List of Abbreviations and Symbols

$A[1..n]$	An array indexed from 1 to n of n elements.
$\mathbb{N}$	Set of all natural numbers, i.e., $\{1, 2, 3, \dots\}$.
$\mathbb{R}$	Set of all real numbers.
$\mathbb{Z}$	Set of all integers, i.e., $\{\dots, -2, -1, 0, 1, 2, \dots\}$.
$\exists$	symbol meaning *there exists*.
$\forall$	symbol meaning *for all*.

Modifiers

To help you with what problems to try, problems marked with **[K]** are key questions that tests you on the core concepts, please do them first. Problems marked with **[H]** are harder problems that we recommend you to try after you complete all other questions (or perhaps you prefer a challenge). Good luck!!!

Contents

§1 Optimisation problems

Exercise 1.1. [K] You are given n intervals on an axis. The ith interval $[l_i, r_i)$ has integer endpoints $l_i < r_i$ and has a score of s_i. Your task is to select a set of disjoint intervals with maximum total score. Note that if intervals i and j satisfy $r_i = l_j$ then they are still disjoint. Design an algorithm which solves this problem and runs in $O(n^2)$ time.

The input consists of the positive integer n, as well as $2n$ integers $l_1, r_1, \ldots, l_n, r_n$ and n positive real numbers $s_1, \ldots, s_n$. The output is the set of intervals chosen, organised in any format or data structure.

Example

For example, if $n = 4$ and the intervals are:

$$\begin{array}{lll} l_1 = 0 & r_1 = 3 & s_1 = 2 \\ l_2 = 1 & r_2 = 3 & s_2 = 1 \\ l_3 = 2 & r_3 = 4 & s_3 = 4 \\ l_4 = 3 & r_4 = 5 & s_4 = 3 \end{array}$$

then you should select only the first and fourth intervals, for a maximum total score of 5. Note that interval 3 is not disjoint with any other interval.

Exercise 1.2. [K] Due to the recent droughts, n proposals have been made to dam the Murray river. The ith proposal asks to place a dam x_i meters from the head of the river (i.e., from the source of the river) and requires that there is not another dam within r_i metres (upstream or downstream). Design an algorithm that returns the maximal number of dams that can be built, you may assume that $x_i < x_{i+1}$ for all $i = 1, \ldots, n-1$.

Exercise 1.3. [K] 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. Design an $O(n^2)$ algorithm that returns the maximum height of the stack of boxes.

Hint — *How can we reduce this problem into a simpler problem without rotations?*

Exercise 1.4. [K] You have n_1 items of size s_1 and n_2 items of size s_2. You would like to pack all of these items into bins, each of capacity $C > s_1, s_2$, using as few bins as possible. Design an algorithm that returns the minimal number of bins required.

Exercise 1.5. [K] Consider a row of n coins of values $v_1, v_2, ...v_n$, 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.

Exercise 1.6. [K] Given a sequence of n positive or negative integers $A_1, A_2, \ldots, A_n$, determine a contiguous subsequence A_i to A_j for which the sum of elements in the subsequence is maximised.

Exercise 1.7. [H] Skiers go fastest with skis whose length is close to their height. Consider n skiers with heights $h_1, h_2, \ldots, h_n$, gets a delivery of $m \geq n$ pairs of skis, with lengths $l_1, l_2, \ldots, l_m$. Design an algorithm to assign to each skier one pair of skis to minimise the sum of the absolute differences between the height h_i of the skier and the length of the corresponding ski they got, i.e., to minimise

$$S = \sum_{i=1}^{n} \left| h_i - l_{s(i)} \right|$$

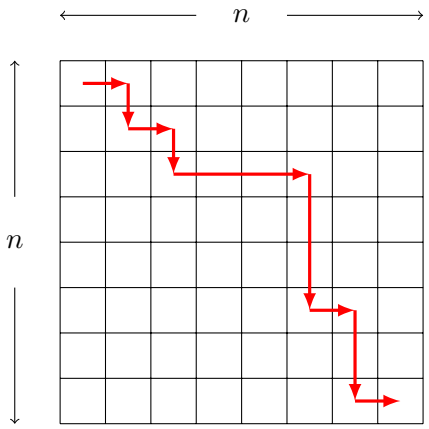
where $s(i)$ is the index of the skies assigned to the skier of height h_i.

Exercise 1.8. [H] You have been handed responsibility for a business in Texas for the next n days. Initially, you have K illegal workers. At the beginning of each day, you may hire an illegal worker, keep the number of illegal workers the same or fire an illegal worker. At the end of each day, there will be an inspection. The inspector on the i^{th} day will check that you have between l_i and r_i illegal workers (inclusive). If you do not, you will fail the inspection. Design an algorithm that determines the fewest number of inspections you will fail if you hire and fire illegal employees optimally.

