Worth: 12% **Due:** By 9:59pm on **Thursday 16 October**

Remember to write the *full name* and *student number* of *every group member* prominently on your submission.

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes, and materials available directly on the course webpage). For example, indicate clearly the **name** of every student from another group with whom you had discussions, the **title** of every additional textbook you consulted, the **source** of every additional web document you used, etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

---

1. The *simple knapsack problem* is: Given a knapsack with weight capacity W, and a collection of weights, select the heaviest possible subcollection of the weights which fit in the knapsack.

   More formally, the problem is defined as follows:

   Input: Positive integers $w_1, \ldots, w_n, W$.

   Output: A subset $S \subseteq \{1, \ldots, n\}$ such that $K = \sum_{i \in S} w_i$ is as large as possible subject to the contraint $K \leqslant W$.

   Give a greedy algorithm which solves the simple knapsack problem for the case that each weight w_i is a power of 2.

   Then, write a detailed proof that your algorithm is correct.

2. A *minimum heavyweight spanning tree* (MHT) is a spanning tree in which the heaviest edge is as light as possible.

   More formally, the minimum heavyweight spanning tree problem is defined as follows:

   Input: A connected undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{N}$.

   Output: A spanning tree T_ℓ for G with the property that *every* spanning tree T for G has some edge e' with $w(e') \geqslant w(e)$ for every edge $e \in T_\ell$.

   (a) Give an efficient greedy algorithm solving this problem, analyze its running time, and prove that your algorithm is correct.

   (b) Is every mininum heavyweight spanning tree a minimum spanning tree? Justify your answer.

3. In bioinformatics, a common task involves determining *alignment* between two genes (represented as strings over the alphabet $\Sigma = \{A, C, G, T\}$). For example, a possible alignment of $x = ATGCC$ with $y = TACGCA$ is:

$$\begin{array}{cccccccc} - & A & T & - & G & C & C \\ T & A & - & C & G & C & A \end{array}$$

   As you can see, this involves writing both strings in columns so that

- the characters of each string appear in order,
- each column contains a character from at least one string,
- subject to the previous constraint, columns can contain "gaps" (represented as "–").

The score of a possible alignment is specified through a "scoring matrix" δ that gives a value for each possible column in the alignment. In our example, the total score would be equal to:

$$\delta(-,T)+\delta(A,A)+\delta(T,-)+\delta(-,C)+\delta(G,G)+\delta(C,C)+\delta(C,A).$$

Write an efficient algorithm that takes as inputs two strings $x, y \in \{A, C, G, T\}^*$ with a $[5 \times 5]$ scoring matrix δ, and that returns the highest-scoring alignment between x and y. (Assume that $\delta(-,-) = -\infty$, representing the fact that no alignment can align two gaps together.)

Give a rigorous argument that your algorithm is correct and analyse its running time.

Hint: You may find it useful to read Section 15.4 of the textbook.

4. An edge in a flow network is called *critical* if decreasing the capacity of this edge reduces the maximum possible flow in the network.

   Give an efficient algorithm that finds a critical edge in a network. Give a rigorous argument that your algorithm is correct and analyse its running time.