Exercise 1.
Efficient searching with suffix arrays

In the lecture we discussed two strategies how to reduce the number of redundant character comparisons during a binary search. One uses the mlr values, while the other one makes use of lcp values. The mlr trick in practice already brings the running time to $O(m + \log n)$.

- Find a pair of pattern and text where the mlr trick still needs time $O(m \log n)$.
- For the same text and pattern perform the binary search using the lcp values.

Exercise 2.
Efficient searching with suffix arrays:

- Prove that using the lcp method the search algorithm does at most $O(m + \log n)$ character comparisons.

Exercise 3.
Prove the following assumption stated in the script:

- For the fixed binary search tree used in the search for LP and RP compute the lcp values for its internal nodes using the array height. The value at an internal node is the minimum of its successors.

Exercise 4.
Suffix array construction:

- What is the worst-case runtime (number of character comparison) when the suffix array is computed with the quicksort algorithm?

Exercise 5.
SWIFT algorithm:

- The Lemma 2.2 in the SWIFT script contains a formula to compute $w$ (for a $w \times e$ parallelogram). Show that every local alignment with $\tau$-hits and $e$ errors lies in a $w \times e$ parallelogram.