Exercise 1.9. [H] A company is organising a party for its employees. The organisers of the party want it to be a fun party, and so have assigned a fun rating to every employee. The employees are organised into a strict hierarchy, i.e. a tree rooted at the president. There is one restriction, though, on the guest list to the party: an employee and their immediate supervisor (parent in the tree) cannot both attend the party (because that would be no fun at all). Give an algorithm that makes a guest list for the party that maximises the sum of the fun ratings of the guests.

Exercise 1.10. [H] You have to cut a wood stick into several pieces. The most affordable company, Analog Cutting Machinery (ACM), charges money according to the length of the stick being cut. Their cutting saw allows them to make only one cut at a time. It is easy to see that different cutting orders can lead to different prices. For example, consider a stick of length 10 m that has to be cut at 2, 4, and 7 m from one end. There are several choices. One can cut first at 2, then at 4, then at 7. This leads to a price of $10 + 8 + 6 = 24$ because the first stick was of 10 m, the resulting stick of 8 m, and the last one of 6 m. Another choice could cut at 4, then at 2, then at 7. This would lead to a price of $10 + 4 + 6 = 20$, which is better for us. Your boss demands that you design an $O(n^2)$ algorithm to find the minimum possible cutting cost for any given stick.

Exercise 1.11. [H] 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 a greedy algorithm which attempts to find such a minimal sum path and show by an example that such a greedy algorithm might fail to find such a minimal sum path.

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

(c) Describe an algorithm which computes the number of such minimal paths.

(d) Assume now that such a chessboard is stored in a read only memory. Describe an algorithm which always correctly finds a minimal sum path and runs in linear space (i.e., amount of read/write memory used is $O(n)$) and in time $O(n^2)$.

> **Hint —** *Combine divide and conquer with dynamic programming.*

§2 Enumeration problems

Exercise 2.1. Some people think that the bigger an elephant is, the smarter it is. To disprove this, you want to analyse a collection of elephants and place as large a subset of elephants as possible into a sequence whose weights are increasing but their IQ's are decreasing. Design an $O(n \log n)$ algorithm which, given the weights and IQ's of n elephants, will find a longest sequence of elephants such that their weights are increasing but IQ's are decreasing.

> **Hint —** *How does this relate to the longest subsequence problem?*

Exercise 2.2. [K] There are n levels to complete in a video game. Completing a level takes you to the next level, however each level has a secret exit that lets you skip to another level later in the game. Determine if there is a path through the game that plays exactly K levels.

Exercise 2.3. [K] A partition of a number n is a sequence $\langle p_1, p_2, \ldots, p_t \rangle$ (we call the p_k *parts*) such that $1 \leq p_1 \leq p_2 \leq \ldots \leq p_t \leq n$ and such that $p_1 + \ldots + p_t = n$. Define numpart(k, n) to be the number of partitions of n such that every part is at most k.

(a) Explain why numpart$(1, n) = 1$ for each n.

(b) Devise a recursion to determine the number of partitions of n in which every part is smaller or equal than k, where n, k are two given numbers such that $2 \leq k \leq n$.

(c) Hence, find the total number of partitions of n.

Exercise 2.4. [K] Given an array of n positive integers, find the number of ways of splitting the array up into contiguous blocks of sum at most k in $O(n^2)$ time.

Exercise 2.5. [H] You are given a boolean expression consisting of a string of the symbols *true* (T) and *false* (F) and with exactly one operation of *and* ($\wedge$), *or* ($\vee$) or *xor* ($\oplus$) between any two consecutive

symbols(truth values). Count the number of ways to place brackets in the expression such that it will evaluate to true.

Example 2.1 (Simple expression)
There are 2 ways to place parentheses in the expression

$$T \wedge F \oplus T$$

such that it evaluates to true, namely $T \wedge (F \oplus T) \iff T$ and $(T \wedge F) \oplus T \iff T$.

Exercise 2.6. [H] We are given a checkerboard which has 4 rows and n columns, and has an integer written in each square. We are also given a set of $2n$ pebbles, and we want to place some or all of these on the checkerboard (each pebble can be placed on exactly one square) so as to maximise the sum of the integers in the squares that are covered by pebbles. There is one constraint: for a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares (diagonal adjacency is fine).

(a) Determine the number of legal *patterns* that can occur in any column (in isolation, ignoring the pebbles in adjacent columns) and describe these patterns.

Call two patterns *compatible* if they can be placed on adjacent columns to form a legal placement. Let us consider sub-problems consisting of the first k columns $1 \leq k \leq n$. Each sub-problem can be assigned a type, which is the pattern occurring in the last column.

