear time solution (not Divide and Conquer based).

Problem 10:

You are given n integers, $a_1, a_2, \ldots, a_n$, in an array. A majority element of this array is any element occurring in more than $\lceil \frac{n}{2} \rceil$ positions. Assume that elements cannot be ordered or sorted, but can be compared for equality. (You might think of the elements as chips, and there is a tester that can be used to determine whether or not two chips are identical.) Design a divide and conquer algorithm to find a majority element in this (or determine that no majority element exists)