In the exercises we use two string matching problems, consisting of a text $T$ and a string $P$. Problem 1 is as follows:

$$
T = aaabaadaaba
P = aabaaa
$$

Problem 2 is as follows:

$$
T = 000010001010001
P = 000101
$$

1. Apply Dijkstra’s algorithm to the following graph (take for the direction from lower to higher in the alphabet), taking $A$ as source node. Give updates step by step by annotating the graph or by giving a table.

![Graph](image)

2. Write down for yourself an analysis of the time complexity of Dijkstra’s shortest path algorithm. Use the pseudo-code as in the book (which is the same as in the slides), and consider different implementations of a priority queue.

3. Apply the brute-force pattern matching algorithm to the first pattern matching problem.

4. Apply the brute-force pattern matching algorithm to the second pattern matching problem.

5. The worst-case time complexity of the brute-force pattern matching algorithm is in $O(m \cdot (n - m + 1))$. Give an example of a text $T$ of length $n$ and a pattern $P$ of length $m$ such that the brute-force pattern matching algorithm indeed performs $m \cdot (n - m + 1)$ comparisons of characters. This illustrates that the bound is actually tight.
6. Assume we have a pattern $P$ of length $m$ with all $m$ characters different. Can we optimize the brute-force pattern matching algorithm for this special case? What is the time complexity of the optimized algorithm?

7. Assume we allow in the pattern $P$ the special symbol $\ast$, which matches with an arbitrary string (of length 0 or larger). We may have an arbitrary number of occurrences of $\ast$ in $P$, but we do not have occurrences of $\ast$ in $T$.

Example: The pattern $ab\ast ba\ast c$ occurs as follows in the text $cabccbacbacab$:

```
c  ab  cc  ba  cba  c  ab
   ab  *  ba  *  c
```

Give a pattern matching algorithm for such patterns and texts, and analyze the time complexity of your algorithm.

8. (a) Give the failure function (or prefix function) of the Knuth–Morris–Pratt pattern matching algorithm for the pattern $P$ from the first pattern matching problem.

(b) Apply the Knuth–Morris–Pratt pattern matching algorithm to the first pattern matching problem.

9. Repeat the previous exercise for the second pattern matching algorithm.

10. Explain why the algorithm for computing the prefix function is in $O(m)$.

11. Explain informally what is the time complexity of the Knuth–Morris–Pratt algorithm.