(b) Using the notions of compatibility and type, give an $O(n)$-time algorithm for computing an optimal placement.

§3 Knapsack and related problems

Exercise 3.1. [K] You have an amount of money M 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 K, as well as the price of each candy. Your task is to choose which candies you are going to buy to maximise the total pleasure you will get by gobbling them all.

Exercise 3.2. [K] Your shipping company has just received N shipping requests (jobs). For each request i, you know it will require t_i trucks to complete, paying you d_i dollars. You have T trucks in total. Out of these N jobs you can take as many as you would like, as long as no more than T trucks are used total. Devise an efficient algorithm to select jobs that will bring you the largest possible amount of money.

Exercise 3.3. [K] Again your shipping company has just received N shipping requests (jobs). This time, for each request i, you know it will require e_i employees and t_i trucks to complete, paying you d_i dollars. You have E employees and T trucks in total. Out of these N jobs you can take as many of them as you would like, as long as no more than E employees and T trucks are used in total. Devise an efficient algorithm to select jobs which will bring you the largest possible amount of money.

§4 Strings

Exercise 4.1. [K] We say that a string A occurs as a subsequence of another string B if we can obtain A by deleting some of the letters of B. Design an algorithm that for two strings A and B gives the number of

different occurrences of A in B, i.e., the number of ways one can delete some of the symbols of B to get A.

Example 4.1

The string $A =$ "*ba*" has three occurrences as a subsequence of string $B =$ "*baba*":

"$\underline{ba}ba$, $b\underline{ab}a$, $\underline{b}ab\underline{a}$".

Exercise 4.2. [K] For bit strings $X = x_1 \dots x_n$, $Y = y_1 \dots y_m$ and $Z = z_1 \dots z_{n+m}$, we say that Z is an interleaving of X and Y if it can be obtained by interleaving the bits in X and Y in a way that maintains the left-to-right order of the bits in X and Y.

Example

If $X = 101$ and $Y = 01$ then $x_1\, x_2\, y_1\, x_3\, y_2 = 10011$ is an interleaving of X and Y, whereas 11010 is not.

Design an efficient algorithm to determine if Z is an interleaving of X and Y.

Exercise 4.3. [H] A palindrome is a sequence of at least two letters which reads equally from left to right and from right to left.

Example 4.2

The string "*ababa*" is a palindrome of size 5.

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

Exercise 4.4. [K] Consider a 2-D map with a horizontal river passing through its centre. 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. Design an $O(n^2)$ algorithm to determine the maximum number of such pairs.

Hint — *Consider how the longest common subsequence problem can help us here.*

Exercise 4.5. [H] A *context-free grammar* is a tuple $\langle V, \Sigma, R, S \rangle$, where:

- V is a set of variables;
- Σ is a finite set of symbols used to define characters of a word;
- R is a set of production rules that determines how the strings can be generated by the grammar;
- S is the start variable.

The rules of a grammar come in the form:

- $A \to \alpha$ where A is a variable and α is a symbol,
- $A \to BC$ where A, B, C are variables and B, C are not the start variable,

- $S \to \epsilon$ where ϵ is the empty character.

To generate a string, we use the production rules given above and keep performing substitutions until we get to a point where there are no variables. That is, we repeatedly replace variables with the corresponding symbol or variable given by the production rules.

(a) Consider the context-free grammar, $G = \langle V, \Sigma, R, S \rangle$, where

- $V = \{A, B, C\}$,
- $\Sigma = \{0, 1\}$,
- R is given by the production rules:
  - $A \to \epsilon$ or $A \to BC$,
  - $B \to 0$ or $B \to BB$,
  - $C \to 1$ or $C \to CC$,
- $S = A$.

We can generate the string 01 because, starting with $A \to BC$, we can replace B with 0 and C with 1. Thus, we generate $A \to 01$ and so, G generates 01. Using these production rules, find another string that can be generated by G.

We now attempt to solve the membership problem; that is, given an arbitrary context-free grammar G and a string w, does G generate w?

(b) Suppose that w is a symbol. Devise an $O(1)$ algorithm to determine if G generates w.

(c) Now suppose that w is a string of length 2. Devise a recursion to determine if G generates w.

(d) Now suppose that w is a string of length n. Devise an $O(n^3)$ algorithm to determine if G generates w.

> **Note** — Note that this formally depends on the size of the grammar but the size of the grammar is independent of the length of the string, so we treat the size of the grammar as a fixed constant.

§5 Shortest paths

Exercise 5.1. [K] You are travelling by 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 algorithm that produces the sequence of trading posts where you change your canoe which minimizes the total rental cost.