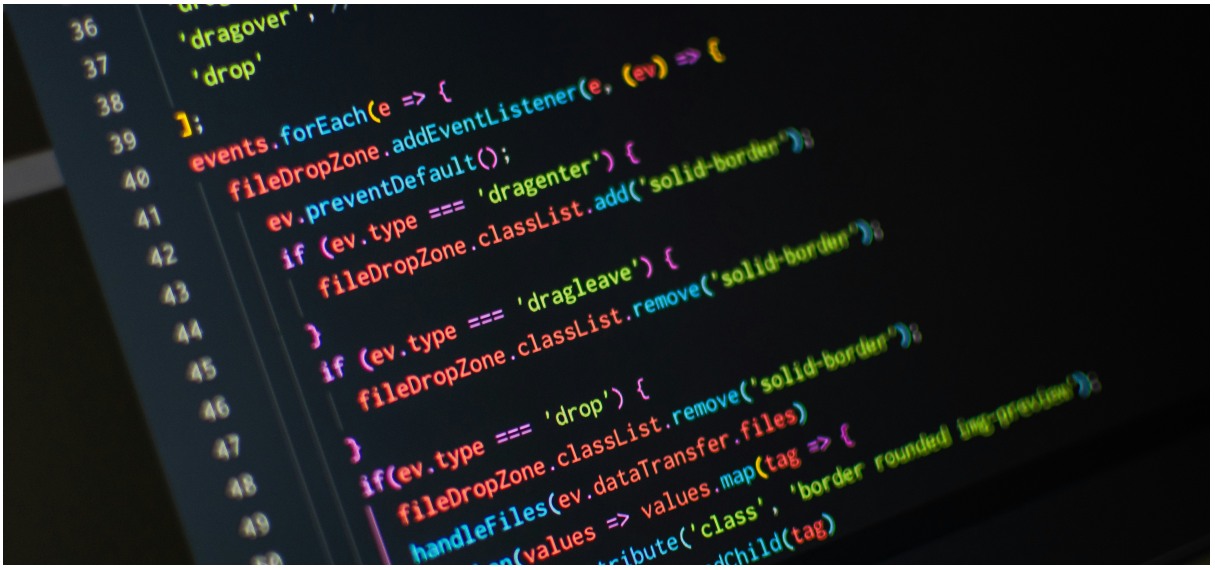

String Matching

Algoritmia I

1. Using the algorithm Rabin-Karp find all the T[s,..., s+m] that have the same hash value than the pattern P.

```
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

/**
 * This class provides methods for pattern matching using the
 * The Rabin-Karp algorithm is used to find all occurrences o
 */
public class Rabin Karp {

  private static final int PRIME = 101;

  /**
   * This method Find all occurrences of a pattern in a text
   *
   * @param pattern the pattern to search for
   * @param text    the text in which to search for the patte
   * @return a list of indices where the pattern is found in
   */
```

```
  public static List<Integer> rabinKarp(String pattern, Strin
    int patLen = pattern.length(), txtLen = text.length();
    long patHash = generateHash(pattern, patLen);

    return IntStream.range(0, txtLen - patLen + 1)
            .filter(i -> {
              long txtHash = generateHash(text.substring(i, i
              return patHash == txtHash && pattern.equals(tex
            })
            .boxed()
            .collect(Collectors.toList());
  }

  /**
   * This method generate a hash value for a string using the
   *
   * @param str    the string for which to generate the hash
   * @param length the length of the string
   * @return the hash value of the string
   */
  private static long generateHash(String str, int length) {
    return IntStream.range(0, length)
            .mapToLong(i -> str.charAt(i) * (long)Math.pow(PR
            .sum();
  }
}
```

```
  /**
   * The Main method to demonstrate the usage of the Rabin-Ka
   * It searches for occurrences of a given pattern within a
   */
  public static void main(String[] args) {
    String pattern = "aba";
    String text = "abacaba";
    List<Integer> matches = rabinKarp(pattern, text);

    System.out.println("Substrings matching the pattern:");
```

```
    matches.forEach(match -> System.out.println(text.substrin
  }
```

```
Pattern:
aba
text:
abacaba
```

```
Sample Input:
abba bab
4 1 2 3 4
```

