Suchalgorithmen **Prüfungsstoff**

1. Du kannst die Suche in einem Array als Entscheidungs- und als Berechnungsproblem beschreiben.

2. Sequentielle Suche (Synonym: lineare Suche)

   (a) Du kannst ein Python-Programm schreiben, das eine lineare Suche nach einem Element `x` in einer Liste `A` ausführt.

   (b) Du kannst die Situationen beschreiben, in denen bei der sequentielle Suche

   - der Best Case,
   - der Avarage Case,
   - der Worst Case

   eintritt und die jeweilige Laufzeitkomplexität angeben.

3. Binäre Suche

   (a) Du kannst das Verfahren (und dessen Voraussetzung) beschreiben und es auf einfachen Arrays nachvollziehen.

   (b) Du kannst ein Python-Programm schreiben, das die binäre Suche nach einem Element `x` auf einer Liste `L` implementiert.

   (c) Du kannst die Situationen beschreiben, in der bei der binären Suche der Worst Case eintritt und die zugehörige Laufzeitkomplexität angeben.

   (d) Du kannst aus gegebenen Worst Case-Laufzeiten die maximale Lösungsdauer für sequentielle und binäre Suchprobleme abschätzen.

4. Du kannst das String-Matching als Entscheidungs- und als Berechnungsproblem beschreiben.

5. String Matching allgemein

   Du kannst mindestens drei Anwendungen von String Matching Algorithmen aufzählen.

6. Naives String Matching

   (a) Du kannst das naive Verfahren für das String-Matching-Problem beschreiben und anhand einfacher Beispieldaten tabellarisch dessen Ablauf zeigen sowie die Anzahl der für die einzelnen Schritte benötigten Vergleiche ermitteln.

   (b) Du kannst die Worst Case-Laufzeit des naiven Verfahrens angeben sowie ein Beispiel, in dem diese auftritt.

7. Boyer-Moore-Horspool (BMH)

   (a) Du kannst für eine Textmuster p die Bad-Character-Tabelle bestimmen und die Laufzeitkomplexität dafür angeben.

   (b) Du kannst mit Hilfe der Bad-Character-Tabelle den Ablauf des BMH-Algorithmus auf einfachen Beispieldaten zeigen.

   (c) Du kannst die Average- und Worst Case-Laufzeit von BMH angeben.