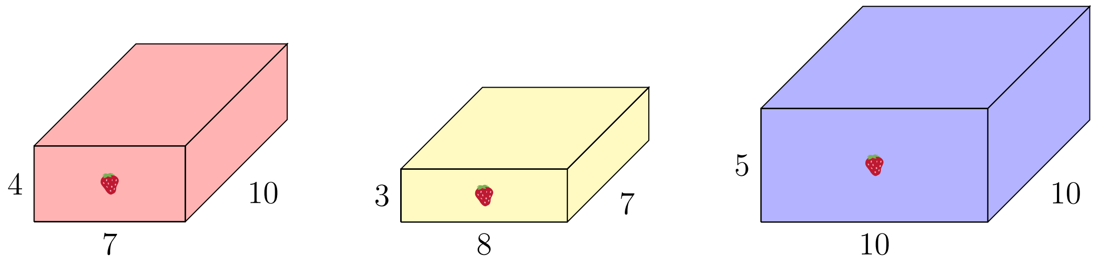
Layer Cake

This is a **regular task**. You must submit a PDF, which can be produced using the LaTeX template on Moodle, exported from a word processor, hand-written or any other method.

Theodore is baking an extravagant layer cake for a wedding. He has made n rectangular layers, the ith of which has length l_i, width w_i and height h_i. Each layer has a strawberry which must be on the front and upright, preventing layers from being rotated, and no two cakes have both the same width and the same length.

A valid layer cake is a stacking of layers such that no layer is wider or longer than the layer below it, otherwise the cake will fall apart. Theodore wants to create the tallest cake that he can, using the layers he has already made.

Here is an example with $n = 3$ layers. All cakes are drawn with height up the page, width across the page, and length into the page.



Either of the first two cakes (red and yellow) can be stacked on top of the blue cake, because both of them have length and width at most 10. However, the blue cake cannot be stacked on top of either the other cakes without overhanging. Further, the red and yellow cakes cannot be stacked on top of each other, because

- the yellow cake is wider (8) than the red cake (7), and
- the red cake is longer (10) than the yellow cake (7).

Note that because of the strawberry which must appear on the front, we cannot rotate the yellow cake by 90°, even though doing so would make it fit on the red cake.

Therefore, because the yellow cake is shorter than the red cake, the best we can do is to put the blue cake on the bottom, and the red cake on top of it, leading to an overall height of 9.

(a) The cakes in the question are not given in any particular order. However, if we choose an appropriate ordering to the cakes, we can turn the problem into one of finding a *subsequence* of the cakes to stack. That is, we want to find an order for the cakes so that *every valid way to stack up the cakes appears as a subsequence in our order.*

In the above example, both [blue, red, yellow] and [blue, yellow, red] are valid orders. However, [red, blue, yellow] is not valid because we can create a wedding cake by stacking red on blue, yet [blue, red] is not a subsequence in this order.

Choose an order for the cakes where every valid stack appears as a subsequence. Make sure the ordering is described unambiguously, and prove a brief justification for why it works.

Hint: If it is possible to stack cake i on top of j, then i needs to appear later than j. How can we order the cakes to make sure of this?

(b) Assuming the ordering you provided in part (a), what property must a subsequence of this ordering have to form a valid layer cake?

(c) Now, we will design an algorithm to find the subsequence with the maximum total height that forms a valid wedding cake. This is now very similar to *longest increasing subsequence* from the lectures. As such, we will try to use a dynamic programming algorithm with a similar structure.

Longest increasing subsequence: Find the subsequence with the maximum *length* such that *each element is larger than the previous.*

Layer cake: Find the subsequence with the maximum *total height* such that *your condition from part (b) holds.*

Define an appropriate subproblem for this problem, and complete the algorithm by defining the recurrence, base case(s), final answer, and order of computation, and analyse the time complexity.

Advice.

- For part (c), make sure to carefully specify the subproblem definition, what the parameter(s) in the recurrence mean, what the base cases are, and what, if any, additional computation needs to be performed to get to the final answer from the recurrence values.

Expected length: Up to one page in total.