13. tutorial

1. Hill. Say we have a sequence of integers $a_1, \ldots, a_n$. A *hill* is a subsequence which first increases and then decreases. Design an algorithm, which in a given sequence finds a longest hill (as a subsequence).

2. Library 1. Let us have a sequence of n books. Each book has a width w_i and a height h_i. We want to arrange the books in a library with several shelves such that we preserve the alphabetical order of the books. So, the alphabetically first few books go on the first shelf, the next on the second shelf, and so on. We are given a library width W, and we want to arrange the books in shelves in such a way that each book has its place and the total height of the library is as small as possible. (Don't count the height of the shelves.)

3. Library 2. The problem is similar as before. However, now we are given a maximum allowed library height H, and the task is to find the minimum possible width W. If you struggle, start by assuming all books have width 1.

4. Transmission. You have decyphered a secret transmission, but the spaces are missing. All is not lost – we have a dictionary containing all the words which may occur in the transmission. The task at hand is thus to split the message into the fewest possible number of dictionary words.
(For example, `ARTISTOIL` can be split as `ART·IS·TOIL`, but splitting it as `ARTIST·OIL` contains fewer words and thus is a better solution.)

5. Edit Distance - Space. The algorithm computing edit distance we discussed in the lecture takes $\Theta(nm)$ memory cells to store the table. Show how to decrease the necessary memory to $\Theta(n+m)$.

6. Similar Strings. A first impression may suggest that if input strings A, B are very similar, it should be easier to compute their edit distance. However, our algorithm always fills in the whole table. Show how to speed it up to compute in time $\mathcal{O}((n+m)(L(A,B)+1)$ (where $L(A,B)$ is the edit distance between A, B, because it is also called the *Levenshtein* distance.)

---

7. Average Complexity. Prove that $\Omega(\log n)$ and $\Omega(n \log n)$ comparisons are needed for searching and sorting, respectively, not only in the worst case but also on average, where the average is taken over all possible inputs. In the case of search, the average is taken over all possible x being searched; for sorting, the average is taken over all permutations.

8. Matrix search. An $n \times n$ matrix A of integers is given, and it satisfies the property that each row and column forms an increasing sequence. How to quickly find indices i, j such that $A_{i,j} = i, j$? If there are multiple solutions, it suffices to report one of them. We don't count the time needed to load the matrix into memory.
Hint: Look at the lower left corner of the matrix. What is implied by $A_{i,j} < i + j$ or $A_{i,j} > i + j$?