# Aufgabe 1: Aufwand der Mustersuche

Auf Folie 32 in PRG1_ALG6 finden Sie den Source Code für eine einfache Mustersuche.

1. Implementieren Sie diesen Code in Java und testen Sie Ihren Code mit Hilfe einer Testklasse.

```
/**
 * Write a description of class TextSearch here.
 * @author Christian Bontekoe, Felix Rohrer
 * @version 1.0
 */
public class TextSearch
{
    /**
     * Constructor for objects of class TextSearch
     */
    public TextSearch()
    {
    }

    public static int textSearch(String text, String pattern)
    {
        for(int i=0; i < (text.length()-pattern.length()+1); i++) {
            boolean success = true;
            for(int j=0; j < pattern.length(); j++) {
                if (text.charAt(i+j) != pattern.charAt(j)) {
                    success = false;
                    break;
                }
            }
            if(success) {
                return i;
            }
        }
        return -1;
    }
}

/**
 * The test class TextSearchTest.
 * @author  Christian Bontekoe, Felix Rohrer
 * @version 1.0
 */
public class TextSearchTest extends junit.framework.TestCase
{
    /**
     * Default constructor for test class TextSearchTest
     */
    public TextSearchTest()
    {
    }

    /**
     * Sets up the test fixture.
     *
     * Called before every test case method.
     */
    public void setUp()
    {
    }

    /**
     * Tears down the test fixture.
     *
     * Called after every test case method.
     */
    public void tearDown()
    {
    }

    /**
     * Test textsuche
     */
    public void testAppointment()
    {
        TextSearch test1 = new TextSearch();
        assertEquals(11, test1.textSearch("abcaaabcacaabcaabaababcabcaaa", "abcaab"));
        assertEquals(-1, test1.textSearch("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa", "aaaab"));
    }
}
```

2. Ergänzen Sie den Code damit sie die Anzahl Vergleiche ausgeben können.

```
/**
 * Write a description of class TextSearch here.
 *
 * @author Christian Bontekoe, Felix Rohrer
 * @version 1.0
 */
public class TextSearch
{

    /**
     * Constructor for objects of class TextSearch
     */
    public TextSearch()
    {
    }

    public static int textSearch(String text, String pattern)
    {
        int loopcount = 0;
        for(int i=0; i < (text.length()-pattern.length()+1); i++) {
            boolean success = true;
            for(int j=0; j < pattern.length(); j++) {
                loopcount++;
                if (text.charAt(i+j) != pattern.charAt(j)) {
                    success = false;
                    break;
                }
            }
            if(success) {
                System.out.println("Compare count: " + loopcount);
                return i;
            }
        }
        System.out.println("Compare count: " + loopcount);
        return -1;
    }
}


test1.textSearch("abcaaabcacaabcaabaabababcabcaaa", "abcaab")       ➔ Compare count: 30
test1.textSearch("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa", "aaaab"));      ➔ Compare count: 125
```

3. Nun erstellen Sie einen Testfall (Text und Muster), damit die Anzahl Vergleiche Maximal wird (siehe dazu Folie 33).
*Siehe Aufgabe 1.*

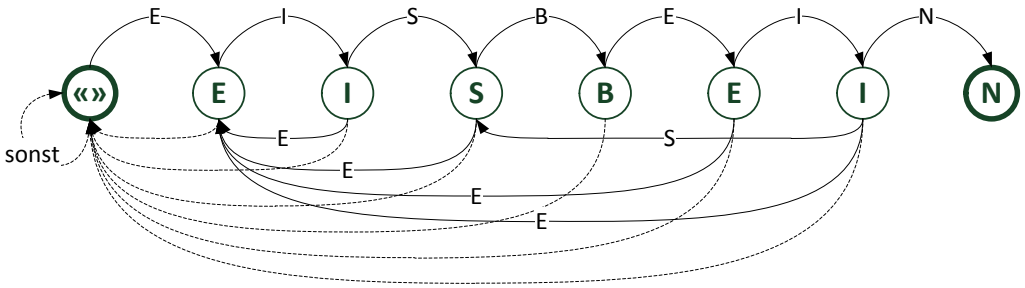
   Wie viele Vergleiche sind notwendig (Zeitkomplexität des Algorithmus)?
*Laufzeitkomplexität: O(n·m)*
*Text hat die Länge n, Muster hat die Länge m*
*n = 30, m = 5*

4. Wie viele Vergleiche benötigt dieser Algorithmus im Idealfall, wenn das gesuchte Pattern im Text nicht vorkommt?
*n, die Länge vom Text*

5. Überprüfen sie den Idealfall mit einem geeigneten TESTFALL (Text und Muster).
*test1.textSearch("aaaaaaaaaaaaaaaaaaaaaaaaaaaaa", "b")  ➔ Compare count: 29*

# Aufgabe 2: Zustandsautomat

1. Zeichnen Sie einen endlichen Automaten, um das Wort "EISBEIN" zu suchen.




# Aufgabe 3: Musterautomat

1. Bestimmen Sie die Ränder aller Teilworte von "EISBEIN".

| Teilwort | Rand | Länge des Randes |
|---|---|---|
| *E* | ∅ | *0* |
| *EI* | ∅ | *0* |
| *EIS* | ∅ | *0* |
| *EISB* | ∅ | *0* |
| *EISBE* | *E* | *1* |
| *EISBEI* | *EI* | *2* |

2. Zeichnen Sie einen "Musterautomaten" für das Muster "EISBEIN".