2. Given a string S, print the prefix that matches the most in S. If there are multiple answers, print the longest prefix. Print the word and how many times it matches. (E.g. S = abababc, output: ab - 3)

```
import java.util.HashMap;
import java.util.Map;

/**
 * This class provides a method to find the most frequent seq
 * It calculates the frequency of all possible subsequences a
 */
public class Task2 {

  /**
   * This method finds the most frequent sequence in the give
   *
   * @param S the input string
   * @return the most frequent sequence along with its freque
   */
  public static String findMostFrequentSequence(String S) {
    Map<String, Integer> sequenceCounts = new HashMap<>();

    for (int i = 0; i < S.length(); i++) {
      for (int j = i + 1; j <= S.length(); j++) {
        String sequence = S.substring(i, j);
        sequenceCounts.put(sequence, sequenceCounts.getOrDefa
      }
    }
```

```
    String mostFrequentSequence = "";
    int maxFrequency = 0;
    for (Map.Entry<String, Integer> entry : sequenceCounts.en
      String sequence = entry.getKey();
      int frequency = entry.getValue();
      if (frequency > maxFrequency || (frequency == maxFreque
        mostFrequentSequence = sequence;
        maxFrequency = frequency;
      }
    }

    return mostFrequentSequence + " - " + maxFrequency;
  }
}
```

```
  /**
   * This Main method to find the most frequent sequence in a
   *
   * @param args command line arguments (not used)
   */
  public static void main(String[] args) {
    String S = "abababc";
    String result = findMostFrequentSequence(S);
    System.out.println("Output: " + result);
  }
```

```
Sample Input:
abababc
```

```
Sample Output:
ab - 3
```

1. Given a text T, a pattern P, and a number x ($x \leq |T|$) followed by x numbers. For each T[1,...,x] find the longest suffix of T[1,...,x] that matches the longest prefix of P.

```
import java.util.Scanner;

/**
```

```
 * This class provides functionality to find the longest suff
 * that matches the longest prefix of a given pattern P, for
 */
public class Task3 {

  /**
   * This method calculates the length of the longest prefix
   *
   * @param pattern The input pattern.
   * @return The length of the longest prefix.
   */

  private static int longestPrefixLength(String pattern) {
    int length = 0;
    for (int i = 1; i < pattern.length(); i++) {
      if (pattern.substring(0, i).equals(pattern.substring(pa
        length = i;
      }
    }
    return length;
  }

  /**
   * This method finds the length of the longest match betwee
   *
   * @param text The input text.
   * @param pattern The input pattern.
   * @param prefixLength The length of the longest prefix of
   * @return The length of the longest match.
   */

  private static int longestMatchLength(String text, String p
    for (int i = prefixLength; i > 0; i--) {
      String suffix = text.substring(text.length() - i);
      String prefix = pattern.substring(0, i);
      if (suffix.equals(prefix)) {
        return i;
      }
```

```
    }
    return 0;
  }
 }
```

```
  /**
   * This method reads input data, calculates the length of t
   */
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    String T = scanner.next();
    String P = scanner.next();
    int x = scanner.nextInt();

    int[] numbers = new int[x];
    for (int i = 0; i < x; i++) {
      numbers[i] = scanner.nextInt();
    }

    int prefixLength = longestPrefixLength(P);

    for (int i = 0; i < x; i++) {
      int length = numbers[i];
      String substring = T.substring(0, length);
      int matchLength = longestMatchLength(substring, P, pref
      System.out.println(matchLength);
    }
  }
```

```
Sample Input:
abba bab
4 1 2 3 4
```

```
Sample Output:
0
1
1
